



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

## Entry Block and Variables

# Entry Block and Variables

## Contents

- **1 Entry Block and Variables**
  - 1.1 Name Property
  - 1.2 Block Notes Property
  - 1.3 Exceptions Property
  - 1.4 Application Root Property
  - 1.5 Global Commands Property
  - 1.6 Global Properties Property
  - 1.7 Scripts Property
  - 1.8 Variables Property
  - 1.9 System Variables
  - 1.10 Condition Property
  - 1.11 Logging Details Property
  - 1.12 Log Level Property
  - 1.13 Enable Status Property

Use an Entry block to begin an application. Only one Entry block can be present in each application. The Entry block:

- Sets global error (exception) handlers.
- Defines all global application-level properties, global variables (which appear in the list of available variables for other blocks in the diagram), and global commands. See topic [Variables in Callflows](#).
- Sets default application scripts and parameters.
- [Accesses Expression Builder](#).

The Entry block is used as the entry point for a main callflow or a sub-callflow. It contains the list of all the variables associated with the callflow (referred to as global variables). Note: [Outlinks](#) starting from the Entry block cannot be renamed or assigned a name through the Properties view. The Entry block has the following properties:

### Name Property

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

### Block Notes Property

Can be used for both callflow and workflow blocks to add comments.

### Exceptions Property

Find this property's details under [Common Properties](#). The Entry block has all global exception events, with the defaults of all, connection.disconnect.hangup, and error. Also see [Exception Events](#).

### Note on No Input and No Match Events

When selecting exceptions for the Entry block, use both `com.genesyslab.composer.toomanynoinputs` / `com.genesyslab.composer.toomanynomatches` and `noinput/nomatch` to catch all the possible no input and no match events. The selection of `com.genesyslab.composer.toomanynoinputs` / `com.genesyslab.composer.toomanynomatches` is required when `noinput` / `nomatch` exceeds the maximum retries in the lower block. The selection of `noinput` / `nomatch` is required when the lower block does not retry at all.

- `com.genesyslab.composer.toomanynoinputs` occurs when the number of no inputs exceeds the maximum retries in the Menu, Input, DBInput, and Record blocks, and the blocks do not have local noinput exception ports.
- `com.genesyslab.composer.toomanynomatches` occurs when the number of no matches exceeds the maximum retries in the Menu, Input, DBInput, and Record blocks, and the blocks do not have a local nomatch exception port.

### Note on error.badfetch.badxmlpage

NGI no longer supports this event. If upgrading an application from an earlier version of Composer that supported this event in its Entry object, you will need to modify that object via the Exceptions property dialog box.

## Application Root Property

- Starting with 8.1.410.14, Composer Projects have a default root VXML file (ComposerRoot.vxml) bundled inside the src folder. This file, which you can edit to create Project-level defaults such as Input and Menu block variables and values, is present in newly created Projects and upgraded Projects. New Callflow diagrams have this default root VXML document automatically configured in the Application Root property of the Entry block.

You have the option to specify a VXML file to be used as an application root document allowing multiple callflows to share variables. Background: Starting with 8.1.1, each Composer **Project** can have (at most) one root document (VXML file). If a Project has no root document, each callflow is its own stand-alone application. If a Project contains a root document, the set of callflows with Entry blocks that reference that root document make up the application.

- If a callflow or sub-callflow references an application root document, the variables specified in the application root become available for selection in all dialogs in that diagram.
- Variables defined in the application root directly under the <vxml> tag become available as global variables to callflows and sub-callflows that access them.

To select an application root document:

- Click the Application Root row in the block property table.
- Click the  button to open the Select Resource dialog box.
- Select the **VXML** file in the Project src folder and click OK.

