Composer Help

Working with CTI Applications

5/12/2025

Working with CTI Applications

Contents

Composer provides CTI blocks for two CTI scenarios supported by GVP:

- SIP Server (SIPS) scenario, which uses the Genesys SIP Server component to gain access to CTI functionality.
- CTI Connector (CTIC) scenario, which uses GVP's CTI Connector component to access CTI functionality provided by Genesys Framework.

These two scenarios do not provide identical capabilities and key differences are highlighted later in these topics. Composer provides four CTI blocks for accessing CTI functions. It generates VXML for each of these blocks that can work in either CTI scenario (SIPS or CTIC), and does not ask the user to choose between the SIPS or CTIC scenarios at design time. The decision to use CTIC or SIPS is made at runtime based on the X-Genesys headers received from GVP's Resource Manager. Therefore, the Composer user interface does not need to expose a Project-level preference for specifying the CTI scenario. **Note:** The CTI Connector provides different capabilities depending on the configuration in which other Genesys components like the IServer are deployed. For more details, please refer to the GVP documentation. Also see GVP Debugging Limitations.

Design Paradigms for CTI Applications

There are two design paradigms for building CTI applications with GVP in which Composer can be used:

- Standard VXML Applications
- URS-Centric Applications

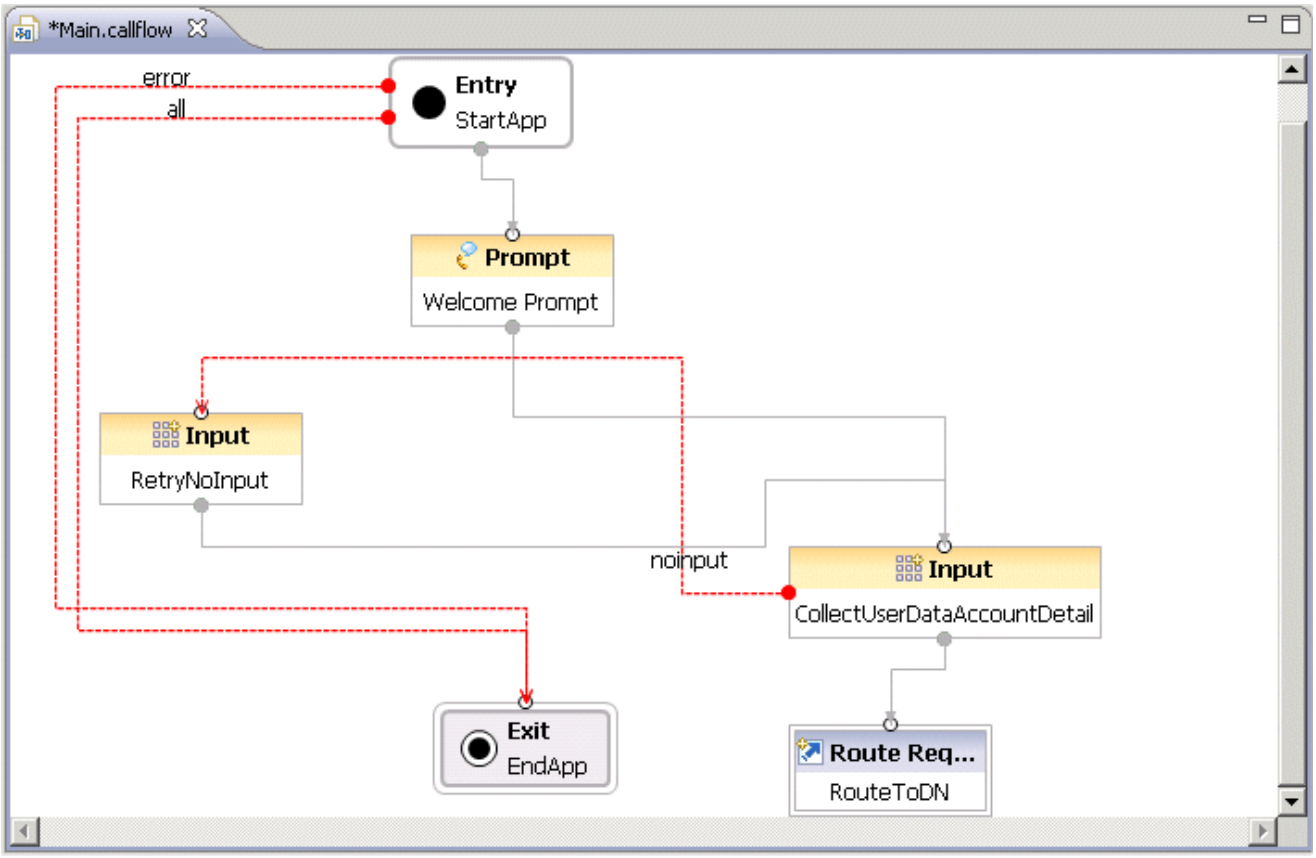
These paradigms differ in the extent to which the VXML application is involved in performing call control. **Standard VXML Applications** In this paradigm, the VXML application gets invoked first and can go through VXML interactions with the caller before using the <transfer> tag to transfer the call to another party such as queuing for an agent. At this point, the control of the call is passed to the SIP Server or CTI Connector while waiting for an agent. During this time, SIP Server or CTI Connector may invoke additional call treatments on GVP like playing music or invoking other applications. **URS-Centric Applications** In this paradigm, the VXML application is always invoked as a treatment by Genesys URS. The incoming call is controlled by Genesys URS and a strategy retains full control of the call. The strategy invokes specific treatments on GVP IVR as a media server to play prompts, play music, collect user input or execute a VXML application. In this paradigm, the VXML application does not use tags like <transfer> nor does any other kind of call control. Those decisions are left to the strategy. The VXML application returns user input collected during the call back to the strategy and lets the strategy make all call control decisions. Composer can be used to write VXML applications following either of the above paradigms.

