Genesys Voice Platform 8.1

Application Migration Guide

About Genesys

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

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

Notice

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

Your Responsibility for Your System's Security

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

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. 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.

Technical Support 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.genesyslab.com

Document Version: 81gvp_amg_07-2013_v8.1.701.00

Table of Contents

List of Procedures

Preface

Welcome to the *Genesys Voice Platform 8.1 Application Migration Guide*. This document provides detailed information about the application modifications that are required to use legacy Genesys Voice Platform (GVP) 7.6 Voice Extensible Markup Language (VoiceXML) and Telera XML (TXML) applications in GVP 8.1.

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

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

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

About GVP

GVP is a group of software components that constitute a robust, carrier-grade voice processing platform. GVP unifies voice and web technologies to provide a complete solution for customer self-service or assisted service.

In the Voice Platform Solution (VPS), GVP 8.1 is fully integrated with the Genesys Management Framework. GVP uses the Genesys Administrator, the standard Genesys configuration and management graphical user interface (GUI), to configure, tune, activate, and manage GVP processes and GVP voice and call control applications. GVP interacts with other Genesys components, and it can be deployed in conjunction with other solutions, such as Enterprise

Routing Solution (ERS), Network Routing Solution (NRS), and Network-based Contact Solution (NbCS).

Note: GVP is a scalable solution with flexible configuration and deployment options, but release 8.1 of GVP is available only for single-tenant configurations.

Intended Audience

This document is primarily intended for anyone who writes voice applications for GVP 8.1, as well as system integrators and administrators. It has been written with the assumption that you have a basic understanding of:

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

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

Making Comments on This Document

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

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

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

Contacting Genesys Technical Support

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

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

Document Change History

This section lists topics that are new or that have changed significantly since the first release of this document.

GVP 8.1.7

- Chapter 2, "Migrating to GVP 8.1 GVPi," on page 25:
  - Corrected the value of `SET VARNAME="$tts-gender$"` in Table 3, "Mapping Legacy AppID.xml File Parameters to the IVR Profile," on page 31.
- Chapter 5, "Migrating From Legacy VoiceGenie to GVP 8.1 NGI," on page 75:
  - Added a note about enabled VCR controls to the section "bargein Property" on page 89.

GVP 8.1.3

- New Chapter 5 on page 75 describes migration from legacy Voice Genie to GVP 8.1 NGI.

GVP 8.1.2

- Hierarchical Multi-Tenancy (HMT) is supported in release 8.1.2. References to that limitation have been amended accordingly.
- Post-connect CPA is supported with GVPi in release 8.1.2. References to that limitation have been amended accordingly.
- Information about AT&T Out-of-Band (OOB) Transfer Connect functionality has been added in release 8.1.2.
- Information about the PSTN-Connector, a new GVP component, has been added in release 8.1.2

GVP 8.1.1

- Information about AT&T Transfer functionality has been added in release 8.1.1
- Information about changes to Full Call Recording (FCR) functionality has been added in release 8.1.1.
- Information about RFC 5552 compliance has been added in release 8.1.1.
- Information about NGI support for offboard DTMF recognition has been added in release 8.1.1.
- Genesys Composer Voice has been renamed Genesys Composer in release 8.0.2 and references have been amended accordingly.

Chapter

1 Introduction

This chapter introduces the Genesys Voice Platform (GVP) interpreters and the Genesys Administrator, the graphical user interface (GUI) for configuring and managing GVP. It also provides a conceptual context for designing voice and call control applications, as well as a high-level description of the features that GVP supports.

This chapter contains the following sections:

About GVP Interpreters

The Next Generation Interpreter (NGI) and the Legacy GVP Interpreter (GVPi) are Voice Extensible Markup Language (VoiceXML) interpreter components on the Media Control Platform. The CCXML Interpreter (CCXMLI) is the Call Control Extensible Markup Language (CCXML) interpreter on the Call Control Platform used for executing call control applications.

VoiceXML Interpreters

The VoiceXML interpreters request VoiceXML 2.0 or VoiceXML 2.1 pages from a web application server (optionally through a fetching/caching proxy), compile the pages into an internal representation, and then execute them in order to manage a dialog with a user. As part of this dialog management process, the VoiceXML interpreter also requests resources (such as speech recognition and speech synthesis sessions) from other platform components.

The interpreters are responsible for driving the underlying platform to execute the VoiceXML application. The interpreters interpret the VoiceXML applications to determine the interactions that occur with a caller, and the Media Control Platform provides the media services.

The VoiceXML interpreters are Windows dynamic link libraries (DLLs) or Linux Shared Objects that run in-process on the Media Control Platform. In GVP 8.1, the GVPi is available only for Windows.

For Windows deployments, the Media Control Platform can run one or both VoiceXML interpreters (NGI and GVPi), and both are installed by default. Voice application, or IVR Profile, provisioning determines which interpreter will be used for a particular voice application. You can also specify which interpreter will be the default VoiceXML interpreter for GVP.

The following subsections briefly describe the GVP 8.1 interpreters, to provide a context for the syntactic and semantic differences between the applications that they support.

The Next Generation Interpreter (NGI)

The NGI is the default VoiceXML interpreter for voice applications running on GVP 8.x. It was introduced with GVP 8.0, and is built on a scalable architecture that leverages multicore/multiprocessor environments. This architecture provides higher levels of performance, and positions the NGI for support of evolving standards, such as VoiceXML 3.

The NGI parses VoiceXML documents in stricter accordance with the VoiceXML and Speech Synthesis Markup Language (SSML) schemas, with GVP extensions. Any element or attribute that violates the schemas generates a parsing error. For more information about NGI support for the VoiceXML and SSML schemas and GVP extensions, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

In GVP 8.1, the NGI supports Linux as well as Windows.

The Legacy GVP Interpreter (GVPi)

The GVPi, which is new in GVP 8.1, is effectively the legacy GVP 7.6.x VoiceXML interpreter that was present in the IP Communication Server (IPCS).

The GVPi is a mature software component that handles millions of calls every day in deployments around the world. However, although it is highly tuned and delivers industry-leading performance, the GVPi does not scale linearly under high port density.

Genesys recognizes the substantial investment that legacy voice applications represent for customers. Therefore, Genesys is providing the GVPi to enable customers to reuse legacy GVP 7.6 voice applications in GVP 8.1 deployments.

In addition to VoiceXML 2.0 and 2.1 applications, the GVPi can process XML applications that use Telera XML (TXML) extensions for call control functionality. TXML call control functions include creating outbound call legs or bridging calls without using the `<transfer>` tag; queuing calls; and managing the call legs.

For more information about GVPi support for VoiceXML and for TXML call control, see the *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual*.

CCXML Interpreter (CCXMLI)

The Call Control Platform is an optional GVP component that provides call control functionality such as accepting, rejecting, or redirecting calls, and creating new sessions. The Call Control Platform can also use Media Control Platform services to initiate VoiceXML dialogs (for example, for transfers).

The CCXMLI, which runs in-process on the Call Control Platform, is responsible for driving the underlying platform to execute the CCXML application. The CCXMLI parses CCXML documents in accordance with the W3C CCXML standard. For more information about GVP support for the CCXML schema, see the *Genesys Voice Platform 8.1 CCXML Reference Manual*.

In GVP 8.1, the CCXMLI supports Linux as well as Windows.

Types of Voice Applications

The role and behavior of GVP within the Voice Platform Solution (VPS) depend on, and in turn dictate, the way in which the voice application is written.

Application Design Paradigms

There are three approaches to designing voice applications for use with GVP:

- IVR-centric—GVP (the IVR) has full control of the call, and maintains the VoiceXML session until the self-service call ends, or the assisted-service call is ready to be transferred to an available agent. If full computer-telephony integration (CTI) is required, the voice application itself controls CTI actions through CTI Connector (CTIC) integration with an IVR Server. Mid-call CTI actions include route requests, getting statistics, and sending and receiving attached data.

  Most legacy GVP applications typify the IVR-centric design approach. GVP 8.1 supports these types of voice applications through both the GVPi and the NGI. The GVPi also supports legacy applications that use TXML call control extensions. The NGI directly supports some TXML

equivalents, and works with the Call Control Platform to provide additional call control functionality through CCXML applications.

The other chapters in this guide describe in detail how the GVPi and NGI, respectively, support IVR-centric applications.

- Standard VoiceXML—The voice application provides voice self-service by using standard VoiceXML tags, with certain GVP extensions. The voice application can use the `<transfer>` tag to initiate a transfer to an agent for assisted service, and control of the call passes to the SIP Server or CTIC. While the call is waiting for an agent, the SIP Server or CTIC can apply call treatments by sending new `INVITE` requests to the Media Control Platform, which performs the actual call treatments. Alternatively, the voice application can use the GVP `<send>` and `<receive>` extensions to request other CTI functionalities through the message tunnel through the CTIC. In this case, the original VoiceXML session remains in progress.

  GVP 8.1 supports these types of voice applications through the NGI, and through the CCXMLI for call control applications.
- URS-centric—The inbound call arrives first at the Universal Routing Server (URS) routing strategy, and the strategy retains full control of the call. The strategy can invoke VoiceXML applications on GVP, but it uses GVP primarily as a media server to play prompts and collect user input. After the voice interaction, call control returns to the routing strategy. The voice application will never initiate transfers, and therefore it will never use the `<transfer>` tag.

  Genesys encourages this type of design to maximize the benefits of the full VPS.

GVP 8.1 supports the IVR-centric approach for customers who require the use of a single VoiceXML session through call queuing and call treatments, and to enable customers to migrate legacy applications for use with GVP 8.1. However, in general, Genesys recommends the standard VoiceXML and URS-centric design approaches for voice application development.

CTI and Non-CTI Voice Applications

The CTIC, which is a new component in GVP 8.1, integrates GVP with IVR Server to support full CTI functionality for IVR-centric voice applications (including legacy GVPi applications and NGI applications with CTI extensions), and for certain switch configurations whose architecture requires CTI through IVR Server.

Note: For GVP 8.1, the CTIC is available only for Windows.

For convenience, voice applications that use the CTIC for CTI functionality are referred to as *CTI applications.* Calls that do not go through the CTIC are

referred to as *non-CTI applications*. However, a limited amount of CTI functionality is available to non-CTI applications through SIP Server.

Configurable policy parameters enable you to specify whether the CTIC will be involved in a call. You can configure the policy at the level of the tenant or the IVR Profile.

For mid-call CTI interactions, the CTIC enables CTI messages to be tunneled between the voice application and the IVR Server. The GVPi uses SIP `INFO` messages for interactions such as getting and sending data, and sending route requests and treatment results. For NGI applications, attached data can be sent in SIP `INVITE` and `BYE` messages, as well as in SIP `INFO` messages. NGI applications can also send interaction data with transfer requests, as headers in SIP `INVITE` or `REFER` messages.

The primary purpose of the CTIC is to provide backward compatibility for legacy voice applications that will use the GVPi with GVP 8.1. However, all VoiceXML applications in GVP 8.1, including NGI applications, must use the CTIC and IVR Server for the following functionality:

- Obtaining `GetStat` and `PeekStat` statistics
- Using `AccessNumGet` (unless you are using URS)
- Depending on how the VoiceXML application has been written, retrieving mid-call data from Genesys Framework

To identify CTI-related SIP messages for processing, the CTIC and GVPi use the `content-type` header, which can be passed to and from the VoiceXML application as `contenttype` shadow variables and tag attributes. The expected content-type value is:

```
application/x-www-form-urlencoded;charset=utf-8
```

Genesys recommends that you use Composer Voice (release 8.0.1 or later) to develop CTI applications for the NGI, so that you can easily take advantage of the full range of CTI functionality available in GVP 8.1. For more information about using Composer, see the *Composer 8.0.1 Help*.

Feature Support

With limited exceptions that are described in the remainder of this guide, GVP 8.1 supports all the features that were available for GVP 7.6 voice applications. This section notes aspects of feature support that are relevant for application developers who are migrating legacy voice applications.

This section provides information about the following topics:

Transfers and Route Requests

The VoiceXML <transfer> tag transfers the caller to another destination, such as an agent. Table 1 summarizes the transfer types and telephony-layer transfer methods that GVP 8.1 supports through the Media Control Platform and the CTIC. For more information about transfers in GVP, see the chapter in the *Genesys Voice Platform 8.1 Deployment Guide* about how GVP works.

Table 1: Supported Transfer Types and Methods

Transfer Type	Supported SIP Transfer Methods		
	NGI	GVPi	With CTIC
Blind—The voice application is detached from the call as soon as the transfer is successfully initiated.	· HKF · REFER · BRIDGE • REFERJOIN (REFER with Replaces) · MEDIAREDIRECT · ATTCOURTESY · ATTCONSULT · ATTCONFERENCE · ATTOOBCOURTESY · ATTOOBCONSULT · ATTOOBCONFERENCE Default: REFER	· REFER	REFER For CTI-based transfer, the Media Control Platform adds an identifying Request-URI parameter (TransferOnCTI=Yes), and the CTIC sends a OneStepXFer message to the IVR Server. TransferOnCTI is an IVR Profile policy parameter. **Note:** GVP supports TransferOnCTI only when the IVR Server is configured in Behind mode.
Consultation, or supervised—The voice application is detached from the call when the transfer process is successfully completed.	· HKF · REFER · BRIDGE • REFERJOIN (REFER with Replaces) · MEDIAREDIRECT · ATTCONSULT · ATTCONFERENCE · ATTOOBCONSULT · ATTOOBCONFERENCE Default: REFERJOIN	REFER with Replaces	Not supported.
Bridge—The voice application is not detached from the call.	· BRIDGE · MEDIAREDIRECT Default: BRIDGE	· BRIDGE · MEDIAREDIRECT	• BRIDGE (INVITE) · MEDIAREDIRECT

Table 1: Supported Transfer Types and Methods (Continued)

Transfer Type	Supported SIP Transfer Methods		
	NGI	GVPi	With CTIC
Explanation of SIP transfer methods: • HKF (Hookflash)—One-leg transfer, using DTMF digits (RFC 4733). • REFER—One-leg transfer, using SIP REFER message (RFC 3515). • BRIDGE—Two-leg, or join-style, transfer, using SIP INVITE message. • REFERJOIN (REFER with Replaces)—Two-leg transfer, using SIP INVITE message to the called party, and SIP REFER with Replaces header to the original caller (RFC 3891). • MEDIAREDIRECT—Two-leg transfer, using SIP INVITE and re-INVITE messages. • ATTCOURTESY—One-leg transfer. AT&T Transfer Connect courtesy inband transfer. • ATTCONSULT—One-leg transfer. AT&T Transfer Connect consultation inband transfer. • ATTCONFERENCE—One-leg transfer. AT&T Transfer Connect conference inband transfer. • ATTOOBCOURTESY—One-leg transfer. AT&T Transfer Connect courtesy out-of-band transfer. • ATTOOBCONSULT—One-leg transfer. AT&T Transfer Connect consultation out-of-band transfer. • ATTCONFERENCE—One-leg transfer. AT&T Transfer Connect conference out-of-band transfer.			

Route Requests

Before a blind or bridge transfer is initiated from CTI applications, a route request can be made through the CTIC to the IVR Server. The RouteRequest invokes URS-controlled services such as queuing the call or performing treatments, or other CTI services such as sending attached data to the agent desktop.

- For GVPi voice applications, the <QUEUE_CALL> tag translates to a SIP INFO message from the Media Control Platform to the CTIC (through the Resource Manager), with the header, content-type=application/x-www-form-urlencoded;charset=utf-8.
- The NGI uses SIP REFER or SIP INVITE messages for route requests, depending on the type of transfer required. The CTIC uses an identifying Request-URI parameter (RouteRequest=1) to identify SIP REFER or SIP INVITE messages that are routing requests or requests for transfers with CTI services, rather than straightforward requests for transfers through the platform. To enable the CTIC to intercept routing requests that it must forward to the IVR Server, ensure that the voice application specifies a RouteRequest=1 attribute.

Changes in Feature Implementation

The implementation of the following features has changed from the way they were implemented in GVP 7.6:

- Use of the Semantic Interpretation for Speech Recognition (SISR) `out` rule—Unlike in GVP 7.6, the contents of the `out` variable are mapped to VoiceXML forms as described in the VoiceXML 2.0 specification. For more information:
  - For the GVPi, see "SISR out Rule" on page 36.
  - For the NGI, see "SISR out Rule" on page 61.
- `<foreach>` array usage—Unlike in GVP 7.6, the interpreters now throw a semantic error (`error.semantic`) when the array of `<foreach>` is undefined. For information about a workaround for the GVPi, see "Undefined <foreach> Array" on page 38.
- Built-in time and date grammars—For both the GVPi and the NGI, the built-in time and date grammars now conform strictly to the VoiceXML specification for dual-tone multifrequency (DTMF) input. For more information:
  - For the GVPi, see "Built-in Time and Date Grammars" on page 38.
  - For the NGI, see "Built-in Time and Date Grammars" on page 62.
- Repeat rules in DTMF grammars—The GVPi and NGI differ in the way that they treat DTMF grammars that use *repeat* in rules. The GVPi retains the legacy GVP 7.6 behavior. For more information for the NGI, see "Repeat Rules in DTMF Grammars" on page 62.
- Offboard DTMF grammar recognition—The GVPi supports off-board DTMF grammar recognition through a third-party Automatic Speech Recognition (ASR) engine, but the NGI does not support it in releases of GVPi up to and including 8.1.0. From version 8.1.1 onwards, NGI does support offboard DTMF grammar recognition. For more information for the NGI, see "Offboard DTMF Recognition" on page 64.
- Embedded references to other grammars—The GVPi does not support embedded references to external DTMF grammars when ASR is not used. In releases up to and including 8.1.0, the NGI does not support this feature for embedded references to all DTMF grammars. In release 8.1.1, both NGI and GVPi support it (but not SISR). For more information:
  - For the GVPi, see the limitation and workaround for DTMF grammars referencing other external grammars, on page page 35.
  - For the NGI, see "Embedded Grammars" on page 64.

