



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.

Composer Deployment Guide

Composer Interface

12/19/2025

Composer Interface

Contents

- **1 Composer Interface**
 - **1.1 Drag and Drop-Based GUI**
 - **1.2 Debugging VoiceXML Applications**
 - **1.3 Debugging Routing SCXML Applications**
 - **1.4 Other Composer Features**

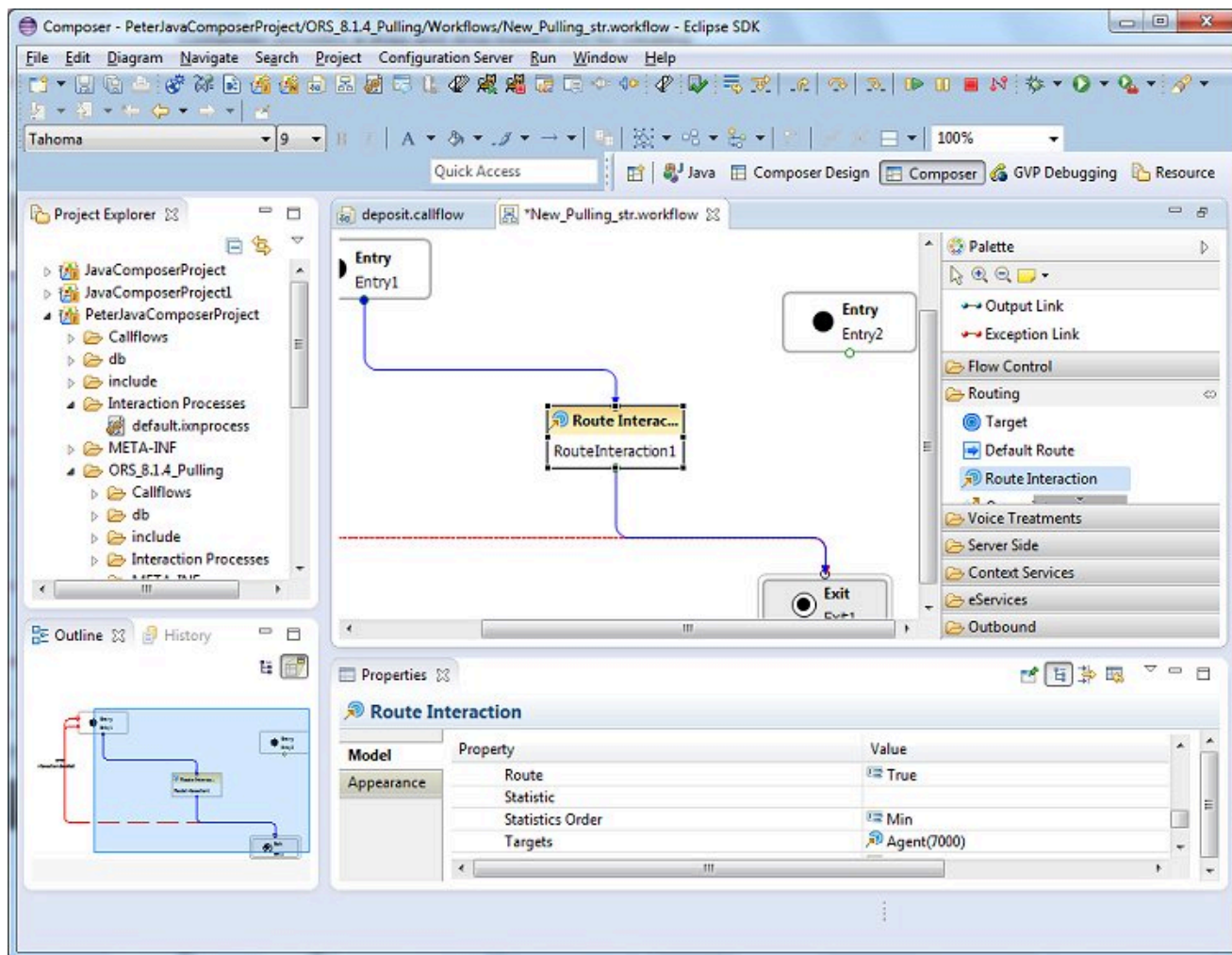
Note: This topic presents a brief overview of the Composer interface. For detailed information on using the interface, see the [Composer Help](#).