Typical CTI Callflow

Before you start building a typical CTI application, the following information is required:

- The Genesys Virtual Route Point destination address. This is the address/location where the Genesys strategy is present (an integer number--for example, 5001).

- Strategy application on the Framework side (IRD) to find and transfers the call to an agent.



The following describes the interaction flow of this callflow:

1. GVP starts executing the generated VoiceXML application script.
2. The caller hears the Welcome prompt.
3. The caller is requested to enter the account details.
4. If the caller does not enter the required details within the maximum time frame provided, the caller is asked to retry.
5. The application issues a route request to the route DN configured in the Route Request block. (This occurs via the <transfer> tag, supported in both CTIC and SIP Server scenarios.)
6. The caller-entered data is sent as UserData to the routed DN, and the called strategy does the knowledge based transfer to the available agent based on the User Data .
7. This application ends after the Route Request has been issued.
8. The called strategy can play Voice treatments to the caller until the next available agent is available.
9. Finally, the caller will be transferred to the Agent.

Note: The Route Request block can be configured in various Transfer modes (Bridge / Consultation) to gain back the control of the callflow after the called strategy returns back the execution. Please check the Route Request topic block for more details.

CTI Scenarios

There are feature differences between the SIPS and CTIC scenarios. The following table gives a summary of the CTI blocks, and for each CTI block it lists the differences in behavior for the two CTI scenarios.

CTI Block Name	Supports CTIC Case?	Supports SIPS Case?	Comments
Interaction Data	Yes	Yes	Supported operations in each scenario: CTIC: • PUT • GET • DELETE • DELETEALL • REPLACE SIPS: • PUT • GET Types of interaction data supported: CTIC: • USERDATA SIPS: • USERDATA
Get access number	Yes	No	Get access number block can only be used in the CTIC scenario. Types of interaction data supported: CTIC: • USERDATA • EXTENSIONDATA
Statistics	Yes	No	Statistics block can only be used in the CTIC scenario.
Route Request	Yes	Yes	Types of interaction data supported: CTIC: • USERDATA

			• EXTENSIONDATA SIPS: • USERDATA Types of transfers supported: CTIC: • Blind • Bridge SIPS: • Consultation • Blind • bridge

In case a CTI block or feature is used in a CTI scenario in which it is not supported, appropriate exceptions will be thrown at runtime indicating that the feature is not supported. The table below gives a list of all exceptions that can be thrown by CTI blocks and other possible CTI-related exceptions that can be thrown if errors are encountered at runtime.