MIBs and Traps

For Simple Network Management Protocol (SNMP) monitoring, GVP 8.1 uses the GVP 8.1 Management Information Bases (MIBs). GVP 7.6 MIBs are not supported.

GVP 8.1 provides equivalent GVPi traps for all legacy GVP 7.6 traps.

For information about how NGI event handling maps to legacy GVP 7.6 traps, see "MIBs and Traps" on page 65.

GVP Reporting

Compared with GVP 7.6, the GVP 8.1 Reporting Server enables equivalent logging and enhanced reporting features. On the various `Monitoring > Voice Platform` tabs in the Genesys Administrator, you can view reports on voice application usage and operational aspects of the deployment. In GVP 7.6, you used to have to obtain this information through a combination of the GVP 7.6 Voice Application Reporter (VAR), Genesys Reporting (CC Analyzer or CCPulse+), and Genesys Info Mart.

GVP Reporting services are based on the following data:

- Logs—Logging events that are raised at the level of the component, or GVP `Application` object. The log files that are generated are similar to the files that were generated for GVP 7.6 except that, with fewer components, there are fewer of them.

  Note: The format for the MRCP log element has changed. The Connection ID, which is exposed to the voice application as a session variable, used to be passed to the MRCP server in the following format: `GenesysLab_ResellerName_CustomerName_ApplicationName_ConnectionID_SessionID`. The format used in GVP 8.x is: `<SessionID>_<ConnectionID>_<TenantID>`.

- Metrics—Logging events that are raised at the level of the VoiceXML or CCXML application. For example, the Media Control Platform logs an `INCALL_BEGIN` metric when an inbound call is accepted.

  The NGI also supports VAR metrics, which are events that are generated by the Media Control Platform when the NGI executes a VAR-specific `<log>` tag in a voice application. VAR-specific `<log>` tags have the prefix `com.genesyslab.var`.

  For full details about the metrics that are available in GVP 8.1, see the *Genesys Voice Platform 8.1 Metrics Reference*. For more information about VAR metrics, see the Media Control Platform reference information appendix in the *Genesys Voice Platform 8.1 User's Guide*. For more information about using `<log>` elements in VoiceXML applications, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.
- Call detail records (CDRs)—Records that describe key attributes of call sessions that are processed by the deployment. For example, all CDRs include attributes that specify the session start and end time, and the type of call. CDRs are submitted by the Resource Manager, Media Control Platform, and Call Control Platform.

- Operational Reporting (OR) data—Statistics about call arrivals and call peaks that are submitted by the Resource Manager, Media Control Platform, and Call Control Platform.

General Limitations

GVP 8.1.0 and GVP 8.1.1

The following features are not supported in GVP 8.1.0 and 8.1.1:

- Multi-tenancy
- Dialogic Host Media Processing (HMP)
- ASR Log Manager system
- Support for Solaris

GVP 8.1.2

The following features are not supported in GVP 8.1.2:

- Dialogic Host Media Processing (HMP)
- ASR Log Manager system
- Support for Solaris

Genesys Administrator

The Genesys Administrator is a GUI that provides a web-based interface to the Genesys Configuration and Management Layers.

Use the Genesys Administrator to provision the IVR Profiles and to deploy, configure, and manage all GVP components.

To access the Genesys Administrator for your Genesys deployment, go to the following URL:

`http://<Genesys Administrator host>/wcm`

Figure 1 shows a typical Genesys Administrator page.

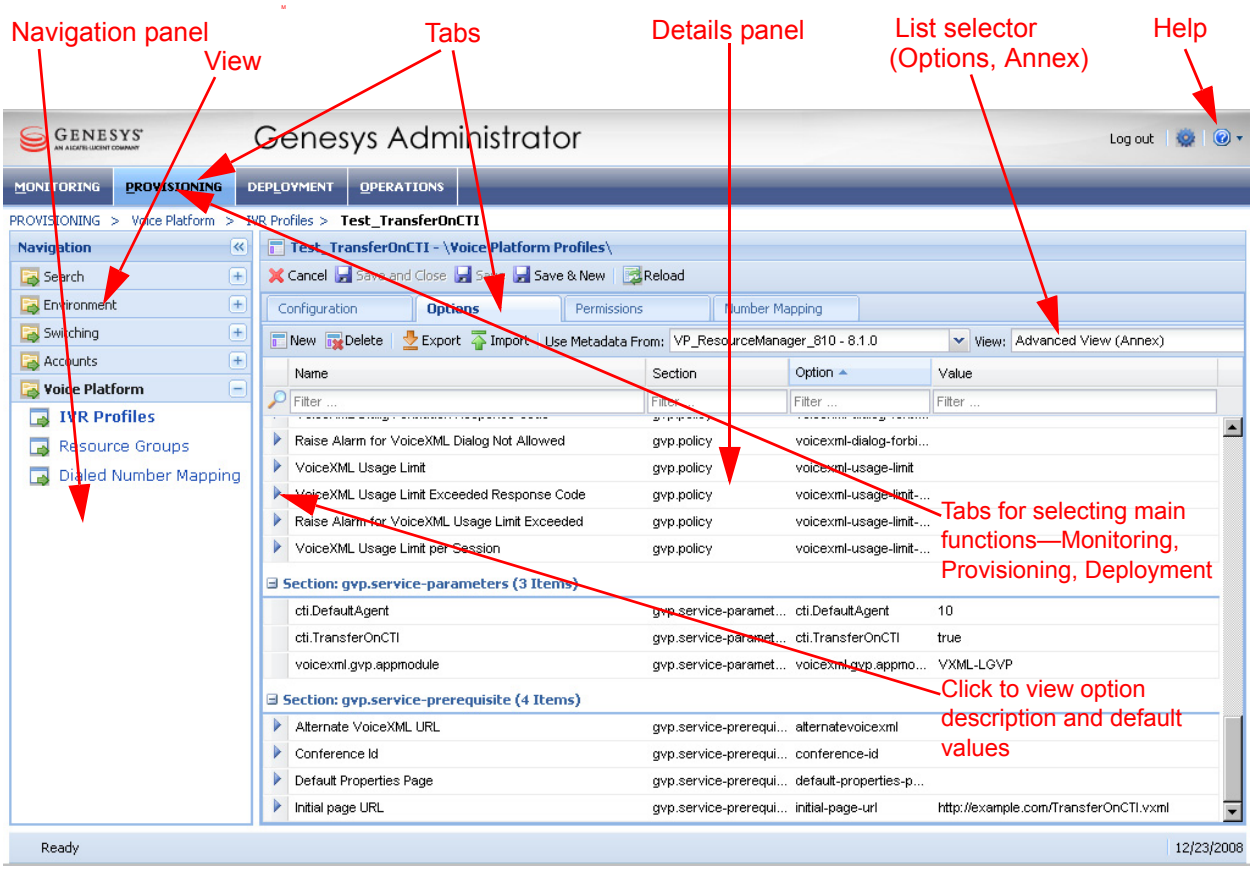

Figure 1: Genesys Administrator

More Information

- For general information about using the Genesys Administrator, see the *Framework 8.0 Genesys Administrator Help.*
- For information about using the Genesys Administrator to configure and provision GVP `Application` objects and IVR Profiles, see the *Genesys Voice Platform 8.1 Deployment Guide* or the *Genesys Voice Platform 8.1 User's Guide*.

Chapter

2 Migrating to GVP 8.1 GVPi

This chapter describes the platform, provisioning, and minor voice application modifications that are required to use legacy voice applications that were written for Genesys Voice Platform (GVP) 7.6, with the GVP 8.1 Legacy GVP Interpreter (GVPi).

This chapter contains the following sections:

For information about the VoiceXML concepts and features that GVPi supports, see the *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual*.

Task Summary

Table 2 summarizes the steps to migrate a legacy GVP voice application for the GVP 8.1 GVPi.

Table 2: Task Summary—Migrating to GVP 8.1 GVPi

Objective	Related Procedures and Actions
Provision a GVP 8.1 IVR Profile for each voice application that you want to migrate.	1. Use the IVR Profile Wizard in the Genesys Administrator to create an IVR Profile of service type `voicexml`, with the VoiceXML Interpreter property set to `Legacy GVP`. 2. Transfer the values of the configuration parameters listed in Table 3 on page 31 from the legacy voice application file (`AppID.xml`, the interpreted `.vxml` file) to the new IVR Profile. 3. If the IVR Profile that was provisioned in the GVP 7.6 Element Management Provisioning System (EMPS) included parameters that had been added to the `Extensions` section, transfer the `Extensions` parameters to the `gvp.service-parameters` section of the new IVR Profile. For more information, see Procedure: Provisioning the GVPi voice application, on page 27.
Configure the `PopGateway1`, `PageCollector` and `GarbageCollector` functions in the Media Control Platform process.	Transfer the values of the required `PopGateway1`, `PageCollector` and `GarbageCollector` configuration parameters from the legacy GVP EMPS to the Media Control Platform `Application` object. For full details, consult Genesys Professional Services.
If appropriate for your deployment, make the GVPi the default interpreter.	On the Media Control Platform `Application` object, in the `sessmgr` section, set the value of the `default_vxml_interpreter` option to `VXML-LGVP`. The default is `VXML-NG` (the Next Generation Interpreter [NGI]).
If your deployment requires the use of computer-telephony integration (CTI) through the CTI Connector (CTIC), configure CTI-related parameters on the IVR Profile and other components.	For full details, consult Genesys Professional Services.
Modify the VoiceXML application as required.	Review the information in "Feature Limitations, Differences, and Workarounds" on page 35, and then edit the VoiceXML file to accommodate any required adjustments.

Provisioning the IVR Profile

The following procedure describes the steps to create and configure an IVR Profile for GVPi.

Procedure:
Provisioning the GVPi voice application

Purpose: To create the GVP 8.1 IVR Profile, which provisions the legacy voice application to the Resource Manager.

You must create a separate IVR Profile for each voice or call control application that you want to be available to GVP.

Prerequisites

- The Resource Manager `Application` object has been installed, as described in the *Genesys Voice Platform 8.1 Deployment Guide*.
- The actual legacy VoiceXML file is available, and you have the required permissions to access it.
- You are logged in to the Genesys Administrator. To access the Genesys Administrator, go to the following URL:

  `http://<Genesys Administrator host>/wcm`

  For more information about using the Genesys Administrator to configure GVP, see the procedure in the *Genesys Voice Platform 8.1 User's Guide* about viewing or modifying GVP configuration parameters.

Start of procedure

1. Use the `IVR Profile Wizard` in the Genesys Administrator to create the GVP 8.1 `IVR Profile` object.
   a. In the Genesys Administrator, go to the `Provisioning > Voice Platform > IVR Profile` tab, and then launch the wizard from the `Tasks` pane.
   b. When prompted, specify the required configuration values. For a legacy GVPi IVR Profile, ensure that you specify the following:
      - On the `Service Type` page, from the `Service Type` drop-down list, select `VoiceXML`. (This sets the value of the `service-type` parameter, in the `gvp.general` configuration section, to `voicexml`.)
      - On the `Service Properties` page, from the `VoiceXML Interpreter` drop-down list, select `Legacy GVP`. (This sets the value of the `voicexml.gvp.appmodule` parameter, in the `service-parameters` configuration section, to `VXML-LGVP`.)

        If applicable, also specify the `Toll Free Number`, which identifies the phone number that the caller dialed for this call. (This sets the value of the `toll-free-number` parameter, in the `gvp.general` configuration section.)

c. Complete the remainder of the wizard pages as appropriate for your deployment. In particular, consider the following IVR Profile policy and service-parameter options:

- Whether REFER transfers are enabled—Specified on the IVR Capabilities page, through the Allow Transfers check box.

  This sets the value of the transfer-allowed parameter (for blind or consultation transfers), in the gvp.policy configuration section. By default, REFER transfers are enabled.
- Whether INVITE transfers are enabled—Specified on the IVR Capabilities page, through the Allow Outbound Calls check box.

  This sets the value of the outbound-call-allowed parameter (for bridge or consultation transfers, as well as for outbound calls), in the gvp.policy configuration section. By default, INVITE transfers are enabled.

Note: To enable consultation transfers, you must enable both REFER and INVITE transfers.

- Whether the voice application requires the use of the CTIC to interface with an IVR Server for CTI—Specified on the CTI Parameters page, through the Require CTI Interaction check box.

  This option sets the value of the cti-allowed parameter, in the gvp.policy configuration section. By default, the CTIC is not required.

  For interactions that use the CTIC and IVR Server, you can also specify whether blind transfers will be handled through the IVR Server interface (Transfer on CTI) and, if applicable, the default agent to whom transfers will fall back if the original transfer fails. (These options set the values of the cti.transferoncti and cti.defaultAgent, respectively, in the gvp.service-parameters configuration section.)

  For more information about voice applications and CTI, see “CTI and Non-CTI Voice Applications” on page 16.

  For more information about transfers, see “Transfers and Route Requests” on page 18.

Note: The cti-allowed flag, in the gvp.policy section, is necessary, but not sufficient, to enable CTI applications. For more information about configuring the GVP deployment to use the CTIC and IVR Server, see the *Voice Platform Solution 8.1 Integration Guide*.

For more information about completing the IVR Profile Wizard, see the procedure in the *Genesys Voice Platform 8.1 Deployment Guide* to create IVR Profiles using the wizard.

For more information about all the IVR Profile configuration parameters, see the chapter in the *Genesys Voice Platform 8.1 User's Guide* about configuring IVR Profiles.

d. When you have completed the required configuration in the wizard, click Finish.

The IVR Profile object is created in the Genesys Configuration Layer. The new IVR Profile appears in the list on the Provisioning > Voice Platform > IVR Profile tab in the Genesys Administrator.

2. Modify the IVR Profile to capture configuration parameters that were previously defined in the AppID.xml file or in the Extensions section of the provisioned IVR Profile in the GVP 7.6 EMPS:

a. Go to the Provisioning > Voice Platform > IVR Profile > <New IVR Profile> > Options tab.

Note: Option values do not display in the Value column on the IVR Profile Options tab unless you have overtly specified a value, either during the IVR Profile Wizard or else by manual configuration.

For information about the default setting in effect for a particular option that does not display a visible value, click the blue arrow to the left of the option name. Alternatively, see the *Genesys Voice Platform 8.1 Configuration Options Reference.*

---

b. Modify the values of the options in the gvp.service-parameters section, transferring the values of certain parameters that were previously specified in the GVP 7.6 AppID.xml file. For details about which parameters you must specify, and how the GVP 7.6 AppID.xml parameters map to GVP 8.1 IVR Profile parameters, see "Mapping IVR Profile Parameters" on page 30.

The syntax for parameters in the gvp.service-parameters section is *ValueType,Value*. For all the parameter values that you transfer (voicexml.gvpi.* in the gvp.service-parameters section), you must specify the *ValueType* as fixed (in other words, the configured value will be in the SIP Request-URI).

Example

- AppID.xml parameter:
  <SET VARNAME="$default-language$" VALUE="en-US" />
- New IVR Profile:
  Section = gvp.service-parameters
  Option name = voicexml.gvpi.$default-language$
  Option value = fixed,en-US

c. If the IVR Profile that was provisioned in the GVP 7.6 EMPS included parameters that had been added to the `Extensions` section, transfer the names and values of the `Extensions` section parameters to new `voicexml.gvpi` configuration options that you create in the `gvp.service-parameters` section of the new IVR Profile.

   In all cases, the syntax that you must use when you specify the option value is *`ValueType,Value,`* and you must specify the *`ValueType`* as `fixed`.

   Example

   - EMPS IVR Profile `Extensions` parameter:
     `name=$my-var$, value="abcd"`
   - New IVR Profile option:
     Section = `gvp.service-parameters`
     Option name = `voicexml.gvpi.$my-var$`
     Option value = `fixed,abcd`

3. If the voice application uses the CTIC (see Step 1c), you may need to configure additional IVR Profile parameters to transfer values that used to be set for the IVR Server Client in the EMPS. For more information, consult Genesys Professional Services.
4. Save the IVR Profile. Click one of the following:
   - `Save`—To save the changes and continue configuring the IVR Profile.
   - `Save and Close`—To save the changes and return to the list of IVR Profiles.
   - `Save and New`—To save the changes and create another IVR Profile with the identical configuration, as a shortcut for configuring multiple IVR Profiles.

End of procedure

Mapping IVR Profile Parameters

Table 3 shows how GVP 7.6 `AppID.xml` parameters map to GVP 8.1 IVR Profile parameters. You must transfer the values that were specified as `VARNAME` parameters in the `AppID.xml` file to the GVP 8.1 IVR Profile configuration.

GVP 8.1 provides unchanged support for all other GVP 7.6 `AppID.xml` parameters that are not listed in Table 3 or explicitly identified as feature limitations (see Table 4 on page 35). For full details about supported VoiceXML features, see the *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual*.

Notes: The values shown in Table 3 are typical sample values. For seamless migration from your legacy applications, ensure that you set the IVR Profile parameters to the actual values in the legacy `AppID.xml` file.