Drag and Drop-Based GUI

Composer provides a drag and drop-based GUI for creating:

- VXML callflow diagrams (for voice applications)
- SCXML workflow diagrams and interaction process diagrams (for routing applications).

Technical and non-technical developers have the option of creating flow diagrams by placing and connecting blocks and configuring properties and/or by writing code. The figure below shows an example callflow in the center editing area in Composer perspective.



The interface elements in Composer perspective are as follows:

- A **Project Explorer** view on the upper left gives access to all the Project files.
- An **Outline** view of the entire callflow or workflow on the lower left is useful when working with complex diagrams.
- The **History** view on the lower left, which maintains previous versions of flows and application files, allowing you to revert to any previous version if needed.
- A center editing area (sometimes referred to as the **canvas** where you drag, drop, and connect blocks.
- A lower view for configuring block **Properties** (fields). Buttons in property rows display dialog boxes.
- A **Palette** of blocks grouped in categories on the upper right for creating flow diagrams.

A Composer perspective can also show various views in the lower pane depending on your actions or what you select from **Window > Show View** . For example, for voice applications, the lower pane can show the following views:

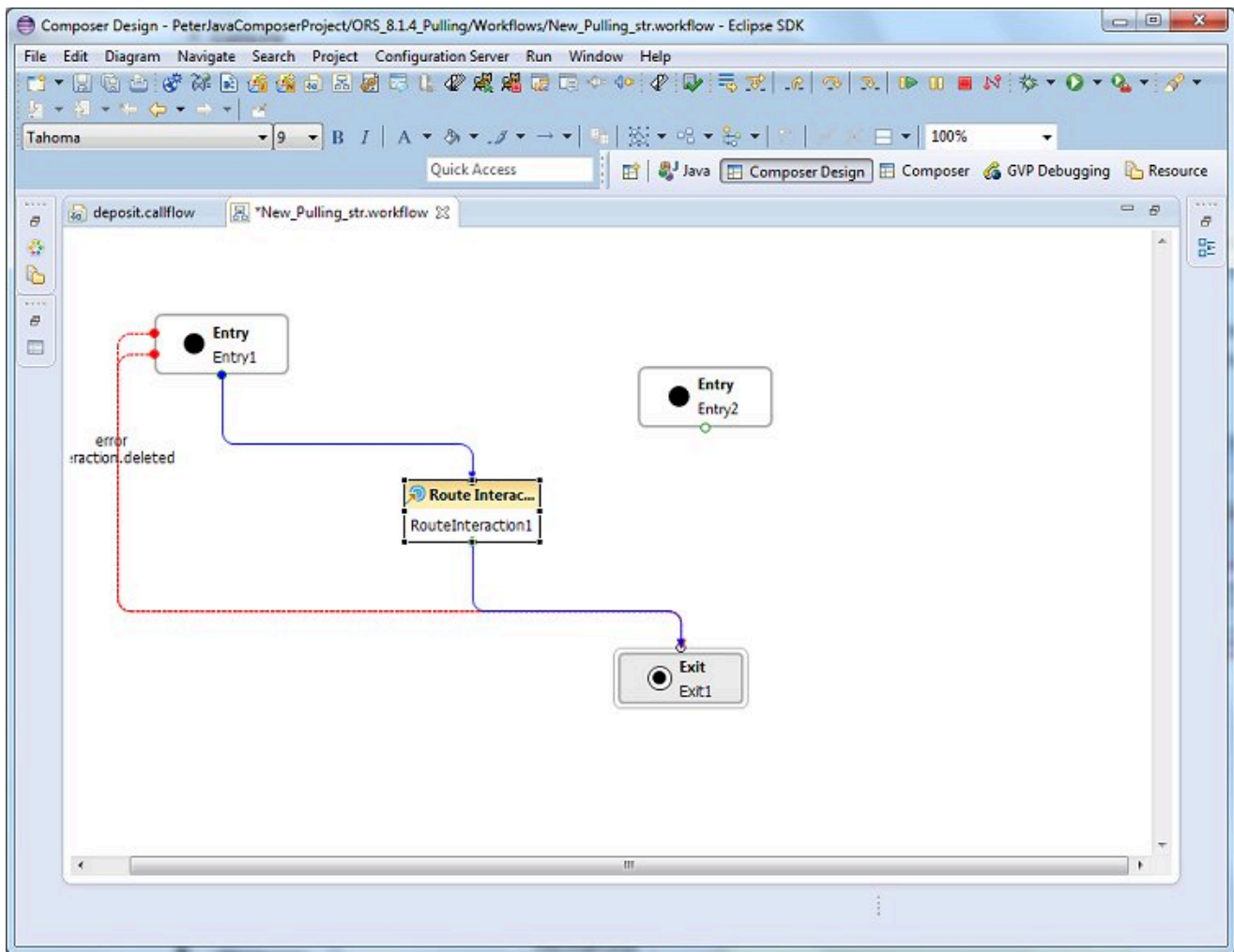
- **Properties**
- **Prompts Manager**
- **Problems**
- **Console**
- **Call Trace**

