



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

Backend Common Block

Backend Common Block

Contents

- **1 Backend Common Block**
 - 1.1 Name Property
 - 1.2 Block Notes Property
 - 1.3 Exceptions Property
 - 1.4 Uri Property
 - 1.5 Encoding Type Property
 - 1.6 Parameters Property
 - 1.7 Pass State Property
 - 1.8 Condition Property
 - 1.9 Logging Details Property
 - 1.10 Log Level Property
 - 1.11 Enable Status Property
 - 1.12 ORS Extensions Property

The Backend block is used for both routing and voice applications. Use to invoke custom backend Java Server Pages (JSP). You have the option to pass back all the application session state data to the backend logic page on the server. Data being returned will be sent back as a JSON string. Other features:

- Provides a mechanism for creating new backend logic JSP. The added JSP file will have a basic template code already filled out. As the application developer, you will only need to implement a performLogic function. The VXML/SCXML to return back control will be auto-generated in the template.
- User-written custom backend logic pages are stored in the Java Composer Project's src folder. Composer provides standard include files for Backend logic blocks in the Java Composer Project's include folder.

Important

If any custom backend logic pages use libraries, place the libraries in the Java Composer Project's WEB-INF/libdirectory. This directory typically contains JAR files that contain Java class files (and associated resources) required for the application. The Tomcat application server should be restarted after changing any JAR files in this folder. Composer includes a CHEAT SHEET for creating a Backend logic application as well.

The Backend block has the following properties:

Name Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Block Notes Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Exceptions Property

Find this property's details under [Common Properties for voice blocks](#) or [Common Properties for Workflow Blocks](#).

Uri Property

The Uri property specifies the http:// page to invoke. To set a URL destination for the Uri property:

1. Select the Uri row in the block's property table.
2. In the Value field, click the  button to open the Uri dialog box.
3. Select a file from the available projects.

Encoding Type Property

The Encoding Type property (used for callflows only) indicates the media encoding type of the submitted document. GVP 8.1 supports two encoding types:

- application/x-www-form-urlencoded
- multipart/form-data

To select a value for the Encoding Type property:

1. Select the **Encoding Type** row in the block's property table.
2. In the **Value** field, select **application/x-www-form-urlencoded** or **multipart/form-data** from the drop-down list.

Parameters Property

Note: Parameters cannot be entered until the Uri property is specified. Use the Parameters property to specify parameters to pass to the invoked backend JSP. To specify parameters:

1. Click the Parameters row in the block's property table.
2. Click the  button to open the Parameter Settings dialog box.

Add Button Use the Add button to enter parameter details.

1. Click **Add** to add an entry to Backend Parameters.
2. In the **Parameter Name** field, accept the default name or change it.
3. From the **Parameter Type** drop-down list, select **In**, **Out**, or **InOut**:

In	Input parameters are variables submitted to the Backend application.
Out	Output parameters are variables that the Backend application returns and will be reassigned back to the current callflow.

InOut

InOut parameters are parameters that act as both input and output.

4. In the **Expression** drop-down list, select from among the listed variables, type your own expression, or click the  button to use **Expression Builder**.
5. In the **Description** field, type a description for this parameter.
6. Click **Add** again to enter another parameter, or click OK to finish.

Delete Button To delete a parameter:

1. Select an entry from the list.
2. Click **Delete**.

Pass State Property

Note: This property is used for callflows only. The Pass State property Indicates whether or not to pass the application state to the backend. The application state includes all the variables shown in the Entry block as well as all variables containing returned values from user Input blocks. You can find Instructions on how to access these backend variables in [Creating a Backend JSP File](#) and [a Backend_ASP_.NET Creating a Backend ASP.NET File](#). The **Parameters** property can also be used to pass specific parameters into the backend and, for efficiency reasons, should be considered first. There is also a Cheat Sheet, [Creating a Backend Logic Block \(Help > Cheat Sheets > Composer > Building Voice Applications\)](#). To select a value for the Pass State property:

1. Select the **Pass State** row in the block's property table.
2. In the **Value** field, select **true** or **false** from the drop-down list.

Condition Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Logging Details Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Log Level Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

Workflow Blocks.

Enable Status Property

Find this property's details under [Common Properties for Callflow Blocks](#) or [Common Properties for Workflow Blocks](#).

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.