Table 3: Mapping Legacy AppID.xml File Parameters to the IVR Profile

AppID.xml Parameter	Equivalent IVR Profile Parameter	
	Option Name	**Default Value**
	Section: gvp.service-prerequisite	

`<SET VARNAME="$ivr-root-dir$" VALUE="http://bpn-cmtyvrws-01:9080/flcarecenter76" />`
 	
`<SET VARNAME="$ivr-url$" VALUE="$ivr-root-dir$/Welcome.jsp" />`
 	
`<SET VARNAME="$start-ivr-url$" VALUE="$ivr-url$?NEXTACTION=START&amp;ANI=$ani$&amp;DID=$toll-free-num$&amp;SESSIONID=$sessionid$" />`
	`initial-page-url`
	Note: Recursively expand the value of `$start-ivr-url$`, replacing `$ivr-url$` with its value and then replacing `$ivr-root-dir$` with its value, so that the three `AppID.xml` parameters result in a single string that sets the value of the initial page URL for the IVR Profile.
		No default value

`<SET VARNAME="$ivr2-root-dir$" VALUE="http://bpn-rosevrws-01:9080/flcarecenter76" />`
 	
`<SET VARNAME="$ivr2-url$" VALUE="$ivr2-root-dir$/Welcome.jsp" />`
 	
`<SET VARNAME="$start-ivr-url$" VALUE="$ivr-url$?NEXTACTION=START&amp;ANI=$ani$&amp;DID=$toll-free-num$&amp;SESSIONID=$sessionid$" />`
	`alternatevoicexml`
	Note: Recursively expand the value of `$start-ivr-url$`, replacing `$ivr2-url$` with its value and then replacing `$ivr2-root-dir$` with its value, so that the three `AppID.xml` parameters result in a single string that sets the value of the alternate page URL for the IVR Profile.
		No default value

	Section: gvp.general	
`<SET VARNAME="$toll-free-num$" VALUE="18002234567" />`	`toll-free-number`	No default value
	Section: gvp.service-parameters	
	Note: The syntax to specify `gvp.service-parameters` options is: *`ValueType,Value`*. To migrate legacy applications unchanged, specify the *`ValueType`* as `fixed` (the parameter will be in the SIP `Request-URI`). For more information about the valid `ValueTypes`, see the chapter in the *Genesys Voice Platform 8.1 User's Guide* about configuring IVR Profiles.	
`<SET VARNAME="$default-language$" VALUE="en-US" />`	`voicexml.gvpi.$default-language$`	`fixed,en-US`

Table 3: Mapping Legacy AppID.xml File Parameters to the IVR Profile (Continued)

AppID.xml Parameter	Equivalent IVR Profile Parameter	
	Option Name	**Default Value**
`<SET VARNAME="$ivr-tmo$" VALUE="6" />`	`voicexml.gvpi.$ivr-tmo$`	`fixed,6`
`<SET VARNAME="$ccerror-telnum$" VALUE="9" />`	`voicexml.gvpi.$ccerror-telnum$`	No default value
`<SET VARNAME="$cpatimeout$" VALUE="0" />`	`voicexml.gvpi.$rexfertimeout$`	`fixed,0`
`<SET VARNAME="$outbound-call-limit$" VALUE="21600" />`	`voicexml.gvpi.$outbound-call-limit$`	`fixed,21600`
`<SET VARNAME="$tts-vendor$" VALUE="MRCP" />`	`voicexml.gvpi.$tts-vendor$`	No default value
`<SET VARNAME="$tts-gender$" VALUE="MALE" />`	`voicexml.gvpi.$tts-gender$`	`fixed,MALE`
`<SET VARNAME="$asrplatform$" VALUE="MRCP" />`	`voicexml.gvpi.$asrplatform$`	No default value
`<SET VARNAME="$asrwavfilelog$" VALUE="FALSE" />`	`voicexml.gvpi.$asrwavfilelog$`	fixed,`False` (record utterance is disabled)
`<SET VARNAME="$transfer-type$" VALUE="1-SignalChannel" />`	`voicexml.gvpi.$transfer-type$` Valid values are: • `1SignalChannel`—For blind transfers • `2SignalChannel`—For consultation and bridge transfers **Note:** The value of `voicexml.gvpi.$transfer-type$` and `voicexml.gvpi.$transfer-option$` must be consistent.	`2Signal Channel`

Table 3: Mapping Legacy AppID.xml File Parameters to the IVR Profile (Continued)

AppID.xml Parameter	Equivalent IVR Profile Parameter	
	Option Name	**Default Value**
`<SET VARNAME="$transfer-option$" VALUE="SIPRefer" />`	`voicexml.gvpi.$transfer-option$` Valid values are: • `SipRefer`—For blind transfer • `ReferWithReplace`—For consultation transfer • `ATTCourtesy`—For AT&T Courtesy Transfer • `ATTConsultative`—For AT&T Consult and Transfer • `ATTConference`—For AT&T Conference and Transfer • `ATTOOBCourtesy`—For AT&T Out-of-Band Courtesy Transfer • `ATTOOBConsult`—For AT&T Out-of-Band Consult and Transfer • `ATTOOBConference`—For AT&T Out-of-Band Conference and Transfer • Not set—For bridge transfer **Note:** GVP 8.1 does not support the use of any other types of SIP messages for blind or consultation transfers with the GVPi. For more information, see "Transfers and Route Requests" on page 18.	Not set
`<SET VARNAME="$adn-flag$" VALUE="0" />`	`voicexml.gvpi.$adn-flag$`	`False` (debugging is disabled)
`<SET VARNAME="$record-pages$" VALUE="" />`	`voicexml.gvpi.$record-pages$`	`False` (dumping of the fetched XML page to a temporary folder is disabled)
`<SET VARNAME="$trap-url$" VALUE="" />`	`voicexml.gvpi.$trap-url$`	No default value

Table 3: Mapping Legacy AppID.xml File Parameters to the IVR Profile (Continued)

AppID.xml Parameter	Equivalent IVR Profile Parameter	
	Option Name	Default Value
<SET VARNAME="$debug-url$" VALUE="" />	voicexml.gvpi.$debug-url$	No default value
<SET VARNAME="$call-trace-url$" VALUE="" />	voicexml.gvpi.$call-trace-url$	No default value
<SET VARNAME="$tntenable$" VALUE="0" />	voicexml.gvpi.$tntenable$	0
<SET VARNAME="$tntscript$" VALUE="" />	voicexml.gvpi.$tntscript$	No default value
<SET VARNAME="$tntreclaimcode$" VALUE="" />	voicexml.gvpi.$tntreclaimcode$	*7
<SET VARNAME="$badxmlpageposturl$" VALUE="" />	voicexml.gvpi.$badxmlpageposturl$	No default value
<SET VARNAME="$defaultroutenum$" VALUE="9002241234" />	cti.defaultAgent	No default value

Configuring GVP Processes

In GVP 8.1, the following components and configuration objects replace GVP 7.6 components to perform processes related to voice application usage:

- For voice applications that use the GVPi, the Media Control Platform performs the `PopGateway` and `Page Collector` functions that the IP Communication Server (IPCS) performed in GVP 7.6 deployments.
- In GVP 8.1.2, the PSTN Connector corresponds to core functionality that the Voice Communication Server (VCS) performed in GVP 7.6 deployments.
- The CTIC interfaces with the IVR Server to perform the functions that the IVR Server Client (formerly called Genesys Queue Adapter [GQA]) performed in GVP 7.6 deployments.
- The CTIC and GVPi, between them, perform the functions that by the IPCS Call Flow Assistant (CFA) performed in GVP 7.6 deployments.

Communications between the CTIC and the IVR Server uses Transmission Control Protocol (TCP). This is unchanged from GVP 7.6.

Communication between the CTIC and the rest of GVP (specifically, the Media Control Platform and Resource Manager) uses SIP.

Before you can use migrated voice applications with the GVPi, you must configure certain equivalent server settings on GVP components such as the Media Control Platform and the CTIC. For full details, consult Genesys Professional Services.

Feature Limitations, Differences, and Workarounds

This section provides information about changes in feature implementation between GVP 7.6 and GVP 8.1. It contains information about the following topics:

Feature Limitations

Table 4 lists the features that were available in GVP 7.6, but that the GVP 8.1 GVPi does not support. The table includes workarounds or voice application modifications where possible.

Table 4: Feature Limitations in GVP 8.1 GVPi

GVP 7.6 Feature	Comments and Workarounds for GVP 8.1
Asynchronous posting of transactional recordings (`async` mode). This feature uses Bandwidth Manager (BWM).	Because BWM is not available in GVP 8.1, `async` mode is not supported. If the voice application specifies `async` mode, it will be silently ignored, and the platform will post in `sync` mode instead.
Asynchronous fetching of utterance recordings. `com.genesys.utterancefetchmode="async"`	GVP 8.1 supports only synchronous fetching of utterance recordings. If the voice application specifies `async` mode, it will be silently ignored, and the platform will fetch in `sync` mode instead.
ASR Log Manager system.	The GVP 8.1 architecture does not include the ASR Log Manager and ASR Log Server.
Post-connect Call Progress Analysis (CPA). For example, the VoiceXML application uses the `AFTERCONNECTTIMEOUT` attribute with the `<CREATE_LEG_AND_DIAL>` or `<transfer>` elements to detect a fax or answering machine.	GVP 8.1.2 supports post-connect CPA. For details, see the *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual.*

Table 4: Feature Limitations in GVP 8.1 GVPi (Continued)

GVP 7.6 Feature	Comments and Workarounds for GVP 8.1
Telera XML (TXML) <REXFER> tag.	GVP 7.6 used this tag to support a feature that is specific to Cisco ICM. Because GVP 8.1 and 8.1.1 do not support Cisco ICM, the <REXFER> tag is no longer applicable.
Various $-variables: • $bwmurl$ • $dialreturncode$ • $did-root-dir$ • $telephony-port$ • $tntbusydtmf$ • $tnterrdtmf_ • $tnthangupdtmf$ • $tntsuccessdtmf$ • $tnttriggerCode$ • $_toneinput$ • $transferscript-connecttimeout$ • $transferscript-number$ • $transferscript-result$ • $tts-cache$ • $tts-current-root-dir$ • $tts-enabled$ • $tts-tmo$ • $tts-url$ • $tts-vendor-specific-param$ • $tts-voice-name$ • $voicefile-format$	Not supported. (They are no longer applicable because the underlying architecture has changed.)

SISR out Rule

The platform supports the out rule as defined in the Semantic Interpretation for Speech Recognition (SISR) specification. The contents of the out variable are mapped to VoiceXML forms as described in the VoiceXML 2.0 specification.

The GVP 7.6 interpreter exposed the out value as a property of the result variable, and the VoiceXML application accessed the value through the fieldname.out variable.

By contrast, the GVP 8.1 GVPi evaluates the contents of the out variable, and it assigns the correct slot value to the field.

Workaround

To enable you to reuse legacy voice applications that rely on the GVP 7.6 behavior, GVP 8.1 provides a configurable option, `SupportSisrOutElement`, in the `Popgateway1` configuration section on the Media Control Platform. For unchanged reuse of legacy voice applications that exposed the `out` result as a property, set the value of this option to `False`. The default value is `true`.

