Composer Deployment Guide

Composer GUI

12/15/2025

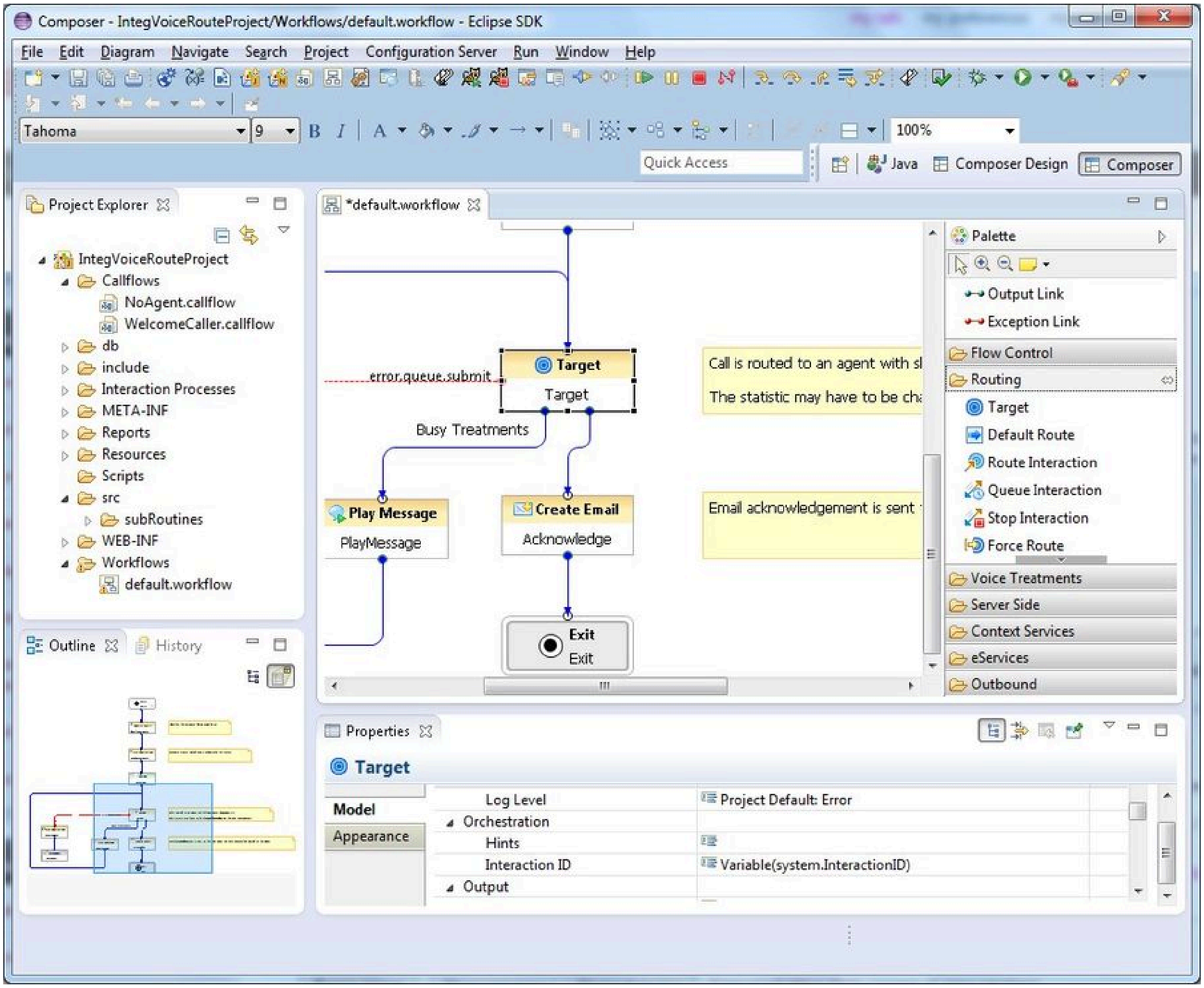
Composer GUI

Contents

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 can easily create flow diagrams by placing and connecting blocks and configuring properties. The figure below shows an example callflow in the center editing area in Composer perspective.



