Composer Help

Starting a New IPD

5/3/2025

Starting a New IPD

Contents

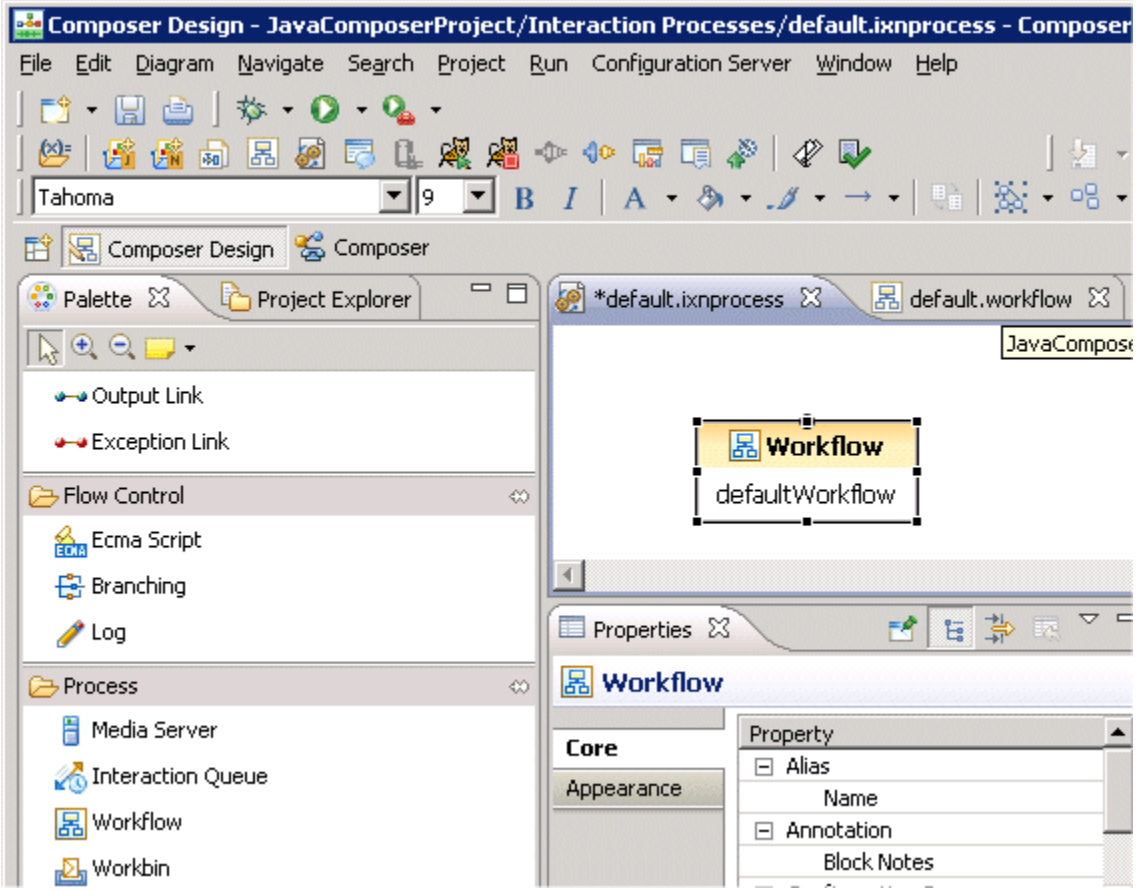
Use these instructions after reviewing the IPD Planning and Preparation topic. Before starting a new IPD, you may wish to Connect to Configuration Server (optional). You can do this now or during the validation phase. When not connected, Configuration Database objects do not appear for selection in Composer dialog boxes; you must know the names in advance.

Method #1: Create a New Java Composer Project

1. Create a new Java Composer Project. In Composer Design perspective, this automatically creates an Interaction Processes folder and default.ixprocess file in the Project Explorer. It also automatically creates the following tabs on the canvas:

    - `default.ixprocess`
    - `default.workflow`

The `default.ixprocess` tab contains a starting Workflow block. The Palette tab shows the Media Server, Interaction Queue, and Workflow blocks. It also access certain Flow Control blocks.



You can rename the IPD at any time by right-clicking the `default.ixnprocess` file in the `Interaction Processes` folder in the Project Explorer and selecting **Rename**. The renaming

operation does not result in any changes being written to Configuration Server.

Method #2: New IPD Diagram Wizard

1. Bring up the wizard. There are various ways:
   - Click the Create New Interaction Process button on the toolbar.
   - Or select **File** > **New** > **Other** > **Composer** > **Diagrams** > **Interaction Process Diagram**.
   - Or in the Project Explorer > Right-click > **New** > **Other** > **Composer** > **Diagrams** > *Interaction Process Diagram*.
2. Name the IPD
3. Select or create the associated Project.
4. Click **Finish**.

Composer creates an Interaction Processes folder and default.ixprocess file in the Project Explorer. It also automatically creates default.ixprocess and default.workflow tabs on the canvas. The `default.ixprocess` tab contains a starting Workflow block. The palette shows the Media Server, Interaction Queue, and Workflow blocks. Note: An IPD does not use Entry or Exit blocks.