## Global Commands Property

The Global Commands property sets rootmap elements for the entire application. A rootmap element is a phrase (user-defined phrase or external grammar) and/or tone the application reacts to at any time the application is running. Use the Global Commands property to set rootmap elements for the entire application. The application uses these rootmap elements as global grammars (subsets of a spoken language that callers are expected to use) in each **Input block**. Composer creates one output for each rootmap element; the output specifies the application path in the event to which the rootmap element is matched. Use the Entry block Global Commands property to set rootmap elements for a subcallflow as well. Note: The RootMap elements defined in the Entry block do not apply to blocks inside a subcallflow. To add, delete, or arrange global phrases, DTMF keys, and grammars:

- Click the Global Commands row in the block's property table.

2. Click the  button to open the Set Rootmap Commands dialog box.

### Fields in Set Rootmap Commands Dialog Box

- Name-- Displays the name of the command.
- DTMF Option--Displays the DTMF key to recognize.
- Phrase-- Displays the phrase to recognize.
- Grammar--Displays the built-in or custom grammar used.

Genesys recommends that you use only the GRXML grammar. Otherwise, GSL support--which is not a part of the VoiceXML 2.1 specification--deprecates over time. **Note:** Built-in grammar support for languages other than U.S. English is dependent on the ASR vendor. Before using this feature, make sure that your ASR Engine supports built-in grammars for your language.

## Add Button

Use the Add button to enter global phrases, DTMF keys, and grammars.

1. Click Add to enable Command Details fields.
2. In the Name\* box, accept the default name or change it.
3. From the DTMF Option drop-down list, select the global DTMF key.
4. In the Phrase box, type the phrase.
5. In the Grammar drop-down list, select a grammar. The grammar source is the custom or built-in grammar for recognition.

## Up/Down Buttons

Use the Up and Down buttons to reorder your rootmap elements. Select the element you want to reposition, and then click Up or Down, as necessary.

## Delete Button

To delete a phrase, DTMF key, or grammar entry:

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

## Global Properties Property

This property allows suppression of data within the Nuance 9 platform ASR logs. For more information on this property, see the Properties topic on the [Genesys Voice Platform wiki](#). Use Global Properties to select global settings for **VXML properties**, Automatic Speech Recognition vendor-specific properties or Text-to-Speech vendor-specific properties. To enter properties and values:

1. Click the Global Properties row in the block's property table.
2. Click the  button to open the Global Property Settings dialog box.
3. Click Add to enable the Property Name and Property Value fields.
4. Enter or select a Property Name by doing one of the following:
  - Select the Property Name from the drop-down list, or
  - Type the Property Name in the Property Name field.
5. Enter or select a Property Value by doing one of the following:
  - Select the Property Value from the drop-down list, or
  - Type the Property Value in the Property Value field.
6. Click OK.

## Scripts Property

Use the Scripts property for including custom JavaScript includes into the application. The JavaScript functions in the specified .js file can then be used in the Assign or Branching blocks in the expression.

1. For this property, enter the filename of your file (for example: script.js). If there are multiple files to be loaded, you can delimit by using the | character; for example: script1.js|script2.js.
2. Then place the custom ECMAScript file in the Scripts subfolder of your project.

There is also a Global Variable SCRIPTSDIR, which specifies the default folder for the scripts files (and works very similar to VOXFILESDIR for audio files).