Call is routed to an agent with sl

The statistic may have to be cha

Email acknowledgement is sent

Log Level	Project Default: Error
Orchestration	
Hints	
Interaction ID	Variable(system.InteractionID)
Output	

The interface elements in Composer perspective are as follows:

- A **Package 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:

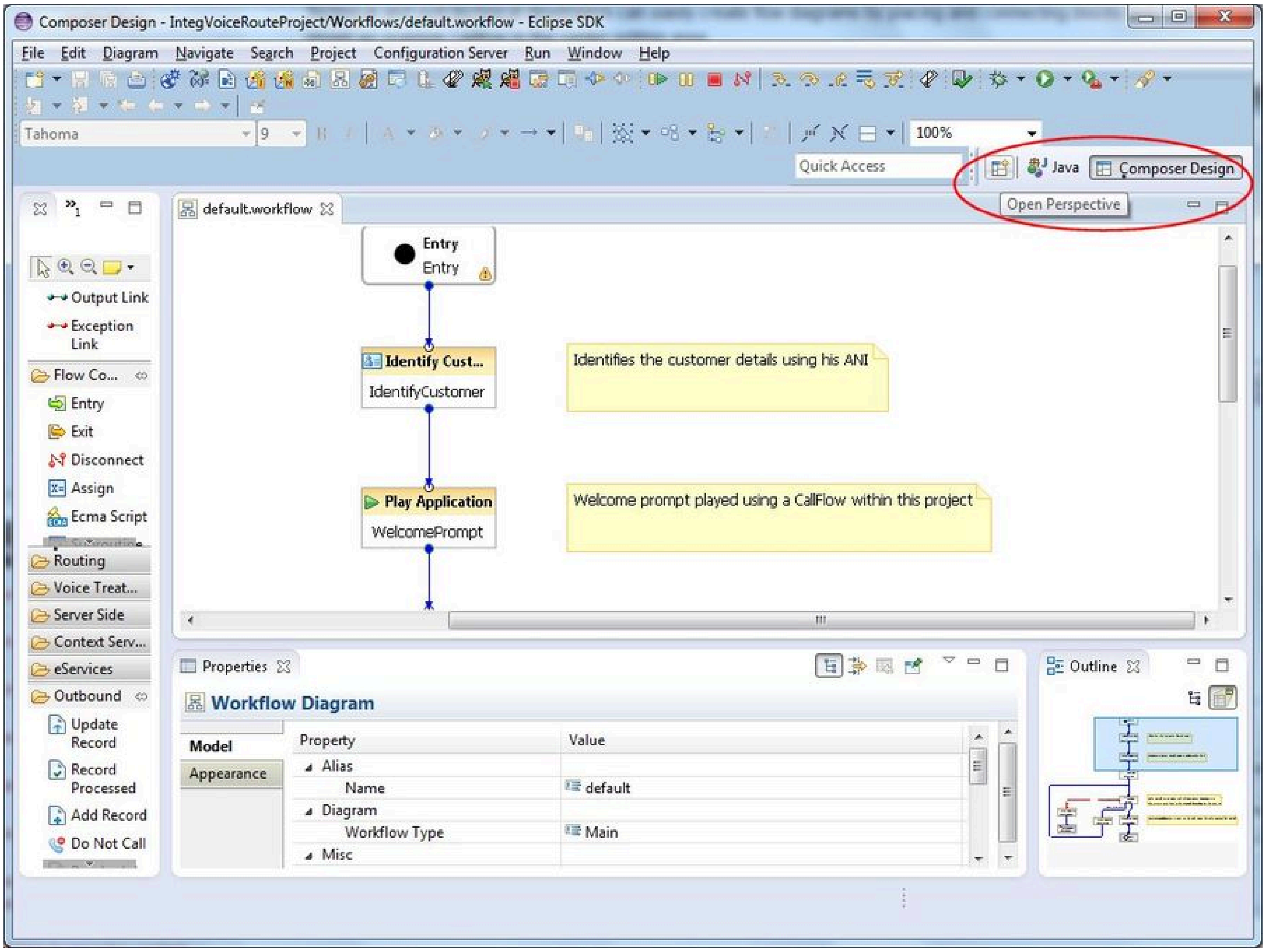
- **Properties**
- **Prompts Manager**
- **Problems**
- **Console**
- **Call Trace**

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.



Identifies the customer details using his ANI

Welcome prompt played using a CallFlow within this project

Property	Value
Alias	
Name	default
Diagram	
Workflow Type	Main
Misc	

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:

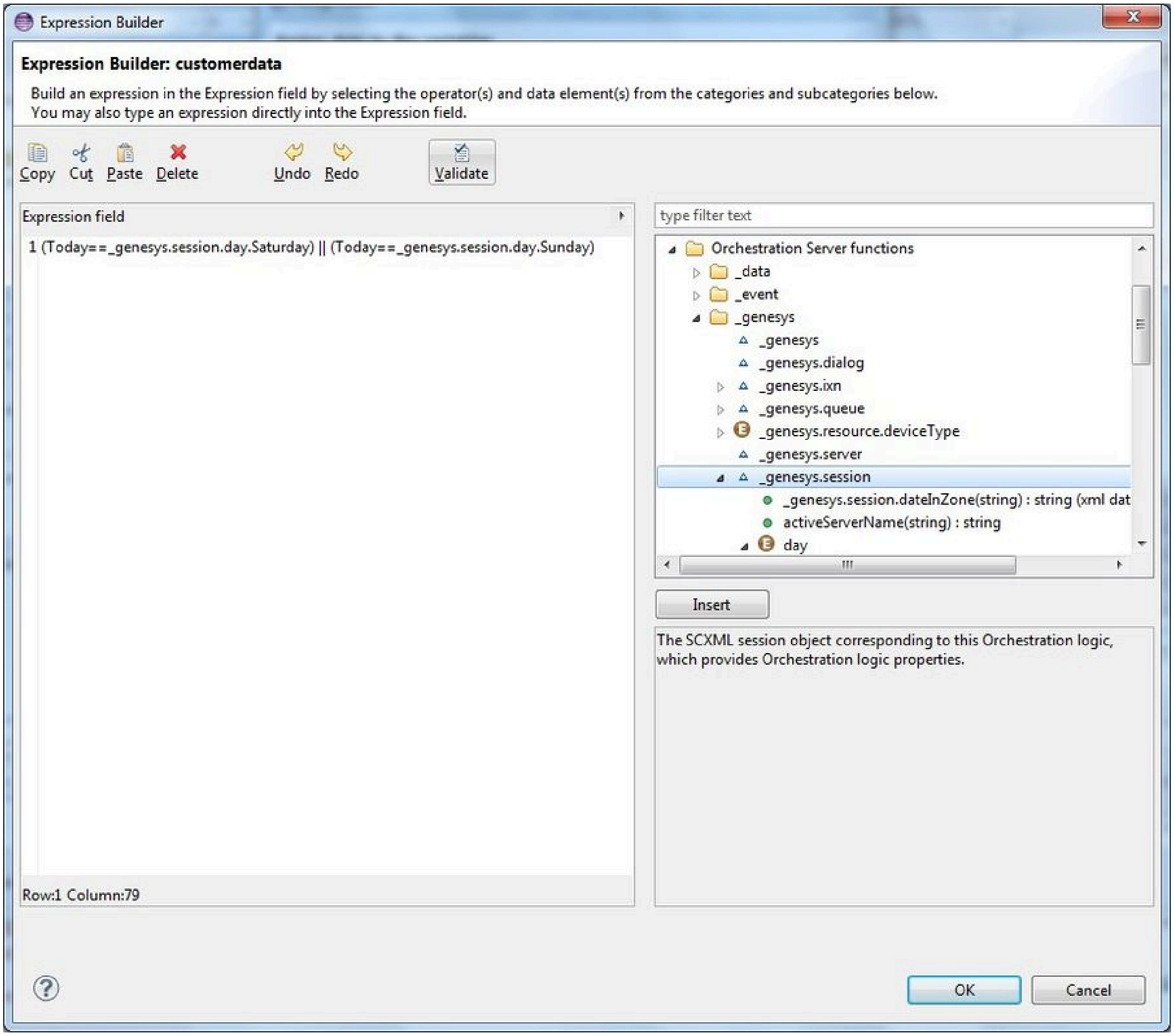
- **Composer** , for both voice and routing applications, shows the Project Explorer, Outline view, canvas, palette, and can show the following tabs in the lower pane: Properties, Prompts Manager, Problems, Console, and Call Trace.

- **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 ECMAScript expressions that use Genesys supplied SCXML functions documented in the the Orchestration Server Developer's Guide, which is the Genesys language specification.

The figure below shows an example Genesys-supplied SCXML function in Expression Builder.



Expression Builder: customerdata

Build an expression in the Expression field by selecting the operator(s) and data element(s) from the categories and subcategories below.
You may also type an expression directly into the Expression field.

```
(Today==_genesys.session.day.Saturday) || (Today==_genesys.session.day.Sunday)
```