> ### Tip
> To display the IPD properties below, select an *.`ixn.process` tab above the design area, then click inside the design area,

An IPD contains the following properties:

Name Property

This property shows the name of the diagram. An IPD diagram can be renamed at any time. The renaming operation will not result in any changes being written to Configuration Server.

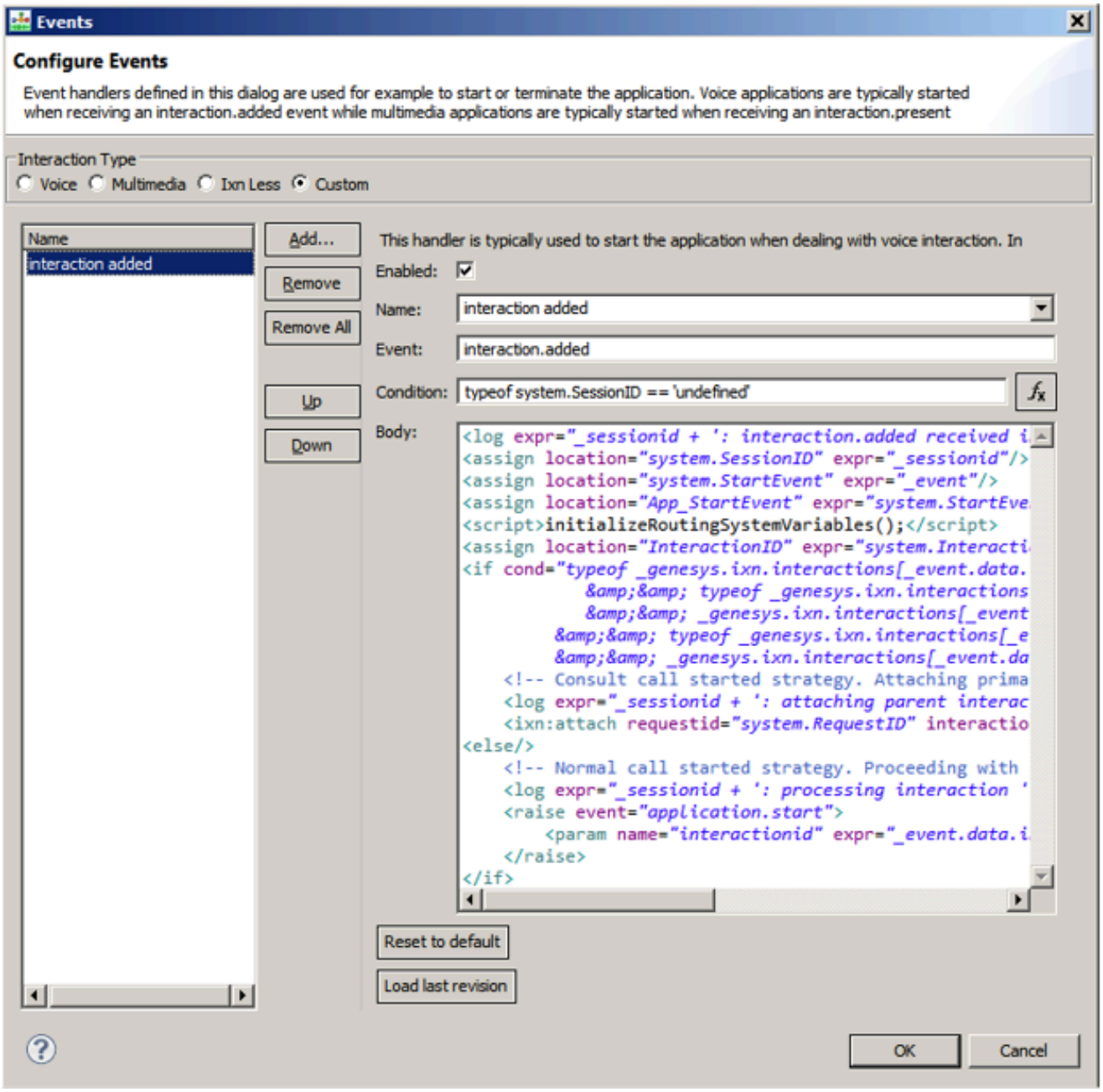
Events Property

With the `.ixnprocess` tab selected, click the empty space in the IPD to see Events in the Properties view.

The Events property (which replaces the 8.1.2 Wait for Event property) works with the Interaction ID property in Routing and certain eServices blocks. You select/enter the event(s) that the generated code will wait for before the workflow code is invoked. If unset, the IPD code will not wait for any event before invoking the workflow code. ORS will transition on the first event received. The

Application does not need to receive all declared events to transition to the next block.

Starting with Release 8.1.400.33, Composer supplies default handler sets (voice, multimedia, interaction-less processing) and a **Custom** option for customizing handlers. You can change the set at any point of time and generate code. Pre-defined sets are non-editable and should be used for specific media processing. The **Custom** type can be used to customize the handlers. The Configure Events dialog box, which opens from the Events property, is shown below when `interaction.added` is selected.