Example

The following is an example of a legacy voice application with a field-level grammar that can have multiple slots, where one slot can match several fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns:telera="http://www.telera.com/vxml/2.0/ext/20020430" xmlns="http://www.w3.org/2001/vxml"
   version="2.0">
<property name="timeout" value="10s"/>
<property name="bargein" value="true"/>
<form>
   <block>
      testing form-level grammar with multiple slots. one slot matches several fields. Press 1.
   </block>
   <field name="Testfield" slot="X">
      <grammar mode="dtmf" type="application/srgs+xml" version="1.0" root="ROOT1"
    xmlns="http://www.w3.org/2001/06/grammar">
         <rule id="ROOT1" scope="public">
            <one-of>
               <item> 1  <tag>out = new Object();out.X='valueX'; out.Y='valueY'; out.Z='valueZ';
    </tag></item>
            </one-of>
         </rule>
      </grammar>
   </field>
   <filled mode="all">
      <if cond=" Testfield.out.X == 'valueX'">pass<disconnect/>
      <else/>fail<disconnect/>
      </if>
   </filled>
</form>
</vxml>
```

In GVP 7.6, when DTMF input of `1` was received, the interpreter exposed the `out` property in `Testfield`, and the voice application could access the semantic interpretation by using `Testfield.out.X`, `Testfield.out.Y`, or `Testfield.out.Z`.

In GVP 8.1, the interpreter evaluates the content of `out` and assigns the correct slot value to `Testfield`. Because `slot X` is specified, the value of `Testfield` will be `valueX`. With default settings for the GVPi, using `Testfield.out.X` in the legacy voice application will result in a semantic error. However, if you set

the `SupportSisrOutElement` in the `Popgateway1` section to `False`, you can reuse the legacy voice application unchanged in GVP 8.1 without generating the error.

Undefined <foreach> Array

Unlike in GVP 7.6, the GVPi now throws a semantic error (`error.semantic`) when the array of `<foreach>` is undefined.

Workaround

To ignore the error, so that you can reuse legacy voice applications that rely on the GVP 7.6 behavior, set the `ThrowErrorWhenForEachArrayUnDefined` option in the `Popgateway1` configuration section on the Media Control Platform to `False`.

Built-in Time and Date Grammars

The default built-in time and date grammars now conform correctly to the VoiceXML specification for DTMF input (see `http://www.w3.org/TR/voicexml20/#dmlABuiltins`).

- For time input, the built-in time grammar uses the correct suffix character. The format of recognition results is `hhmmx`, where `x` is:
  - `?`—for inputs 01:00–12:59
  - `h`—for inputs 13:00–23:59 and 00:00–00:59

  In GVP 7.6, the `x` field always had the question-mark value (`?`).
- For date input of 4 or 6 digits, the built-in date grammar places the filler character(s) in the correct position. The format of recognition results is `yyyymmdd`. The standard specifies that, if any of the `yyyy`, `mm`, or `dd` input is missing, the question-mark character (`?`) substitutes for the missing input in the field.

  In GVP 7.6, the question-mark character (`?`) was always inserted at the beginning of the result.

  For example, if the DTMF input is `200401`, the result in GVP 8.1 is `200401??`. In GVP 7.6, the result was `??200401`.

Workaround

To enable you to reuse legacy voice applications that rely on the GVP 7.6 behavior, GVP 8.1 provides alternative time and date grammar rules that will produce the same results as the GVP 7.6 interpreter. To use the alternative grammar rules, set the following options in the `Popgateway1` configuration section on the Media Control Platform:

- `Built-in Time Grammar Path`—Set this option to the following value: `http://$LocalIP$/vggrammarbase/dtmf/time.grxml#ROOT_GVPI_7X`

- `Built-in Date Grammar Path`—Set this option to the following value: `http://$LocalIP$/vggrammarbase/dtmf/date.grxml#ROOT_GVPI_7X`

Deprecated Functionality

This section describes additional changes in GVPi VoiceXML support since GVP 7.6. This information is provided for clarification only; no changes are required for existing GVP 7.6 voice applications.

Termination Characters

The `com.genesys.returntermchar` property has been deprecated. If this property has not been removed from the VoiceXML application, it will be ignored.

In applications in which the `com.genesys.returntermchar` property used to be set to `true`, the `termchar` property should be set to "" (empty). As a convenience, if the value of the `com.genesys.returntermchar` property has been left as `true`, the GVPi will automatically set the value of `termchar` to empty.

Chapter

3 Migrating to GVP 8.1 NGI

This chapter is intended for customers who want to migrate legacy Genesys Voice Platform (GVP) 7.6 voice applications to use the GVP 8.1 Next Generation Interpreter (NGI).

The chapter maps legacy VoiceXML features to GVP 8.1 NGI support, to provide detailed information about the changes that are required in the VoiceXML application itself. The chapter also provides information about legacy voice application features that were available in GVP 7.6 but that are not possible, or whose implementation has changed, with the GVP 8.1 NGI.

This chapter contains the following sections:

Using the migrated voice applications in GVP requires no special configuration and provisioning steps. You configure and provision the IVR Profiles, Media Control Platform, and other GVP components as described in the *Genesys Voice Platform 8.1 Deployment Guide* and the *Genesys Voice Platform 8.1 User's Guide*.

For more information about the VoiceXML features that are supported on the GVP 8.1 NGI, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

For information about differences between GVP 8.1 and GVP 7.6 support for certain features (for example, time and date built-in grammars), see "Feature Support" on page 17.

For more information about all the legacy VoiceXML features, some of which may not be available on the NGI, see the *Genesys Voice Platform 7.6 VoiceXML 2.1 Reference Manual*.

VoiceXML Extensions

The tables in this section describe NGI support for the following categories of GVP extensions to VoiceXML:

Element Extensions

Table 5 describes the equivalent NGI support for the functionality provided by GVP 7.6 element extensions.

Table 5: Mapping Element Extension Functionality

Functionality	Legacy GVP	NGI Equivalent

Namespaces	Telera namespace:
	`http://www.telera.com/vxml/2.0/ext/20020430`
	Genesys namespace:
	`http://www.genesyslab.com/vxml/2.0/ext/20020430`
	Example:
	`<vxml Version="2.0" xmlns="http://www.w3.org/2001/vxml" xmlns:genesys="http:www.genesyslab.com/vxml/2.0/ext/20020430" xmlns:telera="http://www.telera.com/vxml/2.0/ext/20020430"`
		GVP namespace:
		`http://www.genesyslab.com/2006/vxml21-extension`
		Example:
		`<vxml Version="2.1" xmlns="http://www.w3.org/2001/vxml" xmlns:gvp="http://www.genesyslab.com/2006/vxml21-extension" >`

Asynchronous posting of recorded files	An extension to the `<submit>` element, with the following two attributes added: · `mode="sync"|"async"` · `asyncposturl="`*`URL to which the namelist is sent in asynchronous mode`*`"`	Not supported.

Table 5: Mapping Element Extension Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent

Playing dynamic data	An extension to the `<audio>` element enabled prerecorded snippets of `.vox` files to play currency, dates, days, numbers, digits, letters, or ordinals. (Required use of the Telera namespace.)
	Example:
	`<script src='Languages/en-US/PlayBuiltinType.js'/>`
	`<telera:audio expr='PlayBuiltinType("37", "number");'/>`
		The NGI provides builtin prompts, but the NGI does not support the concept of system prompts. For playback of dynamic data, use the `<foreach>` feature of VoiceXML 2.1.

Playing DTMF tones	An extension to the `<value>` element. (Required use of the Telera namespace.)
	Example:
	`<telera:value mode="dtmfplay" expr='411#'/>`
		Built-in audio is provided to play DTMF tones individually.
		Examples:
		`<audio src="builtin:dtmf/dtmf_4.vox"/>.`
		`<audio src="builtin:dtmf/dtmf_1.vox"/>.`
		`<audio src="builtin:dtmf/dtmf_pound"/>.`

Call Progress Analysis (CPA)	An extension to the `<transfer>` element, with the `afterconnecttimeout` attribute added. (Required use of the Telera namespace.) This feature enabled the platform to determine the result of an outbound call, and to make the result accessible through the `$cparesult$` variable (see page 59). The native IP Communication Server (IPCS) supported only preconnect CPA (such as busy and ring-no-answer). Post-connect CPA (such as fax and answering machine) required the addition of Dialogic Host Media Processing (HMP).	GVP 8.1.1 and earlier does not support post-connect CPA for transfers. Starting in the GVP 8.1.2 release, for a VoiceXML <transfer>, NGI supports post-connect CPA. **Note:** Beep detection is currently not supported.

Table 5: Mapping Element Extension Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent
Extension to the <param> element for sending user data	The append attribute, which was added to the <param> element with the Genesys namespace, indicated whether the header should be appended to the existing header.	Use the <send> tag. This is a GVP extension that must be qualified by the GVP namespace. For more information, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.
Posting <log> element information	The posturl attribute in the <log> tag enabled the voice application to instruct the platform to post text within the <log> element to a specified URL.	For equivalent functionality, use one (or a combination) of the following methods: • Real Time Debugger • Maintainer e-mail • Submitting messages after the call has ended
Controlling clearance of the Key Ahead buffer	The clearbuffer attribute, an extension to the <field>, <record>, <transfer>, or <menu> input elements, controlled whether the voice application would clear any buffered DTMF input that was collected before execution of the input element began.	Use the cleardtmf extension to the <form> tag to control clearance of buffered DTMF input that was collected before entry into the form.

Session Variable Extensions

Table 6 describes the equivalent NGI support for the functionality provided by GVP 7.6 session variable extensions.

Table 6: Mapping Session Variable Functionality

Functionality	Legacy GVP	NGI Equivalent
Connection ID	In the behind-the-switch and in-front-of-the switch modes, GVP fetched the connection ID. In the Network mode, the connection ID was not available to GVP at call setup time. GVP exposed the connection ID to the voice application through a session variable, `session.genesys.connid`. The connection ID was passed to the MRCP server through the logging MRCP tag, in the following format: `GenesysLab_<ResellerName>_<CustomerName>_<ApplicationName>_<ConnectionID>_<SessionID>`.	The Universal Unique Identifier (UUID) that T-Server or SIP Server generates is made available as the `session.connection.uuid` session variable. The connection ID is passed to the MRCP server through the logging MRCP tag, in the following format: `<SessionID>_<ConnectionID>_<TenantID>`.
`session.genesys` object	All the Telera XML (TXML) $-variables could be accessed through the VoiceXML `session.genesys` object. For example, `$application-name$` mapped to `session.genesys.application_name`. **Note:** All hyphens (-) were converted to underscores (_).	For equivalent NGI support for $-variables, see "VoiceXML $-Variables" on page 56.

Full Call Recording Changes in 8.1.1

Support for relative path

In release 8.1.1, the NGI now supports only a relative path for the Full Call Recording (FCR) feature. The path, specified by `<local-directory-name>` is treated as a path relative to the full call recording root path.

Compliance with RFC 5552

VoiceXML session variables

In 8.1 and prior releases, a parameter in the SIP Request URI was exposed as a VoiceXML session variable as follows:

- Request URI: `sip:dialog@host;ParameterA=ValueA`
  Session Variable:
  `session.connection.protocol.sip.requesturi['ParameterA']`

Starting from 8.1.1, the parameter name is converted to lower case in the session variable's array index. This is a change from earlier releases, where the parameter name was passed without change from the client. This change in specification occurred as `ietf-draft-burke` moved to a standard protocol in the form of RFC 5552.

- Request URI: `sip:dialog@host;ParameterA=ValueA`
  Session Variable:
  `session.connection.protocol.sip.requesturi['parametera']`

> **Note:** In release 8.1.1, the index has to be lowercase. Migration to GVP 8.1.1 may require that you update such parameter names used in URS routing scripts, or in third-party SIP clients using the NETANN interface to GVP.

Property Extensions

Table 7 describes the equivalent NGI support for the functionality provided by GVP 7.6 property extensions.

Table 7: Mapping Property Extensions Functionality

Extension	Legacy GVP	NGI Equivalent

com.genesys.ttsfetchhint	Controlled the Text-to-Speech (TTS) fetch.
	For the GVP 7.6 IPCS, the only supported value was `safe`, because the IPCS provided MRCP direct audio streaming to the caller.
	Example:
	• To turn off prefetch (the default):
	`<property name="com.genesys.ttsfetchhint" value="safe"></property>`
	• To turn on prefetch:
	`<property name="com.genesys.ttsfetchhint" value="prefetch"></property>`
		No equivalent. Because the Media Control Platform supports MRCP direct audio streaming to the caller, this feature is not required.

Table 7: Mapping Property Extensions Functionality (Continued)

Extension	Legacy GVP	NGI Equivalent
com.telera. speechenabled	Specified whether an Automatic Speech Recognition (ASR) engine was used in the voice application. The value was either `true` or `false`. Setting the value to `true` also: • Enabled DTMF recognition on the MRCP engine. • Enabled you to effectively override the scoping rules of the standard `inputmodes` property—at each grammar scope, the legacy interpreter considered the `inputmodes` property of that particular grammar scope to determine which grammars to enable within that scope.	The specific property is not supported, but you can use the VoiceXML `inputmodes` property to achieve almost equivalent functionality by limiting the activation of voice grammars. Note the following additional requirements or limitations: • All DTMF grammars are handled natively and cannot be sent to an ASR engine. • Enabling different grammars for different scopes is not supported.
com.telera. audioformat	Specified the audio format for the audio file. The default was `audio/basic`.	Use the MIME type. The web server must correctly set and return the `Content-Type` header in the HTTP response for the audio files.
com.genesys. returntermchar	Specified whether, in addition to the recognized result, the termination character would be returned to the VoiceXML application. When set to `true`: • The VoiceXML `termchar` property also had to be set to the termination character(s) that were to be accepted and returned. • The grammar to be matched had to allow the termination characters as valid input. **Note:** This property could be used only in DTMF-only applications (in other words, in GVP configurations that did not use ASR).	You must set the `termchar` property in the application to "" (empty). The `com.genesys.returntermchar` property is not supported. If this property has not been removed from the VoiceXML application, it will be ignored.

Table 7: Mapping Property Extensions Functionality (Continued)

Extension	Legacy GVP	NGI Equivalent
com.telera.transferscripturl	Specified the transfer connect URL. The URL pointed to an application that the AT&T Transfer Connect feature used to communicate with the agent leg.	This feature is called Multiphase Transfer in GVP 8.x. For more information, see the section about Multiphase Transfers in the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.
com.genesys.utterancefetchmode	Specified whether the utterance would be fetched synchronously or asynchronously.	Not supported.

Error Extensions

Table 8 describes the equivalent NGI support for the functionality provided by GVP 7.6 error extensions.

Table 8: Mapping Error Extensions Functionality

Extension	Legacy GVP	NGI Equivalent
telera.error.name	Specified the name of the error from a list of generic errors.	Use the VoiceXML standard event name (`_event`) in catch handlers.
telera.error.description	Provided a detailed description of the error that occurred.	Use the VoiceXML standard description (`_message`) in catch handlers.
telera.error.currenturl	Specified the URL of the page on which the error occurred.	Supported through the `_url` available in the catch handler.
telera.error.element	Specified the identity of the element in which the error occurred. If this was not available, the type of element was specified instead. **Example:** Suppose that the error occurred in the `<prompt>` element, and the `<prompt>` element had no `id` attribute. In this scenario, the value of the `telera.error.element` was `<prompt>`.	Supported through the `_element` and `_line` available in the catch handler.

Event Extensions

Table 9 describes the equivalent NGI support for the functionality provided by GVP 7.6 event extensions.

Table 9: Mapping Event Extensions Functionality

Extension	Legacy GVP	NGI Equivalent
error.com.telera.createleg	Occurred for a failure condition in the `CREATE_LEG_AND_DIAL` call control element.	Errors related to `<transfer>` are indicated through the standard VoiceXML `error.connection` event.
error.com.telera.dial	Occurred when there was a dial error during the `CREATE_LEG_AND_DIAL` call control element.	
error.com.telera.bridge	Occurred for a failure to bridge after issuing the `<BRIDGE/>` call control element.	
error.com.telera.unbridge	Occurred when there was a failure to unbridge after issuing the `<UNBRIDGE/>` call control element.	
error.com.telera.queue	Occurred for all Universal Routing Server (URS) or Cisco ICM communication, and for interaction failures related to any one of the three `QUEUE` control elements.	The `<send>`, `<receive>`, `<transfer>`, and `<exit>` elements enable communication with external components. You can parse errors from the `<receive>` message or the `<transfer>` form item variable. **Notes:** • `<send>` and `<receive>` are GVP extensions. • Cisco ICM is not supported in GVP 8.1.
error.badfetch.grammar.load	Occurred when the ASR engine failed to load the grammar.	Supported as is.
error.badfetch.grammar.syntax	Occurred when the ASR engine failed to compile the grammar.	
error.badfetch.grammar.uri	Occurred when the ASR engine failed to fetch the grammar URI.	
error.badfetch.grammar.timeout	Occurred when a grammar recognition timed out.	
error.unsupported.object.transrec	Occurred if transaction recording was not supported.	Not applicable, because Full Call Recording (FCR) is supported in all scenarios.

Table 9: Mapping Event Extensions Functionality (Continued)

Extension	Legacy GVP	NGI Equivalent
error.semantic.posturl	Occurred when the `posturl` was not specified, and GVP did not know where to `POST` the recording.	Not applicable, because asynchronous posting is not supported.
error.semantic.transrec.oneallowed	Occurred when an attempt was made to start transaction recording more than once in a single call.	If FCR functionality is used incorrectly, an `error.semantic` is generated, and the `_message` variable is filled in accordingly.
error.semantic.transrec.notstarted	Occurred when an attempt was made to stop transaction recording when there was no command to start it.	
error.semantic.transrec.notenabled	Occurred when transaction recording was not enabled.	
error.semantic.transrec.notallowed.multiplelegs	Occurred when an attempt was made to start transaction recording on multiple legs of the same call.	Not applicable, because the NGI supports transaction recording on multiple legs of the same call.
error.semantic.ASRFreeResource.notapplicable	Occurred when an attempt was made to free the MRCP ASR resource (license) in an ASR-disabled application.	Not applicable.

Object Element Extensions

Table 10 describes the equivalent NGI support for the functionality provided by GVP 7.6 object element extensions.

Table 10: Mapping Object Element Extensions Functionality

Functionality	Legacy GVP	NGI Equivalent

Get data from Framework	Object element attribute:
	`classid="CRData:get"`
	Enabled the voice application to retrieve data from the IVR Server by having a VoiceXML object element with `classid="CRData:get"` in the form. The `param` element specified the keys, and passed the `param` names to the server as a namelist (`key1 key2...`).
	(Applicable only for interactions between IVR Server and URS.)
		The proprietary `<send>` and `<receive>` extensions enable the voice application to request data from a routing strategy. The data key names are passed to URS, and the data values similarly passed back from URS, as a namelist in the body of a SIP `INFO` message.
		Note: `<send>` and `<receive>` are GVP extensions that must be qualified by the GVP namespace.
		See also Perform an action on Framework, on page 52.
		Example:
		Suppose that the following properties are specified in the form:
		`<property name="com.genesyslab.externalevents.enable" value="false"/>`
		`<property name="com.genesyslab.externalevents.queue" value="true"/>`
		and suppose that a script block that defines two variables, `KeyName1 = "GRAMMARFILEDIR"` and `KeyName2 = "VOXFILEDIR"`, is followed by the following code:
		`<gvp:send namelist="Action KeyName1 KeyName2" async="false" />`
		`<gvp:receive maxtime="10s" />`
		This translates to the following SIP `INFO` messages:
		• From the Media Control Platform to the CTI Connector (CTIC)—`Action=GetData&KeyName1=GRAMMARFILEDIR&KeyName2=VOXFILEDIR`
		• From the CTIC to the Media Control Platform—`Action=UserDataResp&sub-action=GetData&Result=Success&GRAMMARFILEDIR=<value1>&VOXFILEDIR=<value2>`

Send attached data to Framework	Object element attribute: `classid="CRData:put"` Enabled the voice application to send data to the server. (Applicable only for interactions between IVR Server and URS.)	

Table 10: Mapping Object Element Extensions Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent

Perform an action on Framework	Object element attribute:
	`classid="CRData:genericAction"`
	Enabled the voice application to obtain additional types of data, such as caller-entered digits (CED) or certain statistics, and to perform certain actions, such as deleting data or getting an access number for remote switch transfers (AccessNumGet).
	More than one object element with `classid="CRData:genericAction"` could be included in the form.
	Example:
	To send data to and from the web server:
	`<object name='myData' classid='CRData:genericAction'>`
	`<param name='IServer_Action' value='CED'/>`
	`<param name='mydata' value='45'/>`
	`.....`
	`.....`
	`</object>`
	The return result is in the following format:
	`<crGetData>`
	`<key name='ResultCode' value='Success'/>`
	`</crGetData>`
		Use the proprietary `<send>` and `<receive>` tags. The `namelist` attribute with the `<send>` tag specifies the required `Action` and `sub_action` (if applicable). The namelist is sent in the body of a SIP `INFO` message.
		GVP 8.1 supports the following actions:
		• `GetData` (see Get data from Framework, on page 51)
		· `AttachData`
		◆ `Add`
		◆ `Replace`
		· `DeleteData`
		◆ `DeleteKey`
		◆ `DeleteAll`
		· `GetStatReq`
		• `PeekStatReq` (for `CurrNumberWaitingCalls` and `ExpectedWaitTime`)
		· `AccessNumGet`
		For more information, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

Table 10: Mapping Object Element Extensions Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent
Send user data through SIP INFO headers	Object element attribute: `Classid="telephonydata:put"` Together with the proprietary <param> element with the append attribute, enabled the voice application to use custom headers in SIP INFO messages to send data to the server. (Supported only on the IPCS.) **Example:** `<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml" xmlns:genesys="http://www.genesyslab.com/vxml/2.0/ext/20020430" xml:lang="en-US"> <form id="user_form"> <object name="user_data" classid="telephonydata:put"> <genesys:param name="msgtype" value="<DATA>" /> <genesys:param name="header" append="<true/false>" value="<DATA>" /> <genesys:param name="body" value="<DATA>" /> </object> </form>`	Use the proprietary <send> tag to send user data in custom headers in SIP INFO messages. If a value already exists for a header, the user-specified value will override the value of that header in the new SIP INFO message that the Media Control Platform generates. Note the following rules: 1. Except for content-type, the NGI does not allow you to use this custom header mechanism to override the known headers. For the list of headers known to GVP, see the Media Control Platform reference information appendix in the *Genesys Voice Platform 8.1 User's Guide*. See also limitations 2 and 3. 2. The NGI does not allow you to override the X-Genesys-CallUUID, X-Genesys-GVP-Session-ID, or X-Genesys-RM-Application-dbid header values. However, any other X-Genesys- header can be overridden, because other X-Genesys- headers are not known to the Media Control Platform and, therefore, limitation 1 does not apply. 3. You can override the value of the content-type header by specifying a different value for the contenttype attribute in the <send> tag. **Note:** The default value of the content-type header is text/plain. If you use the namelist attribute with the <send> tag, then the value of the content-type header is automatically set to application/x-www-form-urlencoded;charset=utf-8 (the value required by the CTIC). You cannot override this value when you specify a namelist. **Note:** Interaction data can similarly be sent with a <transfer> request, and the custom SIP headers are added to the resulting SIP INVITE or REFER.

Table 10: Mapping Object Element Extensions Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent
Receive user data through SIP `INFO` headers	Various objects—`Session`, `Genesys.protocol`, and `Genesys.sip`. Enabled call control data to be exchanged between SIP and the VoiceXML application. (Supported only on the IPCS.) The VoiceXML application used the `Session` object to access all incoming messages, and it used an XML `<object>` tag to send customized SIP messages. The `Genesys.protocol` object indicated the protocol that was being used for call control (`sip` for SIP). To get data from an incoming SIP message, the VoiceXML application used the `Genesys.sip` object. The `Genesys.sip` object was an array of SIP messages. As each new SIP message was received, GVP appended it to the array. The application kept track of which messages it had processed.	Use the proprietary `<receive>` tag to receive data in custom headers in SIP `INFO` messages.

Table 10: Mapping Object Element Extensions Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent

Start transactional recording	Object element attribute: `Classid="transactionalrecord:start"`
	Started transactional recording. Additional parameters were `posturl` and the mode of posting (`sync` or `async`). If the mode of posting was `sync`, the recording was posted directly to the application.
	Example:
	`<object classid="transactional record:start">`
	`<param name="posturl" value="http://... " />`
	`<param name="mode" value= "async" />`
	`</object>`
		Transactional recording is referred to Full Call Recording (FCR), also known as Bidirectional Recording.
		FCR is fully supported. For more information, see the section about Full Call Recording in the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

Stop transactional recording	Object element attribute: `Classid="transactionalrecord:stop"`
	Stopped transactional recording. This tag could not be specified without specifying `transactionalrecord:start`. The `posturl` and mode of posting were optional parameters that, if specified, overrode the values specified in `transactionalrecord:start`. The `namelist` parameter was also optional. The recording and the variables present in the `namelist` parameter were posted to the `posturl` using the specified `mode`.
	Example:
	`<object name="TestRecStop" classid="transactionalrecord: stop">`
	`<param name="posturl" value="http://devtransrec/ upload/capture1.asp" />`
	`<param name="mode" value="sync" />`
	`<param name="namelist" value="mainmenu_input" />`
	`</object>`	

Table 10: Mapping Object Element Extensions Functionality (Continued)

Functionality	Legacy GVP	NGI Equivalent

Free up the ASR MRCP server license during a call	Object element attribute:
	`Classid="asr:freeResource"`
	The GVP interpreter tore down the MRCP session for the call. If there was a subsequent recognition in the call after `classid asr:freeResource` was issued, the interpreter implicitly took care of setting up a new MRCP ASR session, and there was no need to issue a separate `classid` to allocate the ASR resource.
	In addition, the GVP interpreter implicitly freed the MRCP session when bridging a call if there were no active grammars. The MRCP session was freed from both legs of the bridged calls.
	Example:
	`<form id="user_form">`
	`<object name="freeResource"`
	`classid="asr:freeResource">`
	`</object>`
	`</form>`
		A new ECMAScript function, `__ReleaseEngine( engine_name )`, enables this functionality. The function belongs to the `Session` object. When the function is called, the ASR session with name *`engine_name`* is released.
		If the engine specified by *`engine_name`* is not open or if it represents an invalid engine, the request is ignored.
		If `__ReleaseEngine` is called without specifying an engine name, all open ASR engines will be released.
		The NGI will also release all open ASR engines after a successful transfer if there are no active speech grammars, and if the value of the `asr.release_on_transfer` configuration option in the `vxmli` section on the Media Control Platform is `true` (the default value).

VoiceXML $-Variables

Table 11 describes the equivalent NGI support for the functionality provided by GVP 7.6 $-variables.

Table 11: Mapping $-Variable Functionality

Legacy GVP		NGI Equivalent
$-Variable	**Description**	
`$did$`	The called telephone number on which the incoming call came in.	`session.connection.local.uri`
`$ani$`	The caller's telephone number (not always available).	`session.connection.remote.uri`
`$sessionid$`	A unique identifier that the platform generates for the call.	`session.com.genesyslab.sessionid`

Table 11: Mapping $-Variable Functionality (Continued)

Legacy GVP		NGI Equivalent
$-Variable	**Description**	
`$ivr-root-dir$`	The URL of the root directory for the voice application (not including web scripts and the query string).	Not supported.
`$ivr2-root-dir$`	The URL of the root directory for the backup voice application (not including web scripts and the query string).	Not supported.
`$ivr-url$`	The URL of the main web script for the voice application (not including any query strings).	Not supported.
`$ivr2-url$`	The URL of the main web script for the backup voice application (not including any query strings).	Not supported.
`$start-ivr-url$`	The URL of the main web script for the voice application (including any query strings).	Supported by the NGI using `__GetInfo('running_uri')`.
`$last-error$`	The last error encountered by the Voice Communication Server (VCS) or IPCS. Possible values are: • `BAD_XMLPAGE` • `XMLPAGE_NOTFOUND` • `XMLPAGE_TIMEOUT` • `RESOURCE_NOTFOUND` • `RESOURCE_TIMEOUT` • `BAD_XMLTAGNAME` • `BAD_XMLTAGVALUE` • `OPSERVER_ERROR` The VCS or IPCS clears this variable after the next error-free operation.	Use standard VoiceXML error handling: • `<catch>` • `_event` • `_message`
`$last-error-url$`	The URL of the voice application XML page or resource (`.vox` file) that caused the last error.	Use standard VoiceXML error handling: • `<catch>` • `_url`
`$last-error-string$`	A free-format string that the VCS or IPCS code filled in to give diagnostic information about the last error.	Use standard VoiceXML error handling: • `<catch>` • `_message`

Table 11: Mapping $-Variable Functionality (Continued)

Legacy GVP		NGI Equivalent
$-Variable	**Description**	
`$toll-free-num$`	The phone number that the caller dialed to make the call.	In the `gvp.general` configuration section of the IVR Profile, add a parameter named `toll-free-number`, and set the value to `fixed,xxxxxxxx`. The value of this parameter will appear as the `tollfreenum` property on the variable `session.connection.protocol.sip.requesturi`
`$application-name$`	The name of the voice application for the call. The Call Flow Assistant (CFA) used this information to accept or reject the call. It was also used for reporting purposes.	Not applicable.
`$customer-name$`	The name of the customer to bill for the call.	Not applicable.
`$ccerror-telnum$`	The outbound telephone number that the VCS or IPCS code dialed if the CFA and voice applications were not accessible (because of a problem with the data network).	Not applicable.
`$callerhup$`	The VCS or IPCS code set this variable to `true` if the caller terminated the call. The voice application typically used this variable to find out whether the user terminated the call after entering some information in the middle of a form.	Use `<catch>` for `connection.disconnect.hangup`.
`$voicefile format$`	Either the `ACCEPT_CALL` page set this variable, or the first XML page sent back by the voice application set this variable. The VCS or IPCS code needed this value to properly interpret the `.vox` or `.wav` files to be played in the voice application. (Available only in TXML.)	Not supported.

Table 11: Mapping $-Variable Functionality (Continued)

Legacy GVP		NGI Equivalent
$-Variable	**Description**	
`$_toneinput$`	The VCS or IPCS code set this variable to contain the tone that the caller pressed to exit out of a `TONEMAP` (in TXML). A voice application could get the value of this variable by including it as part of the target URL for a subsequent form.	Not applicable.
`$_cparesult$`	The VCS or IPCS code set this variable after a `CREATE_LEG_AND_DIAL` request when the outbound leg had dialed the number. The variable contained the result of the CPA performed by the platform. Possible values were: • `CPA_NORMAL` • `CPA_BUSY` • `CPA_NOANSWER` • `CPA_OPERATORINTERCEPT`	In GVP 8.1.2 release and higher, CPA for VoiceXML transfer is supported. The CPA result will be indicated in the transfer variable. In GVP 8.1.1 release and lower, NGI does not support CPA with `<transfer>`. Therefore there is no equivalent functionality.
`$ivr-error-url$`	The URL on the backup voice application to which the VCS or IPCS performed a `get` request if it encountered any error when making a `get` request to the primary voice application. If the VCS or IPCS could not access this URL, or if there was an error, it dialed out `$cc-error-telnum$`.	Not supported. **Note:** The `alternatevoicexmlurl` parameter, in the `gvp.service-prerequisites` section of the IVR Profile, provides similar functionality. However, `alternatevoicexmlurl` is triggered only at the start of a call, if the `initial-page-url` fails, whereas `$ivr-error-url$` could be invoked at any time during the IVR interaction when an error happened and a catch handler was not found.
`$sid$`	The value of the Script ID as generated by the Queue Adapter or the IVR Server Client.	Not supported.
`$scriptdata$`	Any data specific to the Script ID.	Not supported.
`$scripturl$`	The URL link to the page that the CFA generated in response to run script requests from the IVR Server Client.	Not supported.

Table 11: Mapping $-Variable Functionality (Continued)

Legacy GVP		NGI Equivalent
$-Variable	**Description**	
`$playfilesize$`	This predefined variable supported unified messaging by setting the `playfilesize` after each play. When the caller interrupted the play with the tone input, the voice application knew exactly when the play was interrupted, because it included the predefined variable in the URL.	Use `<mark>`.
`$recordfilesize$`	This predefined variable supported unified messaging by setting the `recordfilesize` after each recording. The voice application could include the predefined variable in the URL as a query string.	*name*`$.size`
`$badxmlpage posturl$`	The voice application could set this variable to point to a URL (within the application) where the platform, upon encountering an XML page with syntax errors, posted the bad XML.	Not directly supported. The Save Temp Files feature, which is implemented through the `com.genesyslab.savetmpfiles` property, enables you to save various types of files (such as prompts, inputs, pages, and recordings) to disk. Use this feature to help identify the parse errors on bad pages.
Variables to enable `Transfer Connect`	$transferscript-url$ $transferscript-number$ $transfer-type$ $transfer-option$	Not supported.

Feature Implementation Changes

This section provides information about changes in the implementation of grammar features between GVP 7.6 and GVP 8.1. It contains information about the following topics:

SISR out Rule

The platform supports the `out` rule as defined in the Semantic Interpretation for Speech Recognition (SISR) specification. The contents of the `out` variable are mapped to VoiceXML forms as described in the VoiceXML 2.0 specification.

The GVP 7.6 interpreter exposed the `out` value as a property of the result variable, and the VoiceXML application accessed the value through the `fieldname.out` variable.

By contrast, the GVP 8.1 NGI evaluates the contents of the `out` variable, and it assigns the correct slot value to the field.

Example

The following is an example of a legacy voice application with a field-level grammar that can have multiple slots, where one slot can match several fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns:telera="http://www.telera.com/vxml/2.0/ext/20020430" xmlns="http://www.w3.org/2001/vxml"
   version="2.0">
<property name="timeout" value="10s"/>
<property name="bargein" value="true"/>
<form>
   <block>
      testing form-level grammar with multiple slots. one slot matches several fields. Press 1.
   </block>
   <field name="Testfield" slot="X">
      <grammar mode="dtmf" type="application/srgs+xml" version="1.0" root="ROOT1"
   xmlns="http://www.w3.org/2001/06/grammar">
         <rule id="ROOT1" scope="public">
            <one-of>
               <item> 1  <tag>out = new Object();out.X='valueX'; out.Y='valueY'; out.Z='valueZ';
   </tag></item>
            </one-of>
         </rule>
      </grammar>
   </field>
   <filled mode="all">
      <if cond=" Testfield.out.X == 'valueX'">pass<disconnect/>
      <else/>fail<disconnect/>
      </if>
   </filled>
</form>
</vxml>
```