Perspectives

When working in Composer, you have the option of working in different **perspectives** .

A perspective is an arrangement of different sections of the GUI in a manner that facilitates easy use of a particular feature, such as design or debugging. For example, the **GVP Debugging** perspective will show those sections that are useful when debugging a voice application: Call Trace, Console, Variables, Breakpoints, and so on.

The figure above shows **Composer** perspective. The figure below shows **Composer Design** perspective, which maximizes the design area. Having a larger design area is useful when creating complex flow diagrams. **Composer Design** perspective shows only the palette of blocks, the canvas area, and the *Properties* view, but can be customized to include other views that you select.



For routing applications, the lower pane can show the following views:

- **Properties**
- **Problems**
- **List Objects Manager**
- **Statistics Manager**

Available Perspectives

Composer includes the following perspectives for building applications:

- **Composer** , for both voice and routing applications, shows the Project Explorer, Outline view, canvas, palette, and can show the following tabs in the lower pane: Properties, Prompts Manager, Problems, Console, and Call Trace.
- **Composer Design** , for both voice and routing applications, can be used to simplify the workbench to

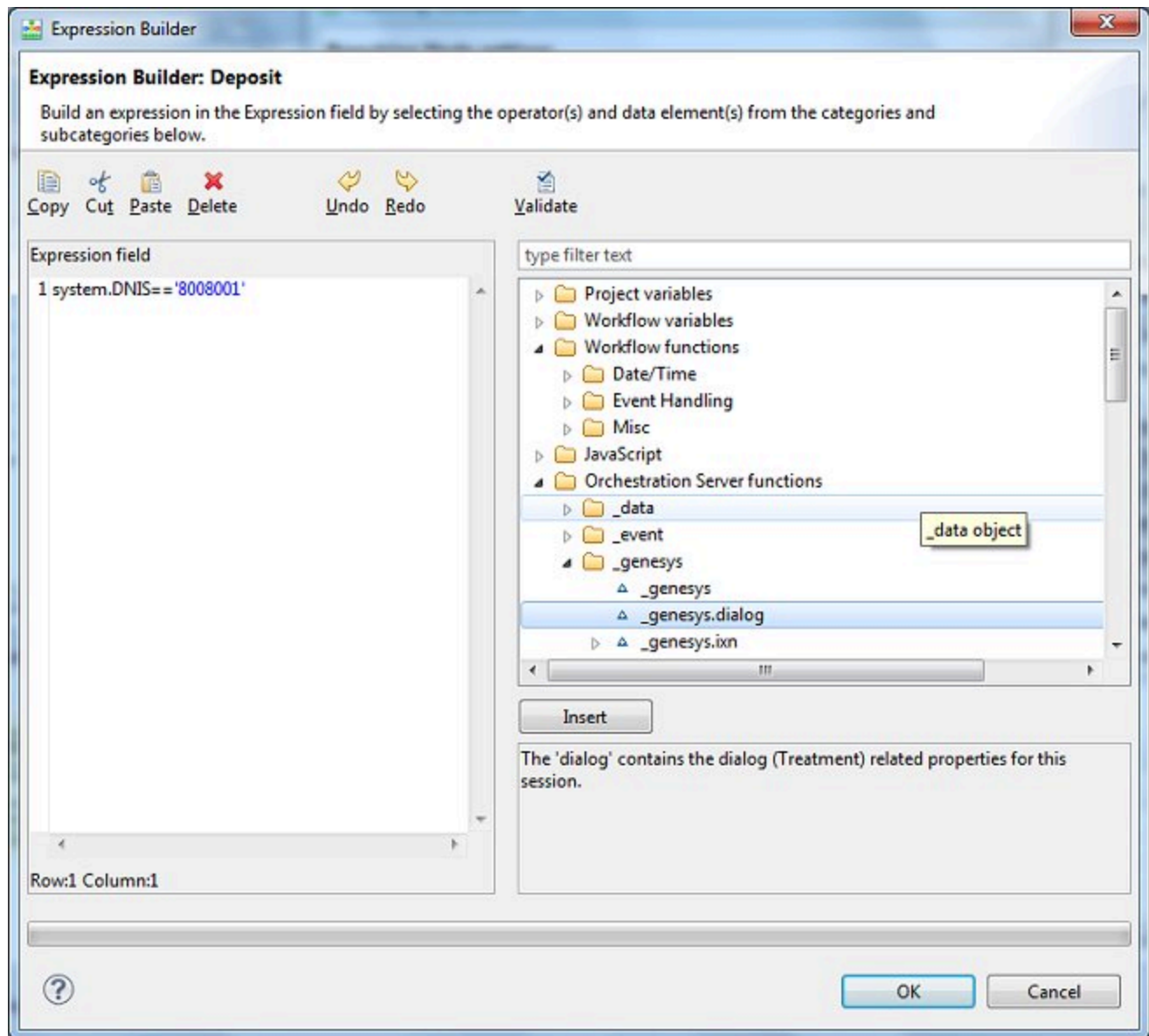
show only the palette of blocks, the canvas area, and the Properties tab.

- **GVP Debugger** , for debugging voice callflows that you build or import.
- **ORS Debugger** , for debugging routing workflows that you build or import.
- **Prompts Manager** , which provides the ability to quickly review all prompts in a Composer Project.

Expression Builder

Composer supplies Expression Builder to easily build expressions that can be used for branching and conditional routing decisions. You can also build expressions that use the Orchestration Server implementation of **SCXML** and **Orchestration Server Extensions**.

