



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

Variables Project and Workflow

# Variables Project and Workflow

## Contents

- **1 Variables Project and Workflow**
  - 1.1 Types of Variables
  - 1.2 Workflow Variables
  - 1.3 Application Variables Dialog Box
  - 1.4 System Variables
  - 1.5 Project Variables
  - 1.6 Internal Variables Naming

This page discusses Project, Workflow, and internal variables.

## Types of Variables

You have the option of defining two types of variables:

1. **Project variables** in the Project Variables dialog box, which opens when you click the **Access Project Variables button** on the toolbar with the IPD in focus. Use Project variables when you need to share information across different workflows. Once defined, Project variables are accessible for use in expressions in Expression Builder.
2. **Workflow variables** in the **Entry** block. Use workflow variables when you need to share information across different blocks in the same workflow. For example, the **Assign block** allows you to assign entered values or values created in **Expression Builder** to variables. Once defined, workflow variables are accessible for use in expressions in Expression Builder.

As can be seen in the Entry block **Applications Variables dialog box**, the workflow variables subtypes are as follows

- **System**—Pre-defined system variables hold Project and application-related values. See **Default Routing System Variables** below. You cannot delete system variables, but applications can modify their values.
- **User**—User-defined custom variables that you create by clicking the Add button in the Application Variables dialog box shown below and selecting User. Applications can delete and modify these types of variables.
- **Input**—Variables which are supplied as input to the called diagram. Created by clicking the Add button in the Application Variables dialog box shown below and selecting User. During runtime, Input variables get automatically filled from the calling context. Typically Input variables are created on the Subworkflow side to notify the Main Workflow about the Parameter passing details while designing the Application flow. Composer does auto-synchronization of the Input variables in the **Subroutine block** and also in the **Play Application** block (CTI).

## Workflow Variables

The following block properties can be specified as workflow variables:

- Target Block properties: Statistic, Timeout, Target Name (if type = Variable), Target Component Selected, Target Object Selected, Target Selected, Virtual Queue Selected, Virtual Queue
- Play Application properties: Resource
- Play Sound Block properties: Resource, Duration
- Play Message Block properties: Prompts > Values field in Prompts dialog box
- User Input Block properties: Prompts > Values field in Prompts dialog box. AbortDigits, BackspaceDigits, Collected Digits Variable, IgnoreDigits, Number of Digits, Termination Digits, ResetDigits, Resource, StartTimeout, DigitTimeout, TotalTimeout, Verification Attempts, Verification Data

- Set Default Route Block: Destination property
- Route Interaction Block: Statistics property
- Subroutine Block: Parameters property
- Stop Interaction Block Reason to Stop Interaction Property
- Context Services: All blocks have certain properties that allow you to select a variable.
- eService: All blocks have certain properties that allow you to select a variable.


### Upgrading from Composer 8.0.2 or earlier

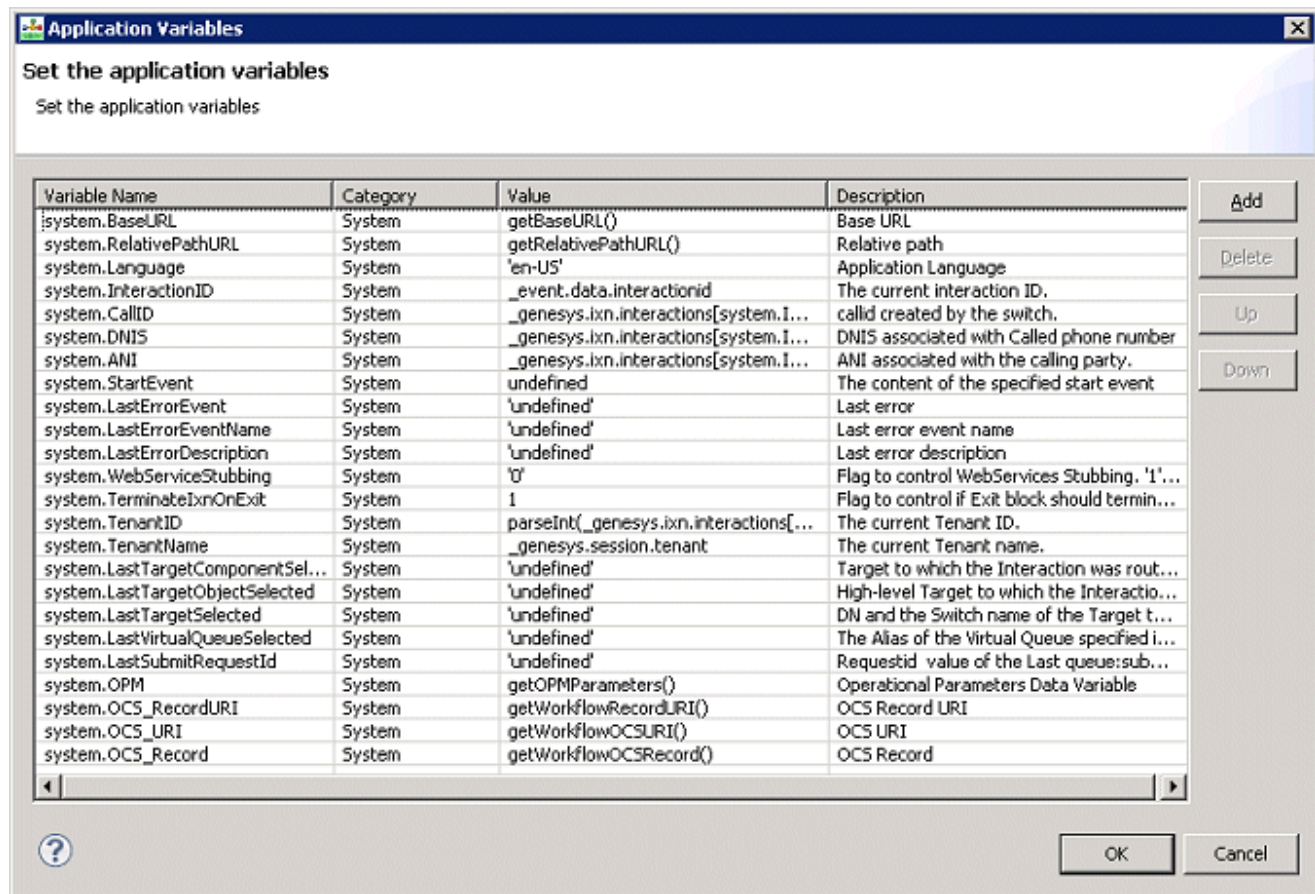
Prior to 8.0.3 release, Composer defined workflow variables in the data model of the SCXML application so they were required to be accessed by prefixing the name of a workflow variable with "\_data.". For example, if you defined a workflow variable named var1, you would access it as \_data.var1. Starting with 8.0.3, Composer defined these variables in the ECMAScript scope so the variable is accessed simply as var1.