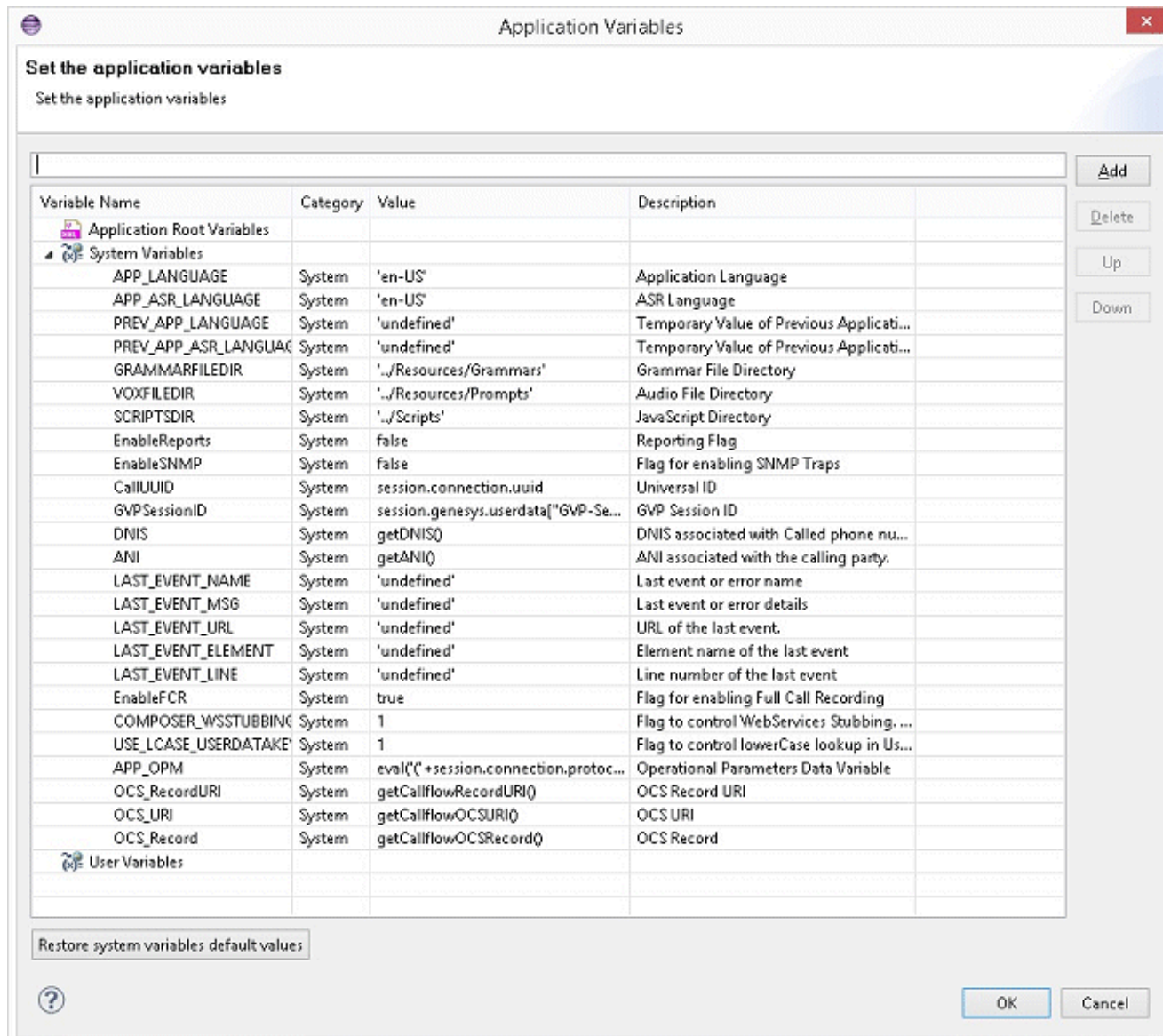
## Variables Property

Variables can be predefined system variables (provided by Composer, which you cannot delete) or user-defined variables. See the [Variables in Callflows](#) topic for more information. Many Composer blocks have properties that require you to select a variable. **Examples:**

- The following callflow blocks contain a mandatory **Output Result property**: **Menu**, **Record**, **DB Input**, **Grammar Menu**, **Input**, **Get Access Number**, **Transfer**, and **Statistics**. After defining variables in the Entry block, you supply this property by selecting the variable to contain the output result.
- When creating a new voice project, a Project-level flag, Enable\_ICM, controls whether **ICM** variables are available for selection and assignment to variables within Composer's Entry block.
- For information on user data and GVPSessionID, see the [Project Properties dialog box](#), Composer Callflow Option.

To declare for the application or subcallflow:

1. In the Properties tab, click opposite Variables under Value to display the  button.
2. Select **Project**, **System**, or **User** Variables.
3. Click the arrow to display the selected type. An example **System Variables** dialog box is shown below.