Block(s)	Exception	Error Message	Description
Interaction Data Get access number Statistics	error.com.genesyslab.composer.invalidkey	Missing <block name> key <key name>	This is the event error for handling an invalid key name.
Interaction Data Get access number Statistics Route Request	error.com.genesyslab.composer.operationtimedout	Operation timed out.	This exception will be thrown when a <receive> operation, executed in the context of a CTIC specific operation, times out.
Interaction Data Get access number Statistics Route Request	error.com.genesyslab.composer.receiveerror	<Error string returned by CTIC>	If the <receive> fails and an error is reported by CTIC, this exception will be thrown.
Interaction Data	error.com.genesyslab.composer.unsupported	Delete operation not supported in case of CTI using SIPServer.	If the user wants to do a userdata DELETE in the CTI using SIPS scenario.
Interaction Data	error.com.genesyslab.composer.unsupported	DeleteAll operation not supported in case of CTI using SIPServer.	If the user wants to do a userdata DELETEALL in the CTI using SIPS scenario.
Interaction Data	error.com.genesyslab.composer.unsupported	Replace operation not supported in case of CTI using SIPServer.	If the user wants to do a userdata REPLACE in the CTI using SIPS scenario.
Get access number	error.com.genesyslab.composer.unsupported	AccessNumGet operation not supported	If the user wants to do a AccessNumGet in the

		in case of CTI using SIPServer.	CTI using SIPS scenario.
Statistics	error.com.genesyslab.composer.unsupported.case	Statistics block not supported in case of CTI using SIPServer.	If the user wants to do a PeekStatReq or GetStatReq in the CTI using SIPS scenario.
Route Request	error.com.genesyslab.composer.unsupported.case	Consultation transfer is not supported in case of CTI using CTIConnector.	If user sets Transfer type to consultation in case of CTI using SIPS.

Script ID Usage in the GVP 8 Environment

In Genesys VoiceXML 2.1, ScriptId refers to the script identifier, as generated by the CTI Connector, to handle call treatments. The use of ScriptId is specific to GVP 7.x and was mandatory for treatments. Since the GVP 7.x design is "IVR-centric," the treatment would be invoked on the same VXML session. Things are a bit different with GVP 8.x and the Next Generation Interpreter (NGI) where APP_URI is used instead of ScriptId and the treatments are executed on different VXML sessions. **GVP 8 and NGI** In GVP 8.x, request for treatment execution comes in as a NETANN request with the APP_URI being passed in as a VoiceXML parameter. GVP executes the requested page to kick off the treatment. Unlike the GVP 7.x environment, treatments get invoked as separate VXML sessions and terminated at the end of the treatment execution. Hence, ScriptId switching is no longer needed here, unless an application wants to do branching based on ScriptId.

- Note: Composer provides support for both SIPS and CTIC scenarios for achieving the CTI functionality. However, SIPS may not support passing additional request-uri parameters like ScriptId, therefore, this option is limited only to CTIC scenarios.

Please refer to *GVP 8.x VXML Help* under Sample Voice XML Applications > CTI Interactions > Treatments for more details on this topic.

Accessing ScriptId in Composer

Use if you want your application to do ScriptId-based switching like GVP 7.x. **CTIC Scenario (IRD strategy + Composer Callflow)**

1. Use the APP_ID property in IRD's Play Application block.
2. Define a new Input type variable named ScriptId in the Entry block of your callflow to collect the ScriptId.

Composer Workflow + Composer Callflow)

1. On the VXML callflow side, define a new Input type variable named ScriptId in the Entry block to collect the APP_ID (i.e., ScriptId) passed from the workflow.
2. On the SCXML workflow side, use the Play Application block to invoke the callflow created using step#1. Then do an auto-synchronize for the parameters, and specify the ScriptId value.
3. The ScriptId (i.e., APP_ID) passed from the workflow will be automatically collected on the VXML side

from the session.connection.protocol.sip.requesturi array.

SIPS Scenario

1. SIPS may not support passing additional request-uri parameters. Pass ScriptId as attached data on the strategy side (If using IRD) or on the SCXML side (If using Composer workflows).
2. Define a new Input type variable named ScriptId in the Entry block to collect the ScriptId.
3. The ScriptId (i.e., APP_ID) passed from the strategy will be automatically collected on the VXML side from the session.com.genesyslab.userdata array.