



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

# Orchestration Server Deployment Guide

SCXML Application Development

12/19/2025

---

## Contents

- 1 SCXML Application Development
  - 1.1 Deploying Voice SCXML Applications
  - 1.2 Manually Deploying a Voice SCXML Application
  - 1.3 Debugging SCXML Applications with Composer
  - 1.4 Samples

# SCXML Application Development

You can create SCXML-based applications by using the following methods:

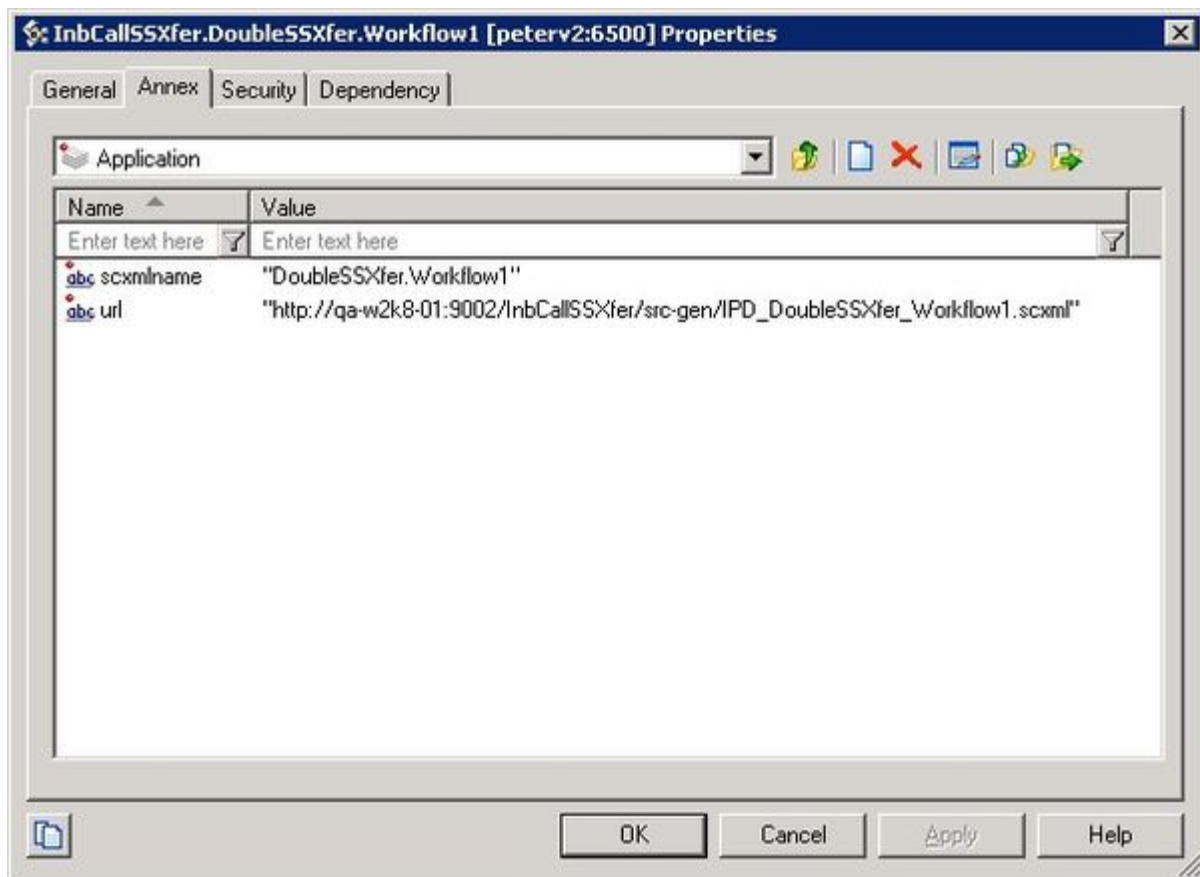
- Any simple text editor such as Notepad or an XML-based editor, with which you are already comfortable. Use the [Orchestration Server Developer's Guide](#) for the details.
- Using [Genesys Composer](#), which is an Integrated Development Environment based on Eclipse. Composer provides both drag-and-drop graphical development of voice applications (or "callflows") and routing strategies (or "workflows") as well as syntax-directed editing of these applications.

For more information on using this GUI, see the [Composer 8.1.x Help](#). Also see the [Composer 8.1.4 Deployment Guide](#).