In GVP 7.6, when DTMF input of `1` was received, the interpreter exposed the `out` property in `Testfield`, and the voice application could access the semantic interpretation by using `Testfield.out.X`, `Testfield.out.Y`, or `Testfield.out.Z`.

In GVP 8.1, the interpreter evaluates the content of `out` and assigns the correct slot value to `Testfield`. Because `slot X` is specified, the value of `Testfield` will be `valueX`. Using `Testfield.out.X` in the legacy voice application will result in a semantic error.

Undefined <foreach> Array

Unlike in GVP 7.6, the NGI now throws a semantic error (`error.semantic`) when the array of `<foreach>` is undefined.

Built-in Time and Date Grammars

The default built-in time and date grammars now conform correctly to the VoiceXML specification for DTMF input (see `http://www.w3.org/TR/voicexml20/#dmlABuiltins`).

- For time input, the built-in time grammar uses the correct suffix character. The format of recognition results is `hhmmx`, where `x` is:
  - `?`—for inputs 01:00–12:59
  - `h`—for inputs 13:00–23:59 and 00:00–00:59

  In GVP 7.6, the `x` field always had the question-mark value (`?`).
- For date input of 4 or 6 digits, the built-in date grammar places the filler character(s) in the correct position. The format of recognition results is `yyyymmdd`. The standard specifies that, if any of the `yyyy`, `mm`, or `dd` input is missing, the question-mark character (`?`) substitutes for the missing input in the field.

  In GVP 7.6, the question-mark character (`?`) was always inserted at the beginning of the result.

  For example, if the DTMF input is `200401`, the result in GVP 8.1 is `200401??`. In GVP 7.6, the result was `??200401`.

Repeat Rules in DTMF Grammars

The GVPi and NGI differ in the way that they treat DTMF grammars that use *repeat* in rules. The NGI uses a native DTMF recognizer that is strictly compliant with the SISR specification.

"Legacy Usage Example" is an example of a grammar to detect a string of one to four DTMF digits on the GVPi. On the NGI, for strings of two to four DTMF digits, the grammar in the legacy GVP example would return only the last digit entered.

Legacy Usage Example

On the GVPi, when ASR is not enabled or on ASR vendors that support this type of rule, the following DTMF grammar detects a string of one to four DTMF digits:

```
<grammar type="application/srgs+xml" version="1.0" root="ROOT"
   xmlns="http://www.w3.org/2001/06/grammar" mode="dtmf">
   <rule id="ROOT" scope="public">
   <item repeat="1-4">
      <ruleref uri="#digit_gram"/>
   </item>
   </rule>
   <rule scope="public" id="digit_gram">
      <one-of>
         <item> 1 </item>
         <item> 2 </item>
         <item> 3 </item>
         <item> 4 </item>
         <item> 5 </item>
         <item> 6 </item>
         <item> 7 </item>
         <item> 8 </item>
         <item> 9 </item>
         <item> 0 </item>
         <item> * </item>
         <item> # </item>
      </one-of>
   </rule>
</grammar>
```

NGI Usage Example

To obtain the same result on the NGI as on the GVPi, rewrite the grammar with semantic interpretation tags, as shown in the following example:

```
<grammar type="application/srgs+xml" version="1.0" root="ROOT"
   xmlns="http://www.w3.org/2001/06/grammar" mode="dtmf">
   <rule id="ROOT" scope="public">
      <tag>out="";</tag>
      <item repeat="1-4">
         <ruleref uri="#digit_gram"/>
      </item>
      <tag>out += meta.current().text;</tag>
   </rule>
   <rule scope="public" id="digit_gram">
      <one-of>
         <item> 1 </item>
         <item> 2 </item>
         <item> 3 </item>
```

```
        <item> 4 </item>
        <item> 5 </item>
        <item> 6 </item>
        <item> 7 </item>
        <item> 8 </item>
        <item> 9 </item>
        <item> 0 </item>
        <item> * </item>
        <item> # </item>
     </one-of>
  </rule>
</grammar>
```

Offboard DTMF Recognition

In releases of GVP up to and including 8.1.0, the NGI does not support offboard DTMF recognition through an ASR server. From release 8.1.1, NGI does support offboard DTMF recognition.

For the NGI, ensure that the DTMF grammars in migrated legacy voice applications do not make use of features that are specific to your ASR vendor. NGI applications must comply with the W3C Speech Recognition Grammar Specification (SRGS).

Embedded Grammars

In releases of GVP up to and including 8.1.0, the NGI does not support a DTMF grammar referencing another grammar. From release 8.1.1 the NGI does support such embedded grammars but only using offboard DTMF recognizers.

Example (in bold text):

```
<grammar type="application/srgs+xml" xml:lang="en-US" version="1.0"
mode="dtmf" root="root">
   <rule id="root" scope="public">
      <ruleref uri="grammars/misc/testfield.grxml" />
   </rule>
</grammar>
```

Workaround (pre-8.1.1)

Do one or both of the following:

- Use ASR.
- Flatten all embedded DTMF grammar references into one DTMF grammar.

MIBs and Traps

Table 12 describes the equivalent NGI support for the event handling provided by GVP 7.6 Management Information Bases (MIBs) and traps.

Table 12: Mapping MIBs and Traps

Legacy GVP		NGI Implementation
Trap ID	**Description**	
cnNetworkTimeout	• RESOURCE_TIMEOUT • XMLPAGE_TIMEOUT Thrown whenever a page or a resource fetch failed because of a timeout.	The NGI generates a Simple Network Management Protocol (SNMP) trap, NGI_LOG_FETCH_RESOURCE_TIMEOUT, for a network timeout that occurs while any resource is being fetched. The trap details include the session ID and URI.
cnResourceNotFound	• RESOURCE_NOTFOUND • XMLPAGE_NOTFOUND Thrown whenever a resource or XML page was not found.	The NGI generates an SNMP trap, NGI_LOG_FETCH_RESOURCE_ERROR, for any resource that is not found when it is fetched. The trap details include the session ID and URI.
cnNetworkError	• NETWORK_ERROR • HTTP_ERROR • PAGECOLLECTOR_ERROR Thrown whenever a page fetch failed because of a network, HTTP, or PageCollector error.	The NGI generates an SNMP trap, NGI_LOG_FETCH_RESOURCE_ERROR, for any network error (while fetching resources) other than network timeout. The trap details include the session ID and URI.
cnBadXMLPage	• BAD_XMLPAGE • BAD_XMLTAGNAME • BAD_XMLTAGVALUE • XMLVERSION_INCOMPATIBLE • SCRIPT_ERROR • VXML_EXEC_ERROR Thrown whenever parsing an XML page resulted in an error because of syntax.	The NGI generates an SNMP trap, NGI_LOG_PARSE_ERROR, for any VoiceXML parse error. The trap details include the session ID and the URI of the page that generated the parse error.
cnCCErrorTrap	Thrown for CFA errors.	Not applicable.
cnIVRErrorTrap	Thrown for IVR failures.	Not supported.

Table 12: Mapping MIBs and Traps (Continued)

Legacy GVP		NGI Implementation
Trap ID	**Description**	
cnDialError	• OUTBOUNDCALL_FAILURE • DIALERROR • NOPORTS • BUSY • NETWORK_BUSY • ANSWERFAILED • INVALID_DIGITS • TELEPHONY_ERROR Thrown for outbound calling errors.	Not applicable.
cnCallDropped	Thrown whenever a call was dropped because of any error during the start of the call.	Not applicable.
cnAbnormalCall Termination	Thrown whenever a call was dropped because of any error during the middle of IVR handling.	Not applicable.
cnCallRerouted	Thrown whenever a call was rerouted to a specified reroute number because of errors (for example, a network error in fetching the `did.xml` file) during the start of the call.	Not applicable.
cnCallTransferred	Thrown whenever a call was transferred to a specified telephone number (`ccerror-telnum`) because of errors during the middle of IVR handling.	Not applicable.
cnTelephonyError	Thrown whenever there were telephony errors.	Not applicable.
cnOutOfMemory	Thrown when `new` failed.	Not supported.