Refer to Intra Version Upgrades for event upgrade logic.

Event Handlers

All system handlers run into the system thread of the application while the workflow generated code runs into the user thread of the application.

Event handlers can control the lifecycle of the user thread by raising the `application.start` event (to start the application, see the interaction added event handler) or the `application.exit` event (to terminate the application, see the `interaction deleted` event handler).

To define this property in case of "interaction-less" processing (defined below), you may:

- **Wait for a user-defined event**:
  - Add a new Event item in the Events property list.
  - Specify the appropriate Event name.
  - In the body of that event, add the code: `<raise event="application.start">` to start the user thread (to run the workflow SCXML).
- **Start the application without waiting for any event**:
  - Remove all Events items
  - Or disable all Events items
  - Or in the Events dialog box, add the predefined "interaction-less processing" event (or any similar event having no Event AND no Condition defined).

In addition to catch all errors, Composer defines the following events:

- `interaction.present` or `interaction.added`. This property will be used by default as a value of the IPD/Events property.
- `interaction.onmerge`. This event will be caught by the generated IPD SCXML. The value of the variable system.InteractionID will be updated when the transfer is completed. The parent or primary interaction ID will be referred to instead of the consult interaction ID.
- `interaction.deleted` (active interaction id). This event will be caught by the generated IPD SCXML. Default behavior will be to exit the session. Default behavior can be overrriden if interaction.deleted is handled in the workflow diagram.
- `interaction.deleted` (consult interaction id). This event will be caught by the generated IPD SCXML. Default behavior will be to exit the session if the parent interaction is gone or to do nothing otherwise. Default behavior can be overridden if interaction.deleted is handled in the workflow diagram.

Note: The condition expression for event-related properties in interaction process (IPD) and workflow diagrams are not XML-escaped when generating the SCXML code. For more information, see Troubleshooting ORS Compile Errors and Non Escaped Special Charcters.

Interaction-less Processing

In the case of "interaction-less" processing (for example, see the Force Route Block Interaction ID Property), you can start an ORS session using the ORS REST API and then decide either not to wait (no Wait For Event defined) or to wait for a user-defined event. The ORS REST API allows you to send an event to a particular ORS session. To define this property:

1. Click under Value to display the  button.
2. Click the  button to open the Wait For Event dialog box.
3. Do one of the following:

    - Leave interaction.present to keep the default value, the system variable InteractionId, which will be initialized automatically in this case.
    - Click Add and select from the list of SCXML events or enter the event name. After selecting an event, the dialog box displays a description of the event.

A system variable, AppStartEvent, will be generated in the IPD <datamodel>, which will be initialized to the contents of the specified start event. If unset, the variable will be set to undefined (not the string undefined).

    - If necessary, click Remove to clear a selected event.

4. Click OK.

Created By Property

To be filled in by the user/author of the document.

Created On Property

Auto-populated by Composer to indicate the timestamp when the diagram was created.

Designed Using Property

Auto-populated by Composer to indicate version of Composer used to create this diagram.

Last Modified By Property

Provided by the user to indicate who updated the diagram last.

Last Modified On Property

Filled in by Composer when the diagram is modified.

Version Property

Provided by the user for versioning purposes during development.

Namespaces Property

Use to refer to custom namespaces in the generated code. To define this property:

1. Click under Value to display the  button.
2. Click the  button to open the Namespaces dialog box.
3. Click Add.
4. Enter the namespace prefix (see example below)
5. Enter the namespace URL (see example below)
6. Click OK.

When an event is sent to an ORS session via http, a response can be sent back from the session via http by using the `ws:response` tag as shown below. `<ws:response requestid="_data.reserveSendId" resultcode="JSON.stringify( resultReserve )" />` Namespace: xmlns:ws=[http://www.genesyslab.com/modules/ws http://www.genesyslab.com/modules/ws]

Extensions Property

This attribute allows arbitrary attributes to be added to the root <scxml> element and is used by Orchestration Server to control persistence, session recovery, and other functionality.

Note: Composer-generated SCXML applications do not support the `w3c` value for the `_transitionStyle` extension attribute.

Refer to `<scxml>` element, `_transitionStyle` extension attribute in the SCXML Language Reference section of the Orchestration Server Developer's Guide for details.

Read Context Property

This is the counterpart of the Pass Context property in the Routing Blocks. The value of this property is true or false. When true, the application will try to read the URL of the originating session from the interaction's User Data. If that URL is defined, it will then attempt to fetch the context from the originating Orchestration Session.

Session Persistence Property

With the `.ixnprocess` tab selected, click the empty space in the IPD to see Session Persistence in the Properties view.

Select true or false. Use this property to set the _persist attribute of the <scxml> tag when generating the SCXML code. For more information, see the <scxml> element Attribute Details, SCXML Language Reference section, in the Orchestration Server Developer's Guide.

Deleting Blocks

IPD diagram non-linked blocks can only be deleted directly from the IPD Diagram canvas. To delete the linked blocks, go to the corresponding workflow diagram and delete or modify the routing blocks.