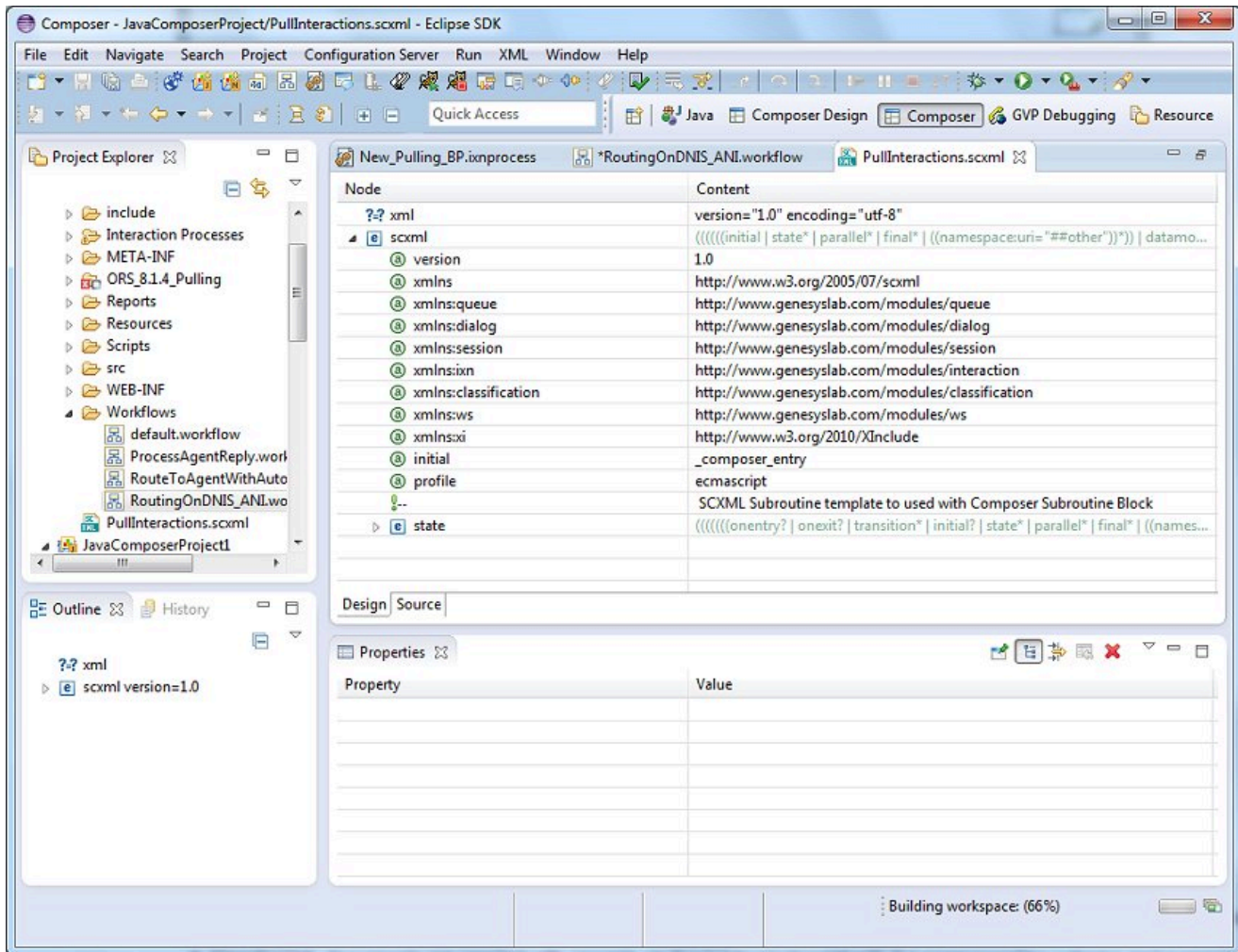
The figure below shows an example Orchestration Server function in Expression Builder.



Rich Editors

For those who prefer to write their own code, Composer provides a set of rich editors, supplying built-in error checking and tooltips, for SCXML, VXML, CCXML, and GRXML along with use case templates.

- The figure immediately below shows example SCXML code in the Design tab of the editor.



You can view and work directly with source code using standard Eclipse text editing features. Features include:

- Smart double-clicking behavior.
- Context-assisted help when typing tags. Also context-assisted help for attributes of a tag upon pressing Space inside a tag.
- New SCXML documents are created with `<scxml>` as the top level element with the corresponding schema and namespace specifications.
- Ability to edit tag attribute values from the **Properties** view.
- Basic editor actions are supported: Cut, Copy, Paste, Save, Save as, Undo, Redo, Search and Replace.
- Syntax highlighting.
- Show and hide Line numbers.
- Add/Remove Bookmark and To-Do markers.
- Task tag feature to auto scan To-Do comments in the code.

- Comparing and reverting to local file history.
- Spell checking by showing yellow squiggly line markers.
- Ability to see the outline structured view of the document in the **Outline** view.
- Validation shows errors in the *Problems* view. Validation happens based on the referenced schema.

Debugging VoiceXML Applications

Composer provides a real-time **GVP Debugger** with support for both **Run** and **Debug** modes.

- In the **Run** mode, call traces are provided and the application continues without any breakpoints.
- In the **Debug** mode, you can input breakpoints, single-step through the VoiceXML code, inspect and modify variable and property values, and execute any ECMAScript from the query console.

Integration with a SIP Phone is provided and click to dial feature is provided for making the test calls.

The Tomcat application server is bundled as part of the Composer and you can auto-deploy applications on Tomcat for testing.