The above figure shows the dialog box after clicking the **Add** button. The Value field for the new variable (Var0) contains a button to access **Expression Builder**.

## GVPSessionID System Variable

Composer Projects have a callflow options property page to control how the GVPSessionID system variable is initialized. It can be used to control if it is initialized from the X-Genesys-GVP-Session-ID SIP header or the session.com.genesyslab.userdata object.

### Restoring System Variable Default Values

Projects created in earlier versions of Composer may throw runtime errors due to incorrectly initialized system variables after upgrading to Composer 8.1.3. This was due to changes in how system variables were stored and handled in 8.1.3. To resolve this, the Entry block Variables dialog adds a button to restore system variables to default values, which can be used to reset variables and fix initialization. Note that this also removes any custom values set in system variables. As system variables cannot be updated, after clicking the **Restore System Variables Default Values** button, you cannot update the customized system variables.

Starting with 8.1.410.14, you can:

- Invoke the Entry Block variables dialog when a property is selected in the Properties view using ALT+V.
- Enable Composer to automatically declare variables in a Main callflow to match input/output variable names in Sub-callflows and perform the mapping. For more information, see the **auto synchronization** option in [Diagram Preferences](#).

### Defining Variables

Important! When defining a variable name, the name:

- Cannot start with **APP\_** (callflow diagrams).
- Must not start with a number or underscore.
- May consist of letters, numbers, or underscores.

When you define and initialize a variable that is expected to be played as a date later on in the callflow, define the value using the following format: `yyyymmdd`. Example: `MyDate=20090618`. You must use this format; Composer does not perform any conversions in this case. When you define and initialize a variable that is expected to be played as a time later on in the callflow, define a 12 hour-based value using the following format: `hhmmssa` or `hhmmssp`. Example: `MyTime=115900a` or `MyTime=063700p`. Define a 24 hour-based value using the following format: `hhmmssh`. Example: `MyTime=192000h`. You must use this format; Composer does not perform any conversions in this case. If variables are set as part of provisioning by the Genesys VoiceXML provisioning system, and if these variables have the same names as variables set in the Variables property dialog box, the VoiceXML provisioning system values take precedence over the global variables set here. Many blocks enable the use of variables rather than static data. For example, the Prompt block can play the value of a variable as Text-to-Speech. Variables whose values are to be used in other blocks must be declared here so that they appear in the list of available variables in other blocks. The value collected by an Input block or a Menu block is saved as a session variable whose name is the same as the block Name.

### System Variables

These variables apply only to the Entry block, unless otherwise indicated.

- **APP\_LANGUAGE**--Holds the application language setting. The value should be the RFC 3066 language tag of an installed language pack. Examples of valid RFC 3066 language tags include `en-US` and `fr-FR`. This setting also acts as a default language for the application. This variable may be set using the Set



Language block for a multilingual application.