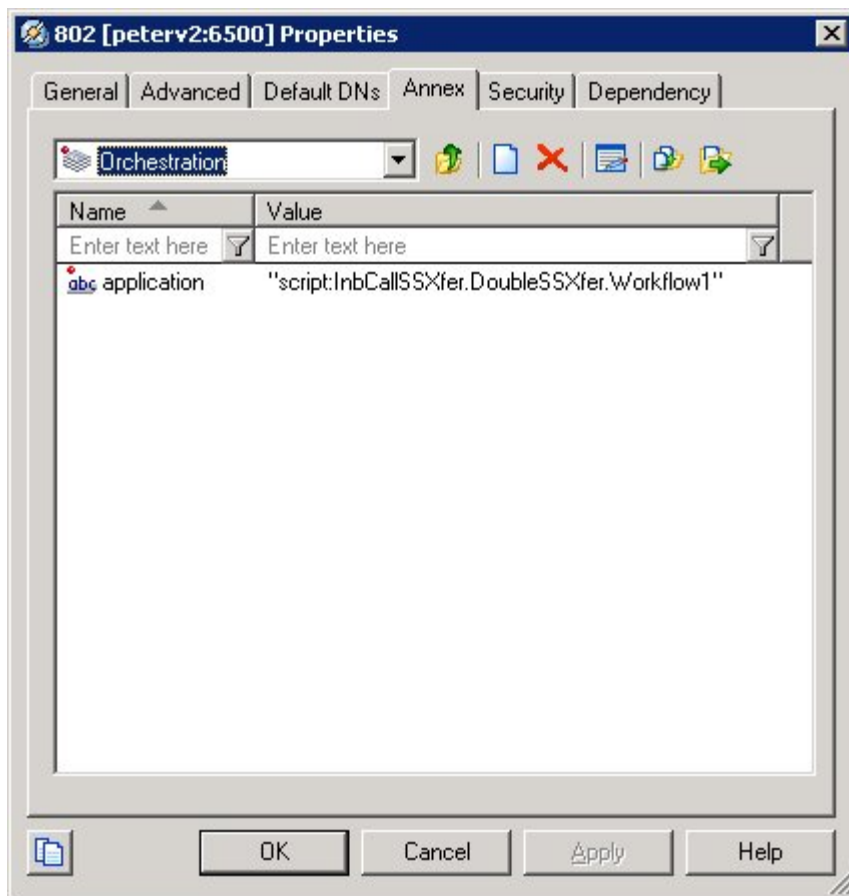
**Note:** You are not required to upgrade Composer when migrating from ORS 8.1.3 to ORS 8.1.4. ORS 8.1.4 is compatible with Composer 8.1.3. You only need to upgrade to Composer 8.1.4 if you are planning to use [enhanced pulling of multimedia interactions](#).

## Deploying Voice SCXML Applications

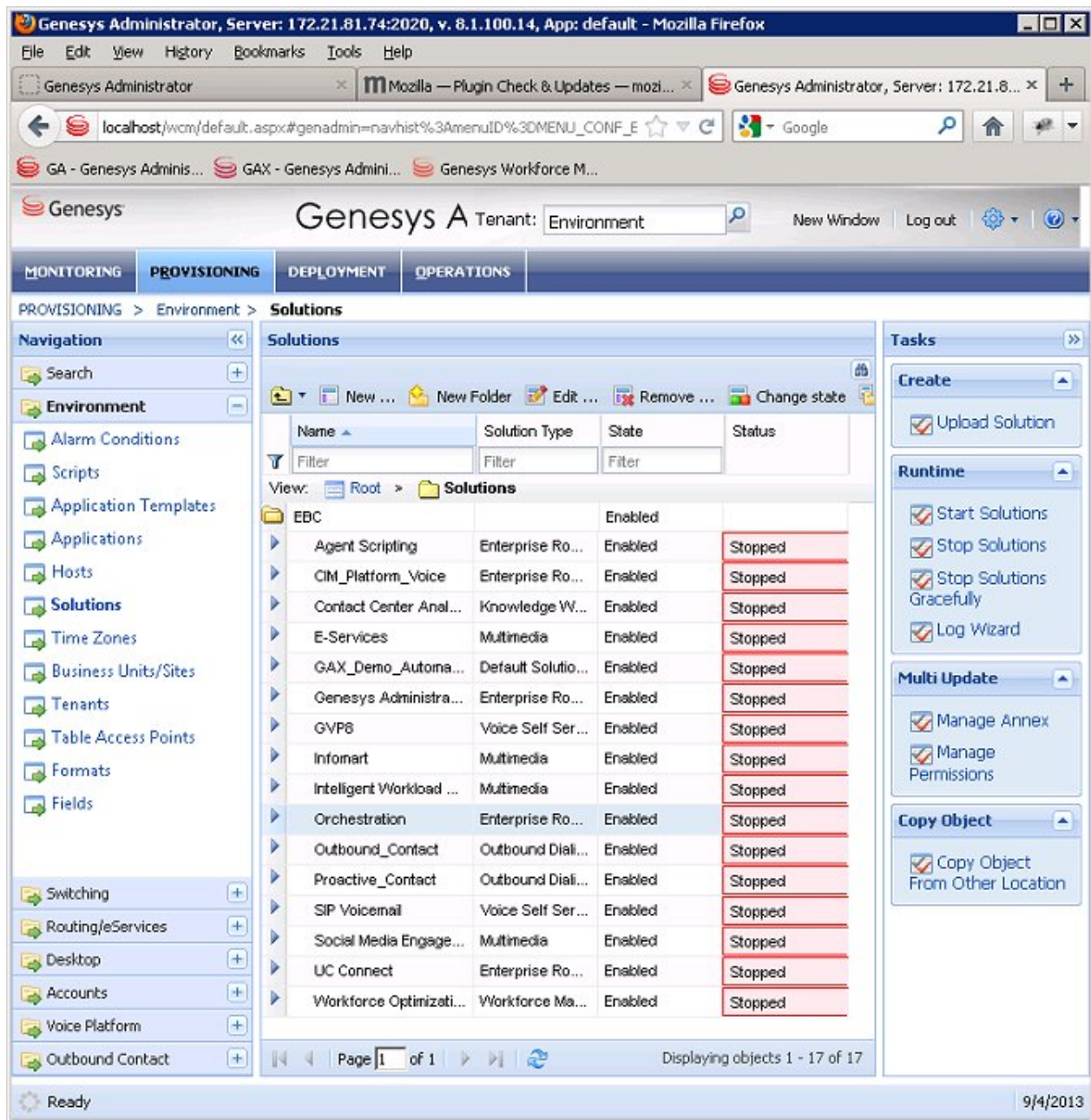
1. Within Composer, click toolbar icon to [create a new Java Composer Project](#).
2. Specify the Route type. Specifying Route indicates the development ORS/URS routing strategies.
3. Build the [interaction process diagram](#), which was created automatically in step 1 above (default name: `default.ixnprocess`).
4. In the interaction process diagram (`default.ixnprocess`), locate the [Resource](#) property and point to the workflow (`Workflows/default.workflow`), which can be developed later.
5. Within Composer, connect to Configuration Server so you can view/select objects, such as agents, places, and so on.
6. Develop your workflow diagram (routing strategy) using the [diagram building blocks](#) from the palette, such as from the Flow Control and Routing block categories.
7. Save the workflow and interaction process diagrams.
8. Validate the workflow and interaction process diagrams.
9. View any warnings or problem messages and correct.
10. Generate the code. As a result, two SCXML files are generated and can be viewed in an SCXML editor.
11. [Publish](#) the interaction process diagram to Configuration Server. An Enhanced Routing Script object will be created in the Configuration Database. The figure below shows an Enhanced Routing Script object with option [url](#) set to the location of the SCXML application.



