



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 Help

ECMAScript Block

ECMAScript Block

Contents

- **1 ECMAScript Block**
 - 1.1 Name Property
 - 1.2 Block Notes Property
 - 1.3 Exceptions Property
 - 1.4 Condition Property
 - 1.5 Logging Details Property
 - 1.6 Log Level Property
 - 1.7 Enable Status Property
 - 1.8 Script Property
 - 1.9 ORS Extensions Property
 - 1.10 Using Genesys Extensions

Orchestration Server (ORS) 8.0+ supports **SCXML** plus ECMAScript as a routing language for use in Composer when creating routing workflows. While the core SCXML provides State Chart functionality, you can specify ORS-specific instructions, such as conditions that can be used for routing decisions, in the form of ECMAScript. The Script property brings up Composer's Expression Builder for creating those conditions in the form of expressions. Use the ECMAScript block to build an ECMAScript expression.

Notes:

- The ECMAScript block supports general ECMAScript in addition to ORS-specific **Extensions**.
- If the Composer Project contains a folder at include/user, then any files with extension .js will be included in the generated SCXML. This allows you to write custom ECMAScript and include it in the application.
- To support creating multiple **views** per interaction queue, the ECMAScript block is available when **creating an IPD**.
- To set external event processing when transitioning out of ECMAScript blocks, select Properties from the Project menu. A dialog box opens showing the properties of the selected Project or of the Project that contains the selected resource. Select Orchestration Server Options to set **external event processing**.
- Also see the **SCXML State Block**.

The ECMA Script block has the following properties:

Name Property

Find this property's details under **Common Properties**.

Block Notes Property

Find this property's details under **Common Properties**.

Exceptions Property

Find this property's details under **Common Properties**.

- For callflows, invalid ECMAScript expressions may raise the following exception event: `error.semantic`
- For workflows, invalid ECMAScript expressions may raise the following exception events: `error.script.SyntaxError` and `error.script.ReferenceError`

You can use [custom events](#) to define the ECMAScript exception event handling.

Condition Property

Find this property's details under [Common Properties](#).

Logging Details Property

Find this property's details under [Common Properties](#).

Log Level Property


Find this property's details under [Common Properties](#).

Enable Status Property

Find this property's details under [Common Properties](#).

Script Property

To create an ECMAScript expression in [Expression Builder](#):

1. Click opposite **Script** under **Value**. This brings up the  button.
2. Click the  button to bring up Expression Builder.

Expression Builder gives access to various [categories of data](#), which can be used in expressions. To create an expression, follow the instructions in the [Creating Expressions](#) topic.

Excluding Agents

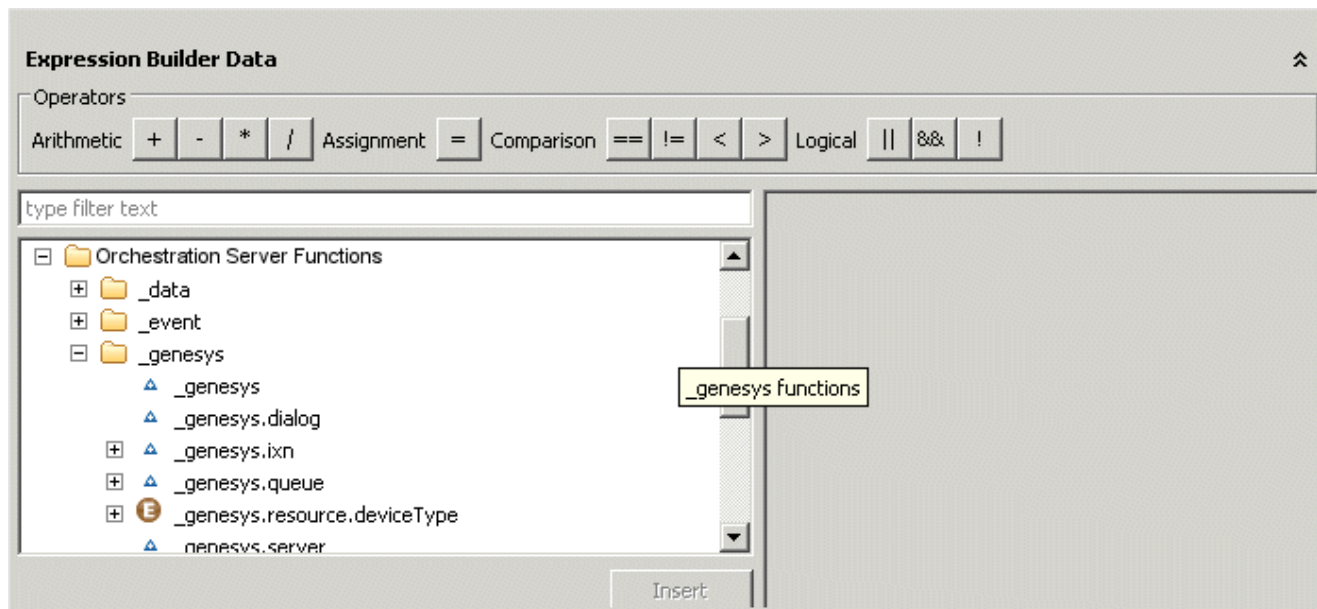
Note: When `_genesys.queue.excludeAgents` is used in a routing workflow before a Target block, the URS-provided list of excluded agents will be applied to the current or any future Target block. The effect of the `_genesys.queue.excludeAgents` execution can be cancelled only by the execution of another `_genesys.queue.excludeAgents` or if URS stops this interaction processing.