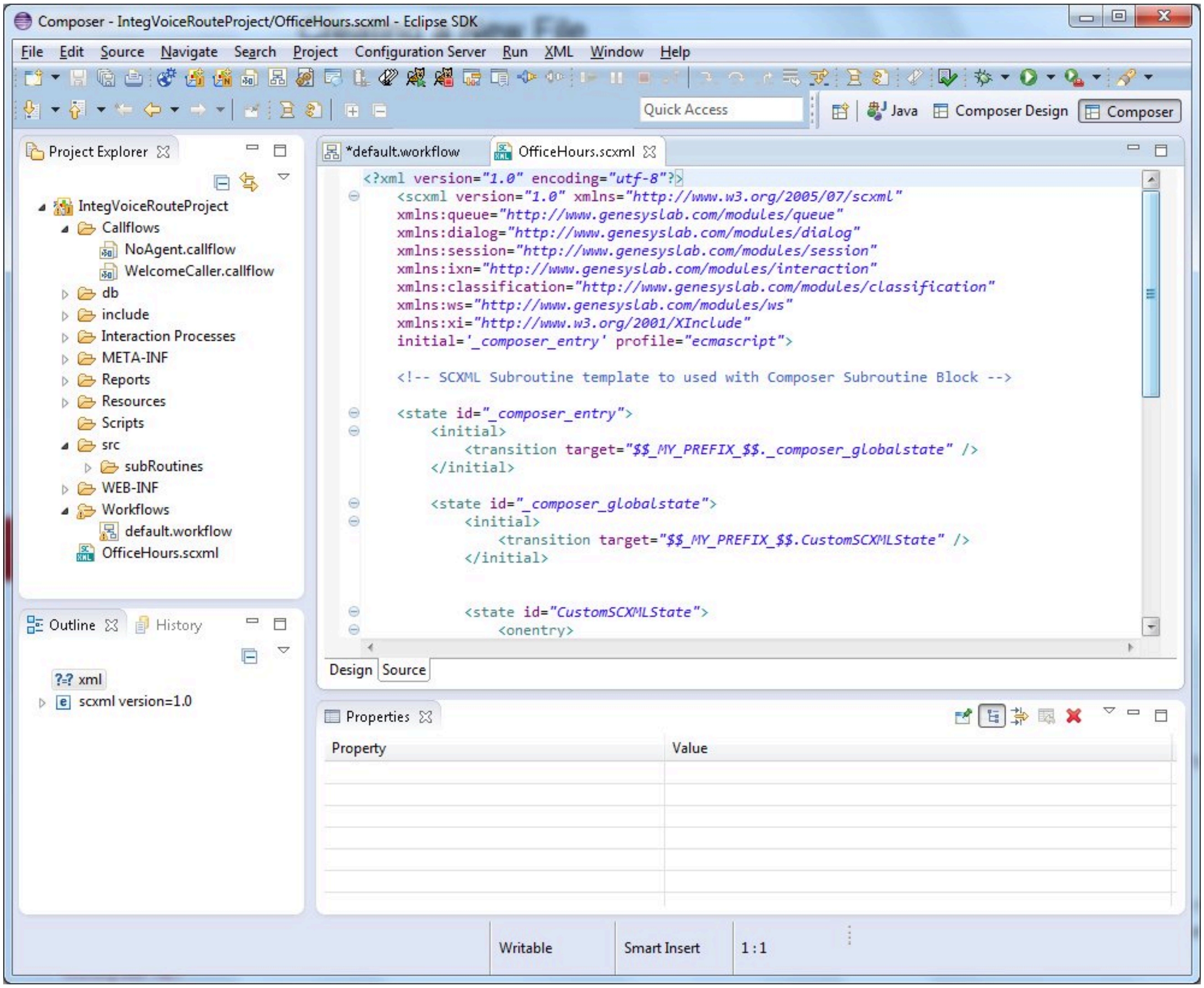
Row:1 Column:79

The SCXML session object corresponding to this Orchestration logic, which provides Orchestration logic properties.

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 *Source* tab of the editor.
- The second figure below shows the *Design* tab of the editor.



```
<?xml version="1.0" encoding="utf-8"?>
    <scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
    xmlns:queue="http://www.genesyslab.com/modules/queue"
    xmlns:dialog="http://www.genesyslab.com/modules/dialog"
    xmlns:session="http://www.genesyslab.com/modules/session"
    xmlns:ixn="http://www.genesyslab.com/modules/interaction"
    xmlns:classification="http://www.genesyslab.com/modules/classification"
    xmlns:ws="http://www.genesyslab.com/modules/ws"
    xmlns:xi="http://www.w3.org/2001/XInclude"
    initial='_composer_entry' profile="ecmascript">

    <!-- SCXML Subroutine template to used with Composer Subroutine Block -->

    <state id="_composer_entry">
        <initial>
            <transition target="$$_MY_PREFIX_$$._composer_globalstate" />
        </initial>

        <state id="_composer_globalstate">
            <initial>
                <transition target="$$_MY_PREFIX_$$.CustomSCXMLState" />
            </initial>


            <state id="CustomSCXMLState">
                <onentry>
```

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.