12. Load the published application to a Routing Point. The figure below shows the Routing Point with the application option set to the name of the Enhanced Routing Script object.



- Log into Genesys Administrator.
- Navigate to Provisioning>Routing/eServices >Orchestration



- Locate the Enhanced Routing Script object.
- Click on **DNs** tab.
- Add your DN (as in the above Routing Point example).
- Load this application to this Routing Point.

13. Test that the application is successfully provisioned and deployed. Make a test call.

## Manually Deploying a Voice SCXML Application

1. Create SCXML in the editor of your choice.
2. Manually deploy your SCXML application on an application server.
3. Load deployed application to a routing point.

## Debugging SCXML Applications with Composer

The SCXML Real Time Debugging feature of **Composer** provides the possibility for developers to debug their SCXML scripts in a manner similar to what is provided by Visual Studio and Eclipse. You can examine variables and properties, as well as pause and resume the flow of execution of a script, by setting breakpoints in the code, viewing standard SCXML Engine metrics (log entries). The ORS feature works using a client-server configuration where the client is the Genesys Composer integrated development environment. The entire RTD system is designed in such a way that it stops at every possible statement, and it is the responsibility of the client to determine whether to resume execution of the script right away or to pause and give control to the developer; in other words, determine whether or not the developer has set a breakpoint to this particular statement.

Debugging can be started on an existing session or it can wait for the next session that runs the application at a given URL.

To enable Orchestration Routing's debugging functionality, configure the debugging options in the SCXML section of the ORS application. See the following options:

- debug-port
- debug-enabled
- max-debug-sessions

The debugging SCXML applications functionality is supported in Composer starting from v8.1.2. For more information, see the **Composer Deployment Guide**.

### Limitation

The `<createcall>` and `<consult>` interaction action elements have a limitation for the attribute "to" (where a destination is specified). Only digits can be specified in the "to" attribute.

The limitation can be removed by configuration of `valid-digits` and `max-digits` options on the Switch object > Annex tab> gts section:

Option: **valid-digits**

Value: An explicit set of all acceptable symbols that can be dialed.

Example: 0123456789\_RP0

Option: **max-digits**

Value: The maximum number of digits that can be dialed.  
Example: 25

## Samples

Orchestration Server supports only SCXML documents. The *Orchestration Developer's Guide* contains [sample SCXML routing applications](#). The samples are not designed for use in a Production environment. Instead, use them to get started configuring your own applications, subroutines, and list objects. Consider them as guides when developing your own objects adjusted to your company's specific business needs.