- **APP ASR LANGUAGE**--Holds the language locale for ASR resources. You must define this variable if the application needs to use a different language locale for ASR from TTS resources.
- **GRAMMARFILEDIR**--Gives the relative path from the application to the directory that contains the grammar files. By default, it is set to ../Resources/Grammars. If a voice application supports multiple languages, you can enable the application to switch between them, by changing the value of this variable. In the Subcallflow\_Start block, the GRAMMARFILEDIR global variables are not defined by default. This allows the subcallflows to inherit the value of this variable from the main callflow. If the subcallflow overrides this value, the variable can be defined in the Subcallflow\_Start block. (**Note:** Composer uses the getGrammarURI() function (from **common.js**) to build the grammar URL. If you include *http*, *https*, *file*, *rtsp*, or *rtsp*s, then it will just use the provided URL (that is, the URL is encoded and the resultant grammarURI is generated). If not, it will build a URL based on AppState.GRAMMARFILEDIR).
- **VOXFILEDIR**--Gives the relative path in the application to the directory that contains the audio files (.vox/.wav). By default, it is set to ../Resources/Prompts. If a voice application supports multiple languages, you can enable the application to switch between them, by changing the value of this variable.
- **SCRIPTSDIR**--Default location for JavaScript files
- **EnableReports**--Enables VAR reporting. (Reporting blocks)
- **EnableSNMP**--Enables the SNMP block, if present in the application
- **CallUUID**--Session connection Universal ID
- **GVPSessionID**--The Genesys Userdata Session ID
- **LAST\_EVENT\_NAME**--Stores the name of the last event or error that was handled in the Entry block.
- **LAST\_EVENT\_MSG**--Stores the message of the last event or error that was handled in the Entry block
- **LAST\_EVENT\_URL**--Stores the URL of the last event or error that was handled in the Entry block.
- **LAST\_EVENT\_ELEMENT**--Stores the element name of the last event or error that was handled in the Entry block
- **LAST\_EVENT\_LINE**--Stores the line number of the last event or error that was handled in the Entry block
- **EnableFCR**--A flag for enabling Full Call Recording
- **COMPOSER WSSTUBBING**
- **App\_OPM**--Used for fetching OPM parameters. Stores JSON content passed by GVP in session variables. Available throughout the callflow diagram. The **OPM block** works with this variable by extracting values from it into application variables. Available for main callflows only.
- **OCS\_RecordURI**--Used by **Outbound blocks**. Its default value will be set from userdata passed into the application. For workflows (SCXML):  
\_genesys.ixn.interactions[InteractionID].udata.GSW\_RECORD\_URI. For callflows: (VXML)  
session.com.genesyslab.userdata.GSW\_RECORD\_URI.
- **OCS\_URI**--Used by **Outbound blocks**. Holds the OCS resource path ([http|https]://<host>:<port>). Its default value will be deduced from OCS\_Record\_URI. You may change this variable value in order to use a different OCS application for all Outbound blocks in the workflow.
- **OCS\_Record**--Used by **Outbound blocks**. Holds the Record Handle value deduced from OCS\_Record\_URI.

**Note:** Request URi parameters created in IVR Profiles during the VoiceXML application provisioning

---

are passed to the Composer generated VoiceXML application as request-uri parameters in the `session.connection.protocol.sip.requesturi` session array. An Entry block variable can use these parameters by setting the following expressions to the variable values: `typeof session.connection.protocol.sip.requesturi['var1'] == 'undefined' ? "LocalDefaultValue" : session.connection.protocol.sip.requesturi['var1']`. If parameters are set as part of IVR Profiles provisioning in the Genesys VoiceXML provisioning system, and if these parameters have the same names as variables set in the Entry block's **Variables** property with the above mentioned `sip.requesturi` expression, then the SIP-Request-URI parameters will take precedence over the user variable values set in the Entry block.

Many blocks enable the use of variables rather than static data. For example, the **Prompt** block can play the value of a variable as Text-to-Speech. Variables whose values are to be used in other blocks must be declared here so that they appear in the list of available variables in other blocks. The value collected by an **Input** block or a **Menu** block is saved as a session variable whose name is the same as the block name.

### Variable Name

You can use the Variable name field for either of the following purposes:

- To enter the name of a new variable.
- To change the name of an existing variable. To do this, select an existing variable from the list of variables. The variable's name appears in the Variable box, and you can change its value in the Value box.

### Excluded Characters

The Variable name field will not accept the following special characters:

- less-than sign (<)
- greater-than sign (>)
- double quotation mark (")
- apostrophe (')
- asterisk (\*)
- ampersand (&)
- pound (#)
- percentage (%)
- semi colon (;)
- question mark (?)
- period (.)

The variable Value field will not accept the following special characters:

- less-than sign (<)
- greater-than sign (>)

- double quotation mark (")
- apostrophe (')
- ampersand (&)
- plus sign (+)
- minus sign (-)
- asterisk (\*)
- percentage (%)

## 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](#).

## Enable Status Property

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