Chapter

4 Migrating TXML Applications

In general, the Genesys Voice Platform (GVP) 8.1 Next Generation Interpreter (NGI) does not support Telera XML (TXML). For the few specific features that the Media Control Platform and NGI do directly support, this chapter maps the legacy TXML features to GVP 8.1 NGI support, to provide detailed information about the changes that are required in the TXML application itself. However, this chapter also shows how more of the legacy TXML features can be supported through Call Control Extensible Markup Language (CCXML), if you include the Call Control Platform in your deployment.

This chapter contains the following sections:

For more information about the Voice Extensible Markup Language (VoiceXML) features that are supported on the GVP 8.1 NGI, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

For more information about the TXML features that are supported on the GVP 8.1 GVPi, see the *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual*.

For more information about all the legacy TXML features that were supported on GVP 7.6, see the *Genesys Voice Platform 7.6 VoiceXML 2.1 Reference Manual*.

For more information about the CCXML features that are supported, see the *Genesys Voice Platform 8.1 CCXML Reference Manual*.

Application Migration

Table 13 describes the equivalent NGI support for the functionality provided by GVP 7.6 TXML tags and features.

Table 13: Mapping TXML Tags and Features

Legacy GVP	NGI Equivalent
<CREATE_LEG_AND_DIAL> Enables a voice application to make outbound calls, with better control of the called party (the second leg) than the VoiceXML <transfer> element provides. **Syntax:** <CREATE_LEG_AND_DIAL TELNUM="telephone number to be dialed" IVRURL="URL" ANI="Callers phone number" BRIDGE="YES | NO" ENDSESSIONHUP="YES | NO" URL_ONLEG2HUP="URL" CPATIMEOUT="timeinsecs" AFTERCONNECTTIMEOUT="timeinsecs" CALLTRIGGEREVENT="connected,alerting, callproceeding,100-199,1xx" /> For more information about the <CREATE_LEG_AND_DIAL> attributes, see the *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual*.	The <transfer> tag and its attributes enable the NGI to support some of the features of <CREATE_LEG_AND_DIAL>: • TELNUM translates to the dest or destexpr attributes of <transfer>. • IVRURL translates to the consultexpr attribute of <transfer>. **Note:** IVRURL can have the value "LEG_WAIT", which is not supported in consultexpr. • ANI translates to the signalvar attribute of the <transfer> tag. The signalvar attribute is a GVP extension that must be qualified by the GVP namespace (gvp:signalvar). • BRIDGE translates to the bridge attribute of <transfer>, but note the following: ◆ BRIDGE=YES—In the legacy GVP interpreter (GVPi), this value means that whisper transfer is disabled. In the NGI, this value means that whisper transfer is still enabled. ◆ BRIDGE=NO—In the GVPi, this value means that whisper transfer is enabled. It also means that the call remains on the platform even after bridging (after BRIDGE_CALL). In the NGI, this value means that the call will be dropped after the transfer is complete. Whisper transfer is possible only if you also specify type=consultation (not blind). • CPATIMEOUT translates to the connecttimeout attribute of <transfer>. • $_cparesult$ has some equivalents in the value of the transfer variable (the name attribute of <transfer>). For more information, see page 59. Whisper transfer is implemented as a multiphase transfer in the NGI. For more information, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

Table 13: Mapping TXML Tags and Features (Continued)

Legacy GVP	NGI Equivalent
<BRIDGE_CALL> Bridges the current call with another call on the same gateway. When the voice application executes the <BRIDGE_CALL> element, it interrupts the leg of the bridged call and aborts any messages being played on the leg. **Syntax:** <BRIDGE_CALL LEG_ID="NAME" />	Not supported on the NGI. However, you can achieve similar functionality by using CCXML applications with the Call Control Platform. For more information, see "Using CCXML" on page 72.
<UNBRIDGE_CALL> Breaks the bridge between designated legs of a call. After the calls are unbridged, the voice application executes the next element after the <UNBRIDGE_CALL> element. If there is no element, the voice application executes the next document from the HREF specified in the XMLPage root element at the top of the page. **Syntax:** <UNBRIDGE_CALL LEG_ID="name" OTHER_LEG_URL="URL" />	Not supported on the NGI. However, you can achieve similar functionality by using CCXML applications with the Call Control Platform. For more information, see "Using CCXML" on page 72.
<LEG_WAIT> Places the interpreter context receiving this element into a Wait state. The conversation controller then waits for one of the following: • An event from the voice application (for example, caller hangup) • An interrupt from the interpreter • An <ALERT_LEG> element from the voice application **Syntax:** <LEG_WAIT TIMEOUT="time in seconds" HREF="URL" />	Not supported.

Table 13: Mapping TXML Tags and Features (Continued)

Legacy GVP	NGI Equivalent

`<QUEUE_CALL>`
When the voice application executes this element, it sends a `QUEUE_CALL_REQ` request to the interpreter context, which calls the Call Router to queue the call.
Syntax:
`<QUEUE_CALL USR_PARAMS="comma separated key-value pairs" AGENT_URL="URL" />`
	You can achieve the same functionality by using the `<transfer>` tag to transfer the call to a route point through the CTI Connector (CTIC). See "Route Requests" on page 19. For more information, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.
	In Composer Voice, use the `Route Point` CTI block.

`<SCRIPT_RESULT>`
The voice application executes this element in response to a `RUN_SCRIPT_REQ` from the Call Router. When the voice application executes this element, it sends a `SCRIPT_RESULT` request to the interpreter context. The interpreter context then presents the data to the Call Router as a response to the script request.
Syntax:
`<SCRIPT_RESULT USR_PARAMS="CDATA" />`
	Use the `<exit>` tag with the `namelist` attribute to pass interaction data back to the strategy in the SIP `BYE` message.
	Example:
	The following code in the VoiceXML application:
	`<script>`
	`<assign name="CED" expr="1"/>`
	`<assign name="UData_param1" expr="3"/>`
	`<assign name="UData_param2" expr="4"/>`
	`<assign name="ExtnsEx_param1" expr="5"/>`
	`<assign name="ExtnsEx_param2" expr="6"/>`
	`</script>`
	`<exit namelist="CED UData_param1 Udata_param2 ExtnsEx_param1 ExtnsEx_param2"/>`
	yields the following `BYE` message body:
	`CED=1&Udata_param1=3&Udata_param2=4&ExtnsEx_param1=5&ExtnsEx_param2=6`
	Note: All `UData` variables must be prefixed with *UData_* and all `ExtnsEx` variables must be prefixed with *ExtnsEx_*. Because there is only one variable for caller-entered digits (CED), no prefix is required for the variable named *CED*.
	For more information, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

Table 13: Mapping TXML Tags and Features (Continued)

Legacy GVP	NGI Equivalent
<REXFER> Transfers a call from one agent to another. This involves unbridging the inbound and outbound legs, dropping the current outbound leg, creating a new outbound leg, dialing a new phone number on the new outbound leg, and, finally, bridging the inbound and new outbound legs. **Syntax:** <REXFER TELNUM="telephone number" LEG_ID="name" IVRURL="URL />	Not supported.
<SET> Provides stateless voice applications with the mechanism to maintain state and use session variables for any web server environment. Voice applications can use the <SET> element to define session variables (variables that last the lifetime of the current session/call). The <SET> element is also used to set certain standard variables for the TXML interpreter. **Syntax:** <SET VARNAME="name" VALUE="value" />	Set equivalent VoiceXML session variables.
<ALERT_LEG> Used after unbridging a call, to alert the other leg and direct it to a specified URL. **Syntax:** <ALERT LEG LEG_ID="name" IVRURL="URL" />	Not supported.

Table 13: Mapping TXML Tags and Features (Continued)

Legacy GVP	NGI Equivalent
<HANGUP_AND_DESTROY_LEG> Instructs the application to terminate the call on this leg, and to destroy the interpreter leg receiving this element. **Syntax:** <HANGUP_AND_DESTROY_LEG REASON="NORMAL | ERROR | CALLERHUP" />	Not supported.
<END_SESSION> Destroys all legs of the session, and hangs up all of the associated calls. **Syntax:** <END_SESSION/> The <END_SESSION> element has no attributes.	Not supported.
<ON_LEGHUP> Specifies whether the session should be ended when the leg hangs up. The default behavior is to end the session when any of the legs hangs up the call. However, if you set the ENDSESSION attribute, the application can control whether the entire session should be ended, or whether only that call leg ends. **Syntax:** <ON_LEGHUP ENDSESSION="NO" OTHER_LEG_URL="<%=m_agentafterURL%>" />	Not supported.

Using CCXML

As Table 13 shows, the GVP 8.1 NGI does not support many legacy TXML features on its own. This section describes how you can add a CCXML application, and use the CCXML Interpreter on the Call Control Platform to achieve additional functionality.

Multiple Bridging and Unbridging of Calls

TXML can <bridge> and <unbridge> a call multiple times, possibly to transfer the call to different agents during the lifetime of the call. The GVP 8.1 NGI can bridge a call only once, and it does not support multiple bridging and unbridging. However, it is possible to use CCXML to provide this feature.

For example, the Media Control Platform cannot support the following scenario if the deployment includes only the NGI and VoiceXML, but it can support this scenario if the deployment includes the Call Control Platform (CCXML Interpreter) and uses CCXML.

Multiple-Bridging Scenario

1. The caller calls in and is connected to a VoiceXML application.
2. After self-service, the caller needs to be transferred to an agent.
3. An agent leg is created, and it is connected to a VoiceXML application that can do some initial play (in other words, “whisper”).
4. The caller and agent are bridged.
5. At some point, the caller and agent are unbridged because they are forced from the CCXML application, not because the agent drops the call.
6. The agent leg is terminated.
7. A second agent leg is created, and the VoiceXML application performs some whisper function.
8. The second agent leg is bridged to the caller.

CCXML Code Sample

The following CCXML code sample serves the preceding, multiple-bridging scenario:

```
<?xml version="1.0" encoding="UTF-8"?>
<ccxml xmlns="http://www.w3.org/2002/09/ccxml" version="1.0">
 <!-- Create our ccxml level vars -->
 <var name="in_connectionid" expr="''" />
 <var name="dialog_1_id" expr="''" />
 <var name="dialog_2_id" expr="''" />
 <var name="agent_1_id" expr="''" />
 <var name="conference_id" expr="''" />
 <var name="loop_counter" expr="'1'" />
 <var name="currentstate" expr="'initial'" />

 <eventprocessor statevariable="currentstate">

  <transition state="initial" event="connection.alerting" name="evt">
    <assign name="in_connectionid" expr="evt.connectionid"/>
     <accept connectionid="in_connectionid"/>
  </transition>
  <transition event="connection.connected" name="evt">
      <if cond="in_connectionid==evt.connectionid">
         <assign name="currentstate" expr="'in_connected'"/>
         <dialogstart src="'file:///usr/local/phoneweb/samples/firstpage.vxml'"
                   dialogid="dialog_1_id"
                   connectionid="in_connectionid"/>
      </if>
  </transition>
```

```
   <transition event="dialog.started" name="evt">
      <if cond="dialog_1_id==evt.dialogid">
         <assign name="currentstate" expr="'dialog_1_started'"/>
      </if>
      <if cond="dialog_2_id==evt.dialogid">
         <assign name="currentstate" expr="'dialog_2_started'"/>
         <send target="session.id" targettype="'ccxml'" name="'self.timer'" delay="'10s'"/>
      </if>
   </transition>
   <transition event="dialog.disconnect" name="evt">
      <if cond="dialog_1_id==evt.dialogid">
         <assign name="currentstate" expr="'dialog_1_terminated'"/>
         <createcall dest="'sip:agent1@10.0.0.2'" connectionid="agent_1_id"/>
      </if>
   </transition>
   <transition event="connection.connected" name="evt">
      <if cond="agent_1_id==evt.connectionid">
         <assign name="currentstate" expr="'agent_1_connected'"/>
         <createconference conferenceid="conference_id"/>
      </if>
   </transition>
   <transition event="conference.created" name="evt">
      <if cond="conference_id==evt.conferenceid">
         <assign name="currentstate" expr="'dialog_2_created'"/>
         <join id1="conference_id" id2="in_connectionid"/>
         <join id1="conference_id" id2="agent_1_id"/>
         <dialogstart src="'file:///usr/local/phoneweb/samples/secondpage.vxml'"
                   dialogid="dialog_2_id"
                   conferenceid="conference_id"/>
      </if>
   </transition>
   <transition event="error.fetch" name="evt">
      <if cond="loop_counter=='1'">
         <disconnect connectionid="agent_1_id"/>
         <createcall dest="'sip:agent1@10.0.0.2'" connectionid="agent_1_id"/>
         <assign name="loop_counter" expr="'2'"/>
      </if>

      <if cond="loop_counter=='2'">
         <exit/>
      </if>
  </transition>
 </eventprocessor>
</ccxml>
```

Chapter

5 Migrating From Legacy VoiceGenie to GVP 8.1 NGI

This chapter is intended for customers who want to migrate legacy Genesys VoiceGenie voice applications to enable capability with the GVP 8.1 Next Generation Interpreter (NGI).

The VoiceGenie Media Platform includes a VoiceXML Interpreter component. This component requests Voice Extensible Markup Language (VoiceXML) pages from a web application server (through a caching proxy), compiling these pages into an internal representation, and then executing them in order to manage a dialog with a user. As part of this dialog management process, the VoiceXML Interpreter also requests resources (such as speech recognition and speech synthesis sessions) from other platform components.

This chapter also includes a mapping of legacy VoiceGenie features to GVP 8.1 NGI supported features and provides detailed information about the changes that are required in the VoiceXML application itself.

This chapter contains the following sections:

General Changes

The following section describes general behavior changes between the legacy VoiceGenie interpreter and the GVP 8.1 NGI.

Strict XML Parser Implementation

The legacy VoiceGenie interpreter uses a relaxed method of parsing on documents that it accepts and executes. It has the following characteristics:

- The legacy interpreter executes certain classes of malformed Extensible Markup Language (XML) documents. For example, if the closing `</vxml>` element is missing, the document is still executed.
- The legacy interpreter accepts special XML characters, such as `&` and `<`, even if they are not XML-escaped characters. In NGI, unescaped XML characters cause parsing errors.
- The legacy parser accepts invalid attributes (that is, attributes that are not part of the VoiceGenie implementation of VoiceXML 2.0 or 2.1, or the VoiceGenie extension set)—the invalid attributes are simply ignored. In the NGI, invalid attributes cause parsing errors.
- The legacy parser accepts an invalid element (that is, an element that is not part of the VoiceGenie implementation of VoiceXML 2.0 or 2.1, or the VoiceGenie extension set)—the invalid element is simply ignored, as are its child elements (whether valid or not). In NGI, invalid elements cause parsing errors.
- The legacy interpreter accepts, in some cases, attribute values that are not enclosed in quotation marks. In the NGI, unenclosed attribute values cause parsing errors.
- The legacy interpreter accepts multiple grammars on the same page with the same id. In NGI, this causes parsing errors.

In NGI, parsing is done in strict accordance with the VoiceXML schema; any elements or attributes that violate the schema cause parsing errors.

Note: To avoid this issue, ensure that your VoiceXML documents are valid XML documents.

Time Designations

If an integer appears where a Time Designation is required (that is, if it does not have the *s* or *ms* suffix, as required by VoiceXML 2.0/2.1), the legacy VoiceGenie interpreter logs a warning and treats the value as milliseconds. If NGI encounters such a value, it generates an `error.semantic` event.

Note: To avoid this issue, ensure that all time measurements include the appropriate unit designation

Access to Application Temporary Data

The `savetmpfiles` property was developed in the legacy VoiceGenie interpreter to control whether the media platform permanently saves temporary files to the file system for debugging purposes. In the legacy VoiceGenie interpreter, the `savetmpfiles` property uses a hexadecimal value, in which each bit represents an application component that the application developer wants to save to the disk—for example, audio files or grammars. In practice, most applications use `0x0` or `0xffffff` as the value for `savetmpfiles`, to completely enable or disable the `savetmpfiles` feature for all types of files.

In NGI, the `com.voicegenie.savetmpfiles` feature uses a list of keywords to represent the different application components that are to be saved. Specifying the keyword `all` causes the Media Control Platform to save everything, whereas specifying the keyword `none` causes it to save nothing. The values `0x0` and `0xffffff` (for any number of `f`s) are equivalent to the keywords `none` and `all`, respectively. For more information about the available keywords, see the *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help*.

Changes to Shadow Variables

The following section describes what has changed in the VoiceGenie shadow variables.

application.lastresult$[i].rawresults

In the legacy VoiceGenie interpreter, the `application.lastresult$[i].rawresults` variable is used to expose detailed information about the recognition results. It uses a cumbersome text format to convey the extra information, such as per-slot confidence and ambiguous results.

In NGI, the `rawresults` variable is no longer available. Instead, the following new variables have been added that indirectly correspond to `rawresults`:

- `application.lastresult$.xmlresult` exposes the Natural Language Semantics Markup Language (NLSML) document as a Document Object Model (DOM), in the same way that a `<data>` object exposes an arbitrary XML document.
- `application.lastresult$[i].ambig_interpretation` exposes any ambiguous results that are returned from the Automatic Speech Recognition (ASR) engine. The variable is an ECMAScript array, in which each element represents an `<instance>` element within the `<interpretation>` element of the NLSML result. If there is only one `<instance>` element within the `<interpretation>` element, the `application.lastresult$[i].ambig_interpretation` variable is set to ECMAScript `undefined`.