Note: Composer 8.1 uses TCP to send SIP messages (previous releases used UDP). This is not a configurable option.

Debugging Routing SCXML Applications

Composer provides **real-time debugging capabilities** for Orchestration Server (ORS) routing applications. The Debugger is integrated within the workflow designer for making test calls, creating breakpoints, viewing call traces, stepping through an SCXML document/workflow, and debugging applications. Debugging can be started on an existing session or it can wait for the next session that runs the application at a given URL.

- Using a Run Configurations launch configuration, metrics (call traces) are provided and the application continues without any breakpoints. When the SCXML application executes, these metrics can describe, for example, state transitions, ECMAScript executions, and execution warnings or errors.
- Using a Debug Configurations launch configuration, you can input breakpoints, single-step through the code, inspect variable and property values, and execute any ECMAScript from the query console.

You can debug:

- A workflow built with Composer, or
- Any SCXML application or set of SCXML pages whether or not they were created with Composer.

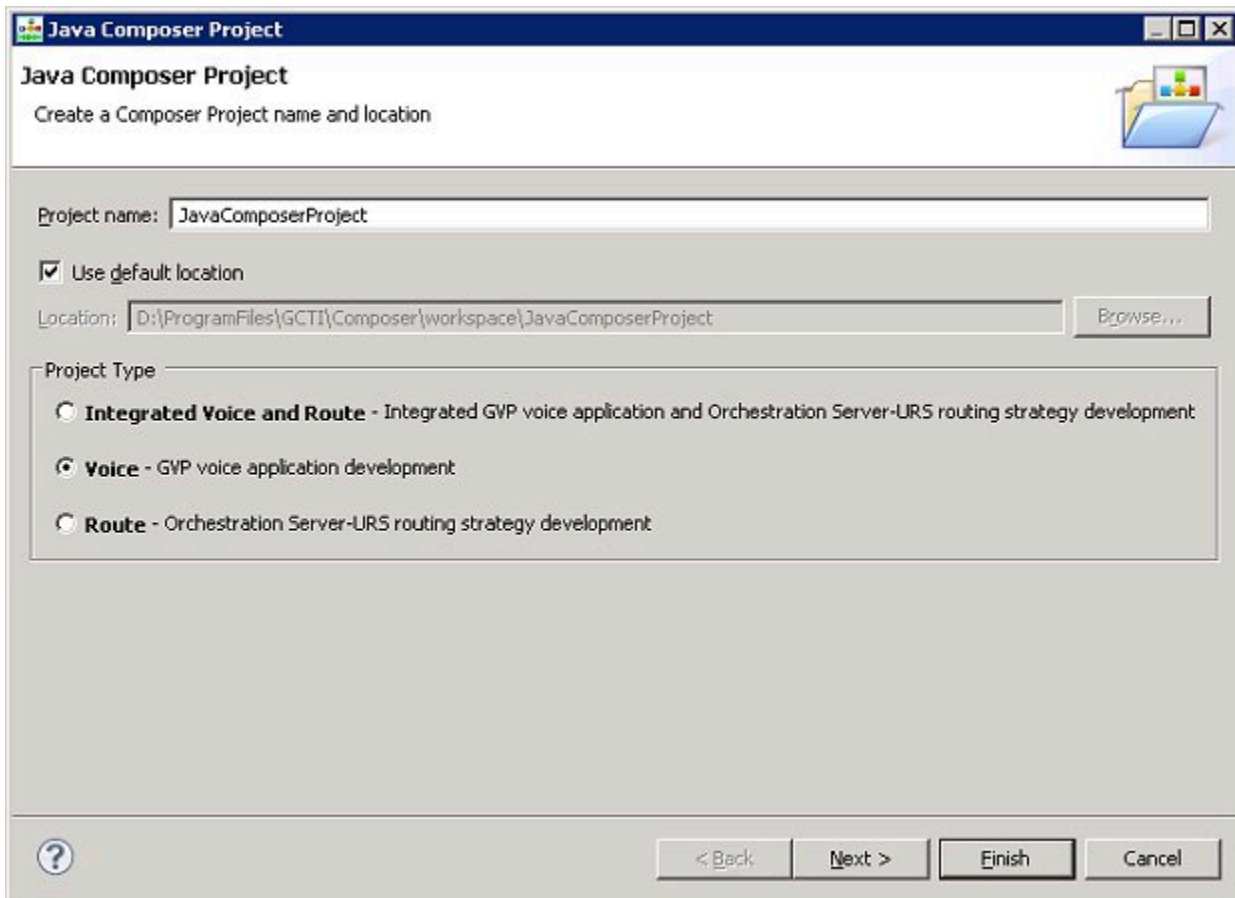
Other Composer Features

Some other Composer main features are summarized below. For details information on all Composer features, see the *Composer 8.1 Help* .

Project Templates

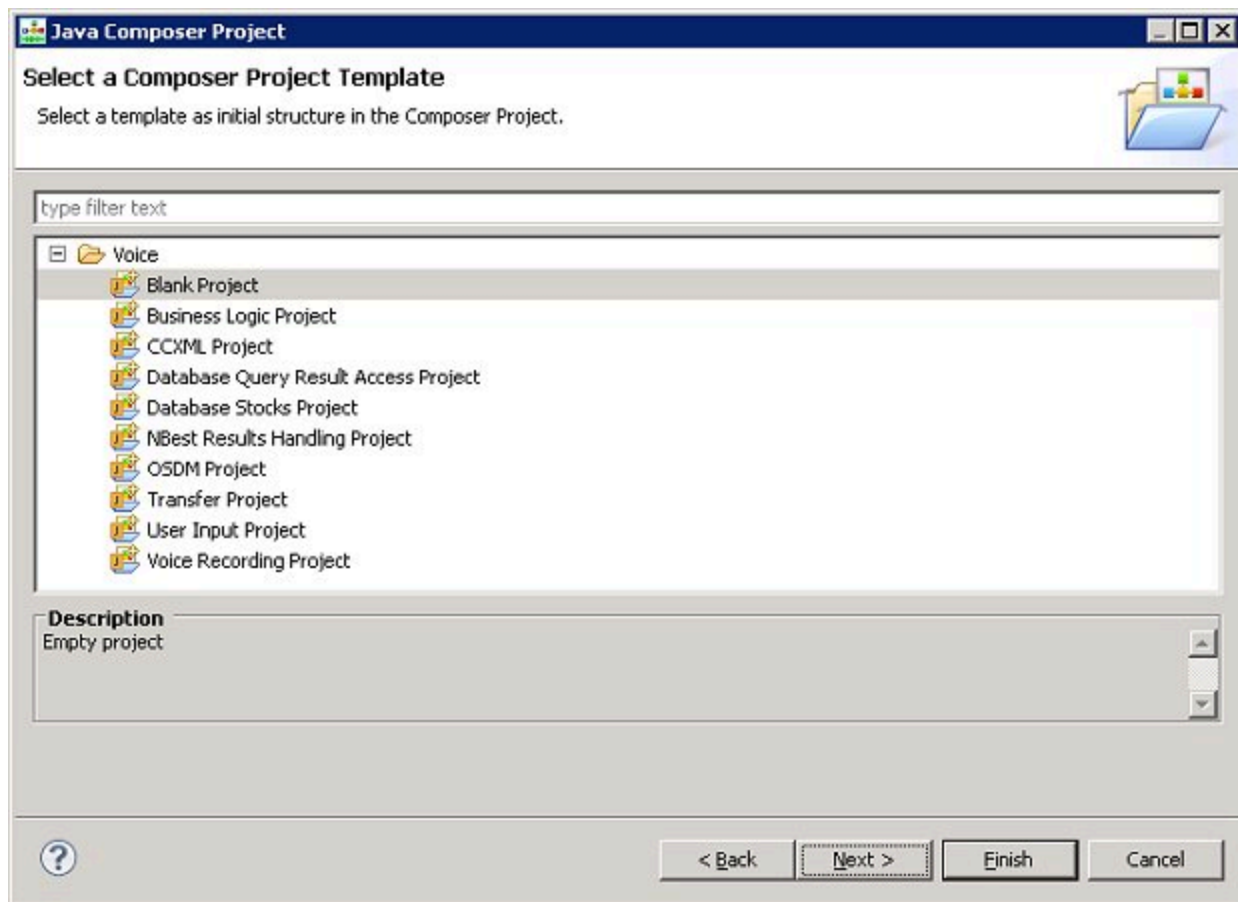
Out-of-the-box, reusable Project templates are provided. A Project wizard lets you select from three categories of templates:

1. **Integrated Voice and Route** : Select to create a Project that contains both callflows and workflows that interact with each other. For example a routing strategy that invokes a GVP voice application.
2. **Voice** : Select to create a Project associated with the GVP 8.x. This type of Project may include callflows, and related server-side files.
3. **Route** : Select to create a Project associated with the Orchestration Server 8.1 SCXML Engine/Interpreter.



The screenshot shows the 'Java Composer Project' wizard dialog box. The title bar reads 'Java Composer Project'. The main heading is 'Java Composer Project' with the subtitle 'Create a Composer Project name and location'. There is a folder icon in the top right corner. The 'Project name:' field contains 'JavaComposerProject'. The 'Use default location' checkbox is checked. The 'Location:' field shows the path 'D:\ProgramFiles\GCTI\Composer\workspace\JavaComposerProject' with a 'Browse...' button to its right. The 'Project Type' section has three radio button options: 'Integrated Voice and Route - Integrated GVP voice application and Orchestration Server-URS routing strategy development', 'Voice - GVP voice application development' (which is selected), and 'Route - Orchestration Server-URS routing strategy development'. At the bottom, there is a help icon (question mark) and four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Clicking *Next* brings up available templates for the selected category.



These can act as a starting point for new projects and visual flows and serve as guidelines and tutorials for routing and voice application developers. Composer also provides templates for its rich editors with the ability to create user-defined custom code snippet templates, which can be exported and imported to share across team members.

Code Generation

When generating code, Composer provides the ability to generate VXML pages to take advantage of the Platform optimizations. For SCXML routing strategies, Composer provides the ability to generate static SCXML pages for improved performance due to caching.

Deployment

Composer provides the ability to **deploy Java Composer Projects and .NET Composer Projects**. The deployment process involves exporting your project, transferring the files to your web/application server, and executing any necessary configuration steps required to make your application work. The Composer deployment process varies depending on the type of project being deployed (.NET Composer or Java Composer) and the associated application server. Future releases will provide the ability to deploy routing applications.

Project Management

Composer uses a Project to contain everything related to a single routing or voice application. A *Project Explorer* on the upper left of the **Composer window** contains all the Projects in your workspace. organize all the application elements.

Hiding Capabilities

Users may hide voice or routing capabilities through a Composer preference setting. This is useful for developers who are only using one of these Genesys platforms.

Builders/Managers

Composer contains several builders/managers, which are used for routing applications.

Statistics Builder/Manager

Use if you wish to use option of instructing Universal Routing Server to use the value of a statistic during target selection, such as *StatTimeInReadyState* . The statistic can be a URS Predefined statistic (as described in the *Universal Routing 8.1 Reference Manual*) or a statistic that you create yourself with **Statistics Builder**. Once you create a statistic, that statistic becomes available for selection in Composer's **Target block**.

List Objects Builder/Manager

A **List object** contains strings of any nature (for example, DNIS or ANI strings), which can be used in workflows. The strings can be as simple as 800 numbers or as complex as routing conditions. In Expression Builder, two URS Functions can be used to access List Objects: `_genesys.session.listLookupValue` and `_genesys.session.getListItemValue`.

Skill Expression Builder

Besides Expression Builder, Composer also has a Skill Expression Builder, which you can use for creating skill expressions used for routing decisions Opens from the *Targets* property in the routing Target block after selecting the *Skill* as the target type. Also opens from the Backend, Subdialog, Subroutine, Web Request, and Web Service blocks.

Customization Manager

The **Customization Manager** view helps you manage various aspects of your Composer installation that you have customized. You can manage any custom workflow and callflow diagram templates that you have created. You can also edit and delete custom templates, add new files, and save diagrams to disk.