ORS Extensions Property

Starting with 8.1.4, Composer blocks used to build routing applications (with the exception of the Disconnect and EndParallel blocks) add a new **ORS Extensions** property.

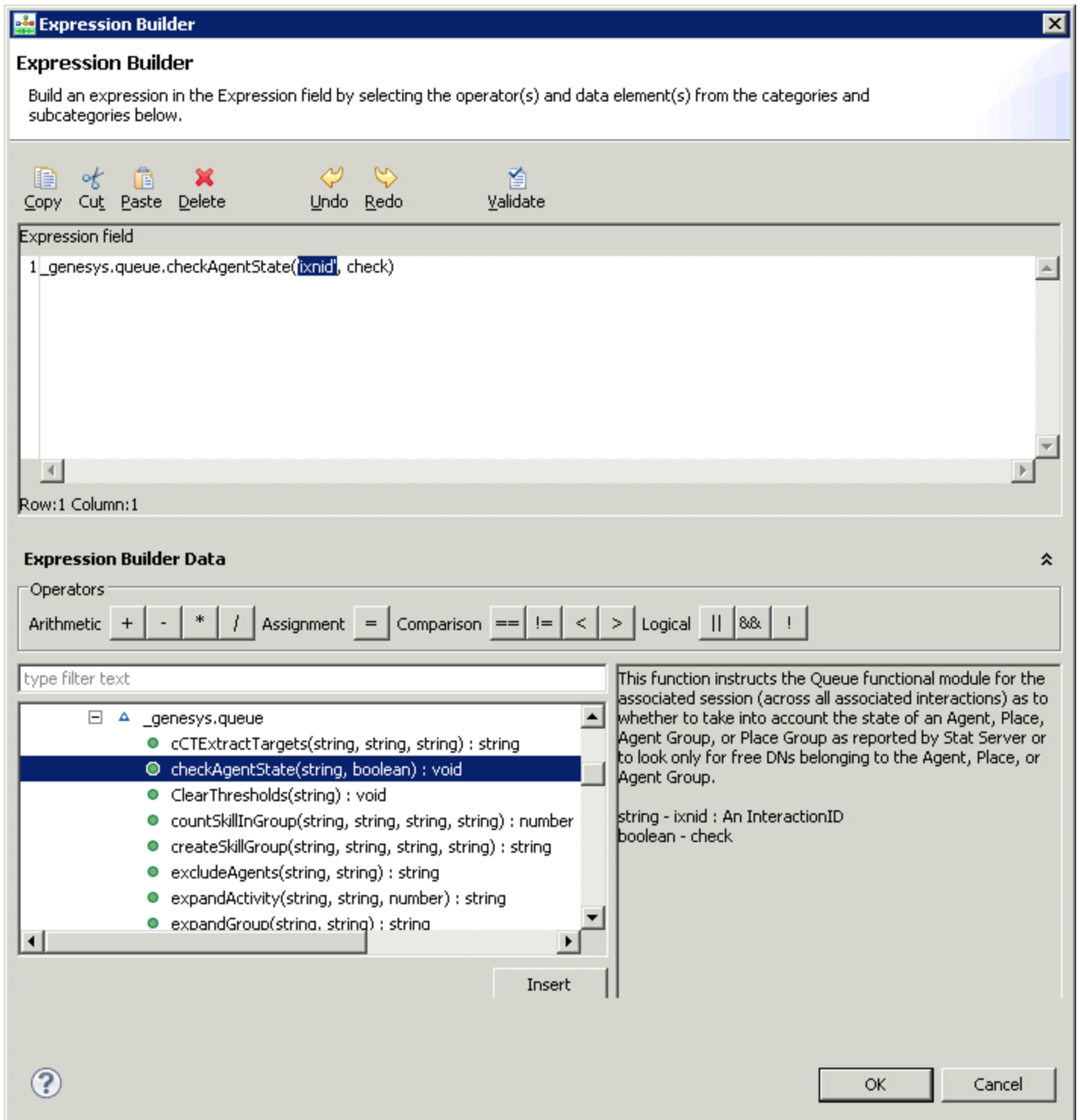
Using Genesys Extensions

Assume you expand Orchestration Server Functions in Expression Builder.



The Orchestration Server Functions category shows various Genesys-supplied Orchestration Extensions described in the *Orchestration Server Developer's Guide*, such as the `genesys.queue.checkAgentState` extension shown below. Also, the *Universal Routing 8.1 Reference Manual* describes many URS equivalent functions, which have similar names but are not necessarily equivalent. For example, the Functions chapter of that manual describes a `CheckAgentState` function. These functions are intended to be called in *Interaction Routing Designer*, which was historically used to create routing strategies prior to Composer.

Assume you double-click `genesys.queue.checkAgentState`. Expression Builder now appears as shown below.



ECMA1.gif

In this case, the `genesys.queue` module implements the target selection functionality of URS (finding resources for interactions and delivering interactions to the resource). When URS executes these extensions, it returns events back to the instance of logic running the SCXML document that requested the action.