- Each descendent node in the `application.lastresult$[0].interpretation` variable may have a corresponding `$.confidence` shadow variable if the corresponding NLSML document node has a per-slot confidence.

If the application relies on the values in the `rawresults` shadow variable, it must be updated so that it uses ECMAScript to walk through and parse the DOM object that is returned in the new shadow variable representation.

application.lastresult$.activegrammar and application.lastresult$.triggeredgrammar

In NGI, these variables are set for DTMF grammars, as well as for ASR grammars.

NGI uses the `application.lastresult$.triggeredgrammar.linenumber` variable for a `<choice>` element within a `<menu>` element as the line number of the menu itself. The legacy VoiceGenie interpreter used the actual line number of the `<choice>` element.

application.lastresult$.triggeredgrammar.pageuri

In the legacy VoiceGenie interpreter, the Uniform Resource Indentifier (URI) is exposed to the application in either the `application.lastresult$.triggeredgrammar.pageuri` or the `application.lastresult$.activegrammars[i].pageuri` shadow variable. The URI parameters are removed in the `application.lastresult$.activegrammars[i].pageuri`. NGI preserves the URI parameters and exposes the entire URI.

In NGI, the `application.lastresult$.triggeredgrammar.pageuri` variable is undefined when a hotkey (universal) grammar is matched.

Removal of saveutterance-Related Shadow Variables

Because VoiceXML 2.1 has standardized the `recordutterance` feature, NGI no longer supports the VoiceGenie `com.voicegenie.saveutterance` extension property . As a result, the following `application.lastresult$` shadow variables are no longer available:

- `application.lastresult$.utteranceaudio`
- `application.lastresult$.duration`
- `application.lastresult$.size`

These variables must be replaced with the `recording`, `recordingduration`, and `recordingsize` shadow variables, respectively. For further details, see `http://www.w3.org/TR/voicexml21/#sec-reco_rec`.

Other Variables Removed

The following shadow variables have been removed in NGI:

- application.lastresult$.info
- Input item–level shadow variable <name>$.info
- Input item–level shadow variable <name>$.bargeinphrase
- Input item–level shadow variable <name>$.bargeinscore

In the legacy VoiceGenie interpreter, the bargein-related shadow variables were supported only with the Watson ASR engine.

Access to <record> Metadata

In the legacy VoiceGenie interpreter, when a variable that represents a recording (either from a <record> input item or from an utterance recording) is logged through the <log> tag, the recording's file path is logged. Due to the changed internal structure of NGI, this information is no longer available from the application environment. The recording file location is logged to the metrics file.

Full Call Recording

In NGI, the format of the Full Call Recording (FCR) instructions has changed. Semicolons are no longer used as command delimiters and are simply ignored (or treated as part of the command). Only line breaks are treated as command delimiters.

As a result, it is no longer possible to have multiple commands on a single line delimited by semicolons.

For example:

```
<log vg:dest="calllog">directory /usr/local/phoneweb/callrec; enable
callrec recsrc=mixed type=audio/x-wav; keep-files true;</log>
```

This must be changed to:

```
<log vg:dest="calllog">
        directory /usr/local/phoneweb/callrec
        enable callrec recsrc=mixed type=audio/x-wav
        keep-files true
</log>
```

Alternate Page Handling

In the legacy VoiceGenie interpreter, when the initial application page cannot be loaded, the alternate page (as specified in the application provisioning) is fetched and parsed, to support failover to an alternate server or application. However, the page does not begin execution as if it were a new page. Instead, after the page has been loaded, the error handler for error.badfetch is selected and executed.

Generating an error immediately is not a good choice for the behavior in this case, because it is not clear:

- Which VoiceXML scopes have been initialized before the error is generated.
- Which scope the error is generated in.
- What happens when an error is encountered during scope initialization.

In NGI, when the initial page cannot be loaded, the alternate page is fetched and parsed, and the page is executed from the beginning.

Pre-Fetching

In the current version of NGI, pre-fetching functionality is not available. If pre-fetching is required, use the legacy VoiceGenie interpreter for the application.

Prompt Queue Fetching

In the legacy VoiceGenie interpreter, the `<audio>` files in the entire prompt queue are fetched from the HTTP server before the first audio file is played (and the alternate audio is enclosed within the `<audio>` elements). In NGI, the the `<audio>` files are fetched from the HTTP server as each is played. This improves the time-to-first-audio because NGI no longer waits until the audio files from the entire prompt queue are fetched before the first audio file is played to the caller.

The legacy VoiceGenie interpreter supports the value stream for the `fetchhint` property. NGI no longer supports this value for `fetchhint`.

TDD

In the current release of NGI, support for Telecommunication Device for the Deaf (TTD) grammars is not available. If TDD is required, use the legacy Voice Genie interpreter for the application.

Page Fetch Changes

Initial Page

In NGI, errors in the initial page (that is, anything that would generate an `error.badfetch`) cause an incall rejection. In contrast, the legacy VoiceGenie interpreter generates an `error.badfetch` to the same page, or possibly the default page. In other words, the legacy VoiceGenie interpreter allows a page that would normally generate an `error.badfetch` to begin execution, and generates the error only if the offending code is encountered.

Transitions with fetchaudio

When fetching pages, scripts, or data with `fetchaudio` specified, NGI begins fetching the target page while queued prompts are playing. The legacy VoiceGenie interpreter waits until these prompts have been played before beginning the fetch.

Submitting Objects

If an ECMAScript object is used in a namelist to fetch something from a web server (for example, `<submit>` or `<data>`), the legacy VoiceGenie interpreter can resolve the sub properties of the object and add the individual properties as parameters to the HTTP request.

For example:

```
<block>
   <script>
      var a = new Object();
      a.foo1 = "hey";
      a.foo2 = "you";
   </script>
   <submit next="next.html" namelist="a" method="get"/>
</block>
```

In the example above, the legacy VoiceGenie interpreter creates an HTTP request with the following Request-URI (because the method is get, the parameters are added to the URI):

```
http://<server>/next.html?a%2Efoo1=hey&a%2Efoo2=you
```

To achieve similar functionality with NGI, the application must use the toSource() function of ECMAScript objects. The preceding example would be translated to:

```
<block>
   <script>
      var a = new Object();
      a.foo1 = "hey";
      a.foo2 = "you";
      var toSend = a.toSource();
   </script>
   <submit next="next.html" namelist="toSend" method="get"/>
</block>
```

From this script, NGI creates an HTTP request with the following `Request-URI`:

```
http://10.0.0.12/next.html?toSubmit=%28%7Bfoo1%3A%22hey%22%2C%20foo
2%3A%22you%22%7D%29
```

After unescaping the `toSubmit` URI parameter, this becomes:

```
({foo1:"hey", foo2:"you"})
```

Note: Application developers may need to change the HTTP server code to accommodate these differences.

Submitting Arrays

If an array is used in a namelist to fetch something from a web server (for example, <submit> or <data>), the legacy VoiceGenie interpreter can resolve each element of the array and add the elements as individual parameters to the HTTP request.

For example:

```
<block>
   <script>
      var foo = [ 1, 2, 3 ];
   </script>
   <submit next="next.html" namelist="foo" method="get"/>
</block>
```

In the example above, the legacy VoiceGenie interpreter creates an HTTP request with the following `Request-URI` (because the method is get, the parameters are added to the URI):

```
http://<server>/next.html?foo1=1&foo2=2&foo3=3
```

To achieve similar functionality with NGI, the application must use the `toSource()` function (which serializes the array into Java Script Object Notation [JSON] format). The preceding example would be translated to:

```
<block>
   <script>
      var foo = [ 1, 2, 3 ];
      var toSend = foo.toSource();
   </script>
   <submit next="next.html" namelist="toSend" method="get"/>
</block>
```

From this, the NGI creates an HTTP request with the following Request-URI:

```
http://10.0.0.12/next.html?toSubmit=[1,%202,%203]
```

Note: Application developers may need to change the HTTP server code to accommodate these differences

VoiceXML Behavior Changes

This section describes how the interpreters behave with each VoiceXML version.

VoiceXML 1.0

The legacy VoiceGenie interpreter implements the following VoiceXML 1.0 features, which are not supported in NGI:

- Documents in which the version attribute of the <vxml> element is set to 1.0—These will be rejected and cause a parsing errors.
- `session.telephone.* session variables`—The application must use the `session.connection.* session` variables instead.
- `telephone.*` events—(controlled by the platform configuration).
- Use of the `<dtmf>` element to define DTMF grammars—The application must use the `<grammar>` element with the mode attribute.
- Playback of recorded input from a `<value>` element—By specifying the `<value>` element with an expression that references an input item that is collected with the `<record>` element, the legacy VoiceGenie platform is able to play back the recorded audio. The application must use the `<audio>` element with the `expr` attribute instead. For further details, see `http://www.w3.org/TR/voicexml20/#dml4.1.3`.
- The `<sentence>` and `<paragraph>` speech markup elements—These have been replaced by the `<s>` and `<p>` elements, as specified by a more recent release of the Speech Synthesis Markup Language (SSML) specification.
- SSML `<audio>` and `<value>` elements as children of the `<choice>` element—These child elements are no longer supported within `<choice>`.

VoiceXML 2.0

Table 14 describes the changes in elements between the legacy VoiceGenie interpreter and NGI.

Table 14: Element Changes

Element	Legacy VoiceGenie Interpreter	NGI
`<catch>`	Uses the anonymous variable `_disconnect_reason` to provide the disconnect reason.	Offers this information with the `_message` variable.
`<choice>`	Supports the `fetchaudio` and `fetchaudiominimum` attributes of the `<choice>` element.	The `fetchaudio` and `fetchaudiominimum` attributes are not supported. If it is necessary to specify the different `fetchaudio` properties on a per-choice basis in the application, use the `<form>` and the `<field>` elements. NGI does not support SSML content within the `<choice>` element.

Table 14: Element Changes (Continued)

Element	Legacy VoiceGenie Interpreter	NGI
<grammar>	When a built-in grammar ,such as <grammar src="builtin:grammar/builtin_name" /> is matched, the legacy VoiceGenie interpreter exposes the matched rule name in the application.lastresult$.triggered grammar.rulename variable, by setting it to the entire builtin:grammar/builtin_name grammar URI.	The information is exposed through the same variable, but it is now exposed simply the built-in grammar name—that is, builtin_name. NGI does not permit multiple ID's on the same page with the same value in accordance with the xml:id schema restriction. In other words, for any given VoiceXML page, no two <form> element or <grammar> element can have the same ID.
<link>	Resolves the URI according to the base URI of the page that contains the link.	The behavior of the <link> element has been corrected so that it more closely conforms to the VoiceXML specification. Specifically, NGI resolves the URI that is specified in a link according to section 2.5 of the specification, which states, "any URIs in an attribute of the link element are resolved dynamically, i.e. according to the base URI in effect when the link's grammar is matched."
<log>		All log data is written in UTF-8 format.
<record>	If there is no type attribute in the <record> element to specify the desired recording format, and if the platform is set up to use ulaw by default, the legacy VoiceGenie interpreter exposes the field item name$.filetype as audio/x-vox shadow variable. If the <record> element contains an invalid type attribute, the legacy VoiceGenie interpreter uses audio/vox as the default value, and the recording is made. The legacy VoiceGenie interpreter supports the beginsilence attribute for the <record> element.	This value is exposed as audio/basic. NGI logs a warning message indicating that the MIME type is bad, and the recording will fail. The attribute has been renamed to timeout because it has semantics that are identical to the timeout extension attribute of the <field> element, and it overrides the timeout property. Additionally, the current order of precedence is: 1. timeout attribute of the last prompt 2. timeout attribute of <record> 3. timeout property

Table 14: Element Changes (Continued)

Element	Legacy VoiceGenie Interpreter	NGI
<script>	Supports the UTF-8, UTF-16, and ASCII character sets for the <script> element.	Supports only the UTF-8 and UTF-16 character sets. If any other character set is specified for the <script> element, an error.unsupported.scriptcharset event is generated when the <script> element is executed.
<subdialog>		When the <param> element is used to pass data to a subdialog, NGI copies variables by value. For object types, NGI attempts to duplicate the object in order to pass it by value. Record variables (created when the <record> tag is used) can be passed to a subdialog. The metadata stored in the copied record object is the same as in the original object. DOM objects that are created as a result of the <data> element, or from the application.lastresult$.xmlresult variable are not supported. The DOM data will not be accessible within the subdialog. When an ECMAScript object is duplicated within the subdialog, the attributes for array item properties are not copied. Function properties are not copied either.

VoiceXML 2.1

With VoiceXML 2.1, the World Wide Web Consortium (W3C) has standardized a number of extensions that were previously proprietary. As a result, the following old extensions are no longer supported by NGI:

- com.voicegenie.saveutterance <property>—Use the input item recording shadow variable instead. For further details, see http://www.w3.org/TR/voicexml21/#sec-reco_reco.
- The expr attribute of <grammar>—Use the srcexpr attribute instead. For further details, see http://www.w3.org/TR/voicexml21/#sec-grammar_expr.
- The expr attribute of <script>—Use the srcexpr attribute instead. For further details, see http://www.w3.org/TR/voicexml21/#sec-script_expr.

VoiceGenie Extensions

This section describes the VoiceGenie extensions for the interpreters.

Namespace Prefixes

In the legacy VoiceGenie interpreter, there are several extension elements and attributes that are used without XML namespace prefixes—for example, the `<send>` element, and the volume attribute of the `<audio>` element. In NGI, all extensions must have XML namespace prefixes. The namespace URI is configurable, and it is set to `http://www.genesyslab.com/2006/vxml21-extension` by default.

The following is an example of how to use namespaces in the extension element and attribute:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns=http://www.w3.org/2001/vxml
xmlns:gvp="http://www.genesyslab.com/2006/vxml21-extension" >
   <form>
      <var name="offset" expr="1+2" />
      <block>
         <gvp:send ... />
         <audio src="123" gvp:volume="5" />
      </block>
   </form>
</vxml>
```

Call-Control Extensions

In NGI, the call-control-related extensions are not available. Application developers should now use the Call Control Extensible Markup Language (CCXML) instead. As a result, the following call-control features are no longer supported in NGI:

- The call-control extension elements `<call>`, `<fork>`, `<join>`, and `<release>`— Multiple sessions and conferencing are supported through CCXML.
- Conference features—A conference can be invoked by other means when a new call goes into the media platform. The media platform remains capable of mixing conferences.
- Talk-Listen-Toggle (TLT).
- The `name` and `chan` attributes of `<disconnect>`.
- The `com.voicegenie.call.*` call-control events.
- The call-control session variables.

If these call-control features are required, you must continue to use the legacy VoiceGenie interpreter. It is highly recommended that you migrate to use CCXML, which provides all of these capabilities by using a standards-based

markup language.

Note: The `<send>` and `<receive>` elements are supported in NGI for asynchronous messaging.

endbeep Attribute

The `endbeep` attribute requests that a beep is played at the end of prompts, just before recognition or recording begins. In the legacy VoiceGenie interpreter, if an `endbeep` is requested, and the prompts have `bargein` enabled, the beep is not played. In NGI, the beep is played unless the prompt has been interrupted due to a bargein.

Transfer Events

The `error.connection.nolicense` event is replaced by the `error.connection.noresource.nolicense` event. This event is generated when the platform can not establish the outbound call due to a lack of licenses.

VoiceXML Properties

This section describes the VoiceXML properties that are used by the interpreters.

Initial Setup

In the legacy VoiceGenie interpreter, the properties that are specified in the defaults.vxml page are set before the application starts. This means that the extensions that operate, based on property values, could behave differently in NGI. In particular, these extensions include metrics and savetmpfiles properties that are used before the main page is initially loaded or started.

Property Prefixes

In the legacy VoiceGenie interpreter, several VoiceGenie extensions have been implemented, some of which do not require a company domain prefix. In NGI, all of the extension properties require the company domain as the prefix. The following extension properties do not require a prefix in the legacy VoiceGenie interpreter:

- `asrengine`
- `ttsengine`
- `endbeep`
- `loglevel`
- `metricslevel`

- savetmpfiles

Note: In NGI, the extension prefix is configurable for all of the extension properties.

Therefore, suppose that you use the following property for the legacy VoiceGenie interpreter:

```
<property name="ASRENGINE" value="REALSPEAK"/>
```

For NGI, this property must be changed to:

```
<property name="com.genesyslab.asrengine" value="REALSPEAK"/>
```

Properties Removed

The following properties are no longer supported in NGI:

- asrinittimeout
- bargeinlevel
- vgasrcalllog
- vgasrconfidencialutterance
- com.voicegenie.asr.nuance.native
- com.voicegenie.asrrawambresult
- com.voicegenie.wakeupwordranges
- com.voicegenie.serverselect
- com.voicegenie.fieldobject
- com.voicegenie.xmlencoding
- com.voicegenie.saveutterance

expr Attribute of <property>

In the legacy VoiceGenie interpreter, the following properties support the `expr` attribute, so that the property values can be evaluated at runtime:

- fetchaudio
- timeout

This capability is not supported in NGI.

asrinittimeout Property

In the legacy VoiceGenie interpreter, the `asrinittimeout <property>` is specified as a time designation, which the media platform uses in a formula to calculate the grammar loading timeout. The media platform uses the smaller value of the `timeout <property>` and the `asrinititmeout <property>` to timeout a grammar loading request that is taking too long.

In NGI, the `com.genesyslab.asr.loadgrammartimeout <property>` is used for this purpose, and this property takes an integer, in milliseconds, as its value.

The `timeout <property>` is not involved in the calcaulation of the grammar-loading timeout.

bargein Property

In the legacy VoiceGenie interpreter, the application can set the value of the bargein property to dtmf. When this value is set, only DTMF input stops the prompt, whereas a start-of-speech that is detected by the speech recognizer does not. NGI does not support this feature.

Note: In NGI, VCR controls can be enabled even though `bargein` is set to `true`. For example, pressing `5` will pause audio that is playing.

bargeintype Property

In the legacy VoiceGenie interpreter, in addition to the `hotword` and `speech` standard values, the `bargeintype` property allows the `recognition` and `energy` values. These values were included in previous drafts of VoiceXML 2.0, but they have been dropped from the W3C Recommendation version of VoiceXML 2.0. As a result, if the `bargeintype` property is to these unsupported values an `error.semantic event` is generated when the property is used.

The `bargeintype` property is used to instruct the speech/DTMF recognizers about when they should perform hotword bargein. In the legacy VoiceGenie interpreter, the `bargeintype` for a particular recognition session is determined by the property that was in effect when the last prompt in the prompt queue was added. In NGI, the recognition mode that is used for a `<field>` or `<initial>` element is the one that is specified by the `bargeintype` property that was in cope during the execution of the input item. For the `<transfer>` and `<record>` elements, the recognition mode is always `hotword`. As a result, the `bargeintype` property of a `<prompt>` element is ignored.

fetchtimeout Property

If the application specifies a negative value for the `fetchtimeout` property, the legacy VoiceGenie interpreter ignores the negative value and uses a configured default value. NGI generates an `error.semantic` event when it uses this property.

nputmodes Property

In the legacy VoiceGenie interpreter, the value of the `inputmodes` property can be set to `both`, as shorthand for `dtmf voice`. In NGI, this value is not supported, and the application must use a value of `dtmf voice` instead. Invalid values generate an `error.semantic messages`.

Additionally, if the value of `inputmodes` is `voice`, DTMF input is ignored in NGI. In the legacy VoiceGenie interpreter, DTMF input triggers a nomatch message.

recordutterance Property

The `recordutterance` property is introduced in VoiceXML 2.1. In the legacy VoiceGenie interpreter, this property can be used only when the `<vxml>` version is set to `2.1` (otherwise, it is ignored). In NGI, this property can be used even when the `<vxml>` version is not set to `2.1`.

session.com.voicegenie.transfer.allowed Variable

According to the VoiceXML specification, all session variables must be read-only. Therefore, in NGI, this function is replaced by the ECMAScript functions `session.com.genesyslab.setTransferAllowed()` and `session.com.genesyslab.getTransferAllowed()`.

session.connection.answeredby Variable

In the legacy VoiceGenie interpreter, if the VoiceXML session is associated with an outbound call, this session variable in a VoiceXML session is used to indicate what type of entity answered the call. It takes one of the following values: `human`, `fax`, `machine`, or `modem`.

This variable is not supported in NGI.

VG Properties

This section describes the VoiceGenie properties.

com.voicegenie.maxrecordtime Property

In the legacy VoiceGenie interpreter, this property can be specified only in the platform defaults file. In NGI, this property is called `com.genesyslab.maxrecordtime`, and the normal rules of property scoping apply.

Note: No platform default or maximum can be specified.

com.voicegenie.disablerecord Property

This property is not supported in NGI.

Miscellaneous Differences

This section describes other differences between NGI and the legacy VoiceGenie interpreter.

CDATA Block in SSML

In the legacy VoiceGenie interpreter, markup that is contained within a `CDATA` block that is intended for Text-to-Speech (TTS) is not escaped. It is used exactly as it appears within the VoiceXML page.

This means that the following prompt will *not* generate `<mark>` elements, but will begin speaking “mark name equals beg slash”:

```
<prompt>
   <![CDATA[
      <mark name = " beg "/>
      12
      <mark name = " xxx "/>
      Go from
      <mark name ="here"/>
      hear to
      <mark name="there and here"/>
      <mark name="end"/>
      there!
      <mark name="endend"/>
   ]]>
</prompt>
```

Form Item Properties

All form item properties and shadow variable properties that belong to a dialog scope are removed when the corresponding `<form>` element has finished processing.

DTMF Recognition

This section describes how DTMF recognition works with the interpreters.

New Implementation

A new DTMF recognizer is now integrated into the media platform. This DTMF recognizer supports only the standard Speech Recognition Grammar Specification (SRGS) grammars (both XML and Augmented Backus-Naur Form (ABNF) grammars. See `http://www.w3.org/TR/speech-grammar/`). Furthermore, it supports only those semantic interpretations that conforms to

the Speaker-Independent Speech Recognition System (SISR) 1.0 Candidate Recommendation (see `http://www.w3.org/TR/semantic-interpretation/`).

Changes in semantic interpretation might present a problem for application upgrades; therefore, as an aid to the developer, Genesys attempts to support some simple ABNF grammars, without requiring the full ABNF declaration.

The inline DTMF grammars are prepended by `#ABNF 1.0;$a =` and appended by a semi-colon (`;`) if an inline DTMF grammar meets all of the following conditions:

- It does not provide a grammar type, or if it provides a type of `application/x-abnf`.
- It does not start with the required SRGS ABNF declaration of `#ABNF 1.0;`.
- The `vxmli.legacy.simple_dtmf_grammars` configurable option value is set to `true`.

This enables automatic processing of the most common shorthand grammar formats. However, existing application grammars should be migrated to the formal SRGS XML or ABNF formats.

Note: Extended timing properties are preserved in the new DTMF recognizer, including critical digit timeout.

Hotword Recognition

In the legacy VoiceGenie interpreter, hotword `bargein` is supported for speech recognition only. In NGI, both speech and DTMF recognition support hotword `bargein`.

Error Handling

For DTMF grammars that contain content that is not in `[0-9abcd*#]` format, NGI generates an `error.badfetch` message. The legacy VoiceGenie interpreter generates an `error.grammar.dtmf` message.

Grammars

This section describes the differences between the legacy VoiceGenie interpreter and NGI grammars.

DTMF Interpretation

In NGI, the DTMF recognition results results are formatted differently than the results from the legacy VoiceGenie interpreter. For example, by using the

following grammar, a result of "12345" in the legacy VoiceGenie interpreter, whereas in NGI, the result is "1 2 3 4 5":

```
<grammar xmlns="http://www.w3.org/2001/06/grammar" mode="dtmf"
version="1.0" root="my_root2" type="application/srgs+xml">
     <rule id="my_root2" scope="public">
     <one-of>
        <item>12345</item>
     </one-of>
   </rule>
</grammar>
```

External Grammars

This section describes how the interpreters handles external grammars.

Error Handling

In the legacy VoiceGenie interpreter, the `error.asr.grammar` message could be generated in the following situations:

- Fetching the grammar failed due to a timeout.
- The grammar does not exist.
- A referenced rule is not defined.
- A grammar syntax error is detected.

In NGI, the `error.badfetch message` is generated, as specified in the VoiceXML specification.

External Grammar Fetching

The legacy VoiceGenie interpreter includes a configuration option that instructs the interpreter to fetch external grammars, and to pass the local address of these grammars to the speech engine. This capability is not supported in NGI.

All currently supported speech engines are fully capable of directly fetching and caching external grammars, as is required by the SRGS.

Inline Grammars

One consequence of NGI's strict parsing, is that it rejects certain malformed inline XML grammars that are accepted by the legacy VoiceGenie interpreter.

In particular, some of the elements or attributes in older versions of the SRGS standards (such as the `<count>` element and the `tag` attribute) are no longer supported.

To avoid this issue, ensure that inline grammars are properly formed. Also ensure that grammars are compliant with the current release of the SRGS.

Attributes

To improve conformance with the VoiceXML specification, NGI requires that inline grammars specify the `type` attribute.

Metrics and Logging Changes

Every effort has been made to replicate the format of the legacy VoiceGenie interpreter's metrics file as closely as possible in the NGI. Nevertheless, there are several differences between the two interpreters. For example, there is not guarantee that metrics will be logged in the same sequence. Additionally, the formatted output might changed.

Note: Consider these changes for external applications that perform metrics file processing. If you currently rely on the parsing of the metrics file data for report or billing purposes, your parsing application might need to be updated.

Tool-Related INFO Messages

This section describes `INFO` messages.

Script Operations

In NGI, the Script Operations `INFO` message is logged when:

- Executing a `<field>` element.
- Executing a `<transfer>` element.
- Executing a `<record>` element.
- Ending a call element.

Compile Time

In NGI, the Compile Time `INFO` message is logged immediately after a page is compiled. A message is not logged if the page was cached. Additionally, for the first pages of a session (defaults, main page, and root page), the message is logged prior to initialization, therefore, the value of the `com.voicegenie.metricslevel` option that is used at that time is specified in the

configuration through the Service Management Center (SMC) or the `callmgr.cfg` file.

Metrics

In NGI, the level of a metric is configurable through the SMC configuration. This differs from the legacy VoiceGenie interpreter, where each metric has an assigned level that cannot be changed. The following default values are the same as those found in the legacy VoiceGenie interpreter. Changing these values might disable some of the functionality of the VoiceGenie Tools, such as the Quality Advisor. Table 15 describes these metrics.

Table 15: NGI Metrics

Metric	Description
`appl_begin`	Logged before the first page is executed. **Note:** The properties that are currently in effect are the properties that are specified in the interpreter configuration, *not* in the `defautl.xml` file.
`appl_end`	Logged after all pages have been executed. Note: The properties that are currently in effect are the properties that are specified in the interpreter configuration.

`asr_trace`	The format of the result string differs. NGI provides the full raw result that is returned from the ASR engine.
	Examples:
	`asr_trace ASR_DONE:results:<?xml version='1.0'?><result><interpretation grammar="session:0x0026" confidence="90"><input mode="speech">tea</input><instance/></interpretation></result>`
	
	`asr_trace ASR_DONE:results:<?xml version='1.0'?><result><interpretation grammar="session:0x007b6" confidence="97"><input mode="speech">No</input><instance><SWI_meaning>{SWI_literal:No}</SWI_meaning></instance></interpretation></result>`

Table 15: NGI Metrics (Continued)

Metric	Description
call_appl	Logged before the first page is executed. **Note:** The properties that are currently in effect are the proterites that are specified in the interpreter configuration, not in the default-ng.vxml file.
dtmf	The dtmf parameter action is no longer specified. Examples: • dtmf :3 • dtmf :99
dtmf_end	The dtmf_end metric is no longer recorded.
exec_error	The legacy format is not precisely followed. Instead, a free-form message is logged. Example: exec_error TypeError: session.transfer has no properties
exec_warning	The exec_warning metric is no longer recorded.
form_exit	NGI logs normal as a reason; it never logs internal_error.
input_end	The input_end metric now contains more information. The format is: input_end [reason]|[mode]|[grammar_scope]|[grammar_url]|[phrase]|[confidence] Where reason can be one of the following: DISCONNECTED, FAILED, NO_INPUT, ERROR, NO_MATCH, MAX_SPEECH_TIMEOUT, ASR_MAXSPEECHTIMEOUT, RECORD_END, or TRANSFER_END. Examples: input_end MATCHED|dtmf|Field|inline|991058|1.000000 input_end NO_INPUT||||| input_end MATCHED|dtmf|Field|file:///usr/local/phoneweb/samples/testapp/test. vxml|2|1.000000
log	The format for the log metric has changed to the following: log <label>:<message><expression>

Table 15: NGI Metrics (Continued)

Metric	Description
parse_error	In NGI, the application name and line number are always empty. Example: parse_error (http://138.120.72.51/testapp/test3.vxml,, line):Invalid child element Block in element Catch at line 15
parse_warning	In NGI, the application name and line number are always empty. Example: parse_warning (http://10.0.0.1/property_vgaudiostop.vxml,,):Unknown property name: VGSTOP
prompt	The prompt metric has no data. It provides an indication that one or more prompt will be played. Each individual prompt is identified from the prompt_play metric.
prompt_play	The prompt_play metric has the same data as the original prompt metric, but for only one item of a prompt queue. Also, the filename element is no longer supported. The new format is: <type>|<data> Examples: prompt_play audio|http://127.0.0.1/audio/goodbye.vox prompt_play tts|<?xml version="1.0" encoding="UTF-8"?> <speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">Please press a number between 1 and 5. </speak>
prompt_end	In NGI, the input element is always empty, and the hangup element is no longer offered as a value for reason. Examples: prompt_end done prompt_end dtmfbargein prompt_end aborted

Table 15: NGI Metrics (Continued)

Metric	Description
record_end	In NGI, there are two changes: • A local grammar match is not logged as an outcome message; the RECORD SUCCESS message is logged instead. • The MSG_INTERRUPT message is not logged as a term reason message. Example: record_end :RECORD SUCCESS|MAXTIME|2000|audio/basic|/usr/local/phoneweb/tmp/00020069-100000A9/3262_2884936863.00020069-100000A9.ulaw
subdialog_return	The format of the subdialog_return metric depends on the return type, and it has changed in NGI: • For the event message, the format is: event|<event name> • For the namelist message, the format is: namelist|<the evaluated namelist>
subdialog_start	The param_name and param values are now listed as one entry and are encoded in JSON (see http://json.org/).
submit	The namelist value is now encoded in JSON (see http://json.org/).

Genesys

Supplements

Related Documentation Resources

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

Management Framework

- *Framework 8.0 Deployment Guide,* which provides information about configuring, installing, starting, and stopping Framework components.
- *Framework 8.0 Genesys Administrator Deployment Guide,* which provides information about installing and configuring Genesys Administrator.
- *Framework 8.0 Genesys Administrator Help,* which provides information about configuring and provisioning contact center objects by using the Genesys Administrator.
- *Framework 8.0 Configuration Options Reference Manual,* which provides descriptions of the configuration options for Framework components.

SIP Server

- *Framework 8.0 SIP Server Deployment Guide,* which provides information about configuring and installing SIP Server.

Genesys Voice Platform

- *Genesys Voice Platform 8.1 Deployment Guide,* which provides information about installing and configuring Genesys Voice Platform (GVP).
- *Genesys Voice Platform 8.1 User's Guide,* which provides information about configuring, provisioning, and monitoring GVP and its components.
- *Genesys Voice Platform 8.1 Troubleshooting Guide,* which provides troubleshooting methodology, basic troubleshooting information, and troubleshooting tools.

- *Genesys Voice Platform 8.1 SNMP and MIB Reference,* which provides information about all of the Simple Network Management Protocol (SNMP) Management Information Bases (MIBs) and traps for GVP, including descriptions and user actions.
- *Genesys Voice Platform 8.1 Genesys VoiceXML 2.1 Reference Help,* which provides information about developing Voice Extensible Markup Language (VoiceXML) applications. It presents VoiceXML concepts, and provides examples that focus on the GVP Next Generation Interpreter (NGI) implementation of VoiceXML.
- *Genesys Voice Platform 8.1 Legacy Genesys VoiceXML 2.1 Reference Manual*, which describes the VoiceXML 2.1 language as implemented by the Legacy GVP Interpreter (GVPi) in GVP 7.6 and earlier, and which is now supported in the GVP 8.1 release.
- *Genesys Voice Platform 8.1 CCXML Reference Manual,* which provides information about developing Call Control Extensible Markup Language (CCXML) applications for GVP.
- *Genesys Voice Platform 8.1 Configuration Options Reference,* which replicates the metadata available in the Genesys provisioning GUI, to provide information about all the GVP configuration options, including descriptions, syntax, valid values, and default values.
- *Genesys Voice Platform 8.1 Metrics Reference,* which provides information about all the GVP metrics (VoiceXML and CCXML application event logs), including descriptions, format, logging level, source component, and metric ID.

Genesys Voice Platform 8.1 Web Services API wiki, which describes the Web Services API that the Reporting Server supports.

Voice Platform Solution

- *Voice Platform Solution 8.1 Integration Guide,* which provides information about integrating GVP, SIP Server, and, if applicable, IVR Server.

Composer Voice

- *Composer 8.0 Deployment Guide,* which provides installation and configuration instructions for Composer.
- *Composer 8.0.3 Help,* which provides online information about using Composer, an Integrated Development Environment used to develop applications for GVP and Universal Routing.

Open Standards

- *W3C Voice Extensible Markup Language (VoiceXML) 2.1, W3C Recommendation 19 June 2007,* which is the World Wide Web Consortium (W3C) VoiceXML specification that GVP NGI supports.
- *W3C Voice Extensible Markup Language (VoiceXML) 2.0, W3C Recommendation 16 March 2004,* which is the W3C VoiceXML specification that GVP supports.
- *W3C Speech Synthesis Markup Language (SSML) Version 1.0, Recommendation 7 September 2004,* which is the W3C SSML specification that GVP supports.
- *W3C Voice Browser Call Control: CCXML Version 1.0, W3C Working Draft 29 June 2005,* which is the W3C CCXML specification that GVP supports.
- *W3C Semantic Interpretation for Speech Recognition (SISR) Version 1.0, W3C Recommendation 5 April 2007,* which is the W3C SISR specification that GVP supports.
- *W3C Speech Recognition Grammar Specification (SRGS) Version 1.0, W3C Recommendation 16 March 2004,* which is the W3C SRGS specification that GVP supports.

Genesys

- *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library DVD and provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms that are used in this document.
- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact Genesys Technical Support for more information.
- Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at `http://genesyslab.com/support`.

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

Genesys Supported Operating Environment Reference Guide

Genesys Supported Media Interfaces Reference Manual

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

Genesys product documentation is available on the:

- Genesys Customer Care website at http://genesyslab.com/support.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.
- Genesys Online Documentation at docs.genesyslab.com.

Document Conventions

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

Document Version Number

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

80fr_ref_06-2008_v8.0.001.00

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

Screen Captures Used in This Document

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

Type Styles

Table 16 describes and illustrates the type conventions that are used in this document.

Table 16: 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 104).	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 . . .

Table 16: Type Styles (Continued)

Type Style	Used For	Examples
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

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W