### Application Variables Dialog Box

Note: When using the **ORS Debugger**, are not displayed correctly in the **variables view toolbar** if the value contains XML or variables that are of type E4X.

To view workflow variables:

1. In the Properties tab for the Entry block, click opposite Variables under Value to display the  button.  
Or use the **toolbar button**.
2. Click the button to open a dialog box for defining and initializing variables for the workflow.



To add a new variable in the Application Variables dialog box:

1. Click **Add**. Composer add a row for variable and generates a temporary name and number; for example: var7.
2. Select the row and supply the **Name**, **Type**, **Value**, and **Description** fields.
3. Click **OK**.

## System Variables

- **system.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.
- **system.CallID**—Call identifier created by the switch. It is initialized from `_genesys.ixn.interactions[system.InteractionID].voice.callid` (voice only).
- **system.DNIS**—Number that the caller dialed. It is initialized from `_genesys.ixn.interactions[system.InteractionID].voice.dnis` (voice only).
- **system.ANI**—Caller's phone number. It is initialized from `_genesys.ixn.interactions[system.InteractionID].voice.ani` (voice only).

- **system.LastErrorEvent**—Stores the last error that was handled in a block.
- **system.LastErrorEventName**—Stores the name of the error that was handled in a block.
- **system.LastErrorDescription**—Stores the description of the last error that was handled in a block.
- **system.WebServiceStubbing**— Flag to control **Web Services Stubbing** (1 = ON).
- **system.TerminateIxnOnExit**—Used to automatically stop an interaction that was not stopped by the **Route Interaction**, **Queue Interaction**, or **Stop Interaction** block in a multimedia workflow. New workflow entry blocks have this variable pre-populated with 1.
- **system.TenantID**—The current Tenant identifier. It is initialized from `_genesys.session.tenantid` or from `_genesys.ixn.interactions[system.InteractionID].tenantid` (if available). See the Update Contact, Identify Contact, Create Interaction, or Render Message block for more information.
- **system.TenantName**—The current Tenant name. It is initialized from `_genesys.session.tenant`.
- **system.LastTargetComponentSelected**—Target to which the interaction was routed definitively. See the Target Component Selected property of the Target block.
- **system.LastTargetObjectSelected**—High-level target to which the interaction was routed definitively. See the Target Object Selected property of the Target block.
- **system.LastTargetSelected**—DN and Switch name to which the interaction was routed definitively. See the Target Selected property of the Target block.
- **system.LastVirtualQueueSelected**—The Alias of the Virtual Queue specified in the target list where the interaction was routed. See the Virtual Queue Selected property of the Target block.
- **system.LastSubmitRequestId**—RequestId value of the last `<queue:submit>` execution. This variable is automatically updated when a successful (`queue.submit.done`) or unsuccessful (`error.queue.submit`) event is received. `<queue:submit>` is generated when using Target or RouteInteraction blocks. `<queue:submit>` may also be used if using SCXMLState or BeginParallel blocks.

**Operational Parameter Management (OPM):** These parameters are defined and provisioned in Genesys Administrator Extension (GAX)

- **system.OPM**—Used by the **OPM Block** (**App\_OPM** is used in callflow diagrams).
- **system.ThisDN**— Initially set to the same value as `system.DNIS`. The value is updated by the interaction party state changed event handler (see IPD/Events property below) to the value of `focusdeviceid`. This variable becomes the default value for properties: ForceRoute/From, SingleStepTransfer/From, Target/From.
- **system.ParentInteractionID**— In case of Transfer scenario, this variable is assigned the ID of the parent interaction of the current interaction being processed.
- **system.OriginatingSession**— In case of context passing (see 'Pass Context' property description above), this variable holds the context of the originating session.

**Outbound Contact Server (OCS) variables used by Outbound blocks:**

- **system.OCS\_RecordURI**— Its default value is set when the application starts executing from data passed into the application by the GVP or Orchestration platform. For workflows (SCXML), it is initialized from the userdata key "GSW\_RECORD\_URI". For callflows (VXML), `session.com.genesyslab.userdata.GSW_RECORD_URI` is used. This variable points to the 'current' record as determined by OCS and is provided to the application as a convenient way to communicate actions back to OCS for the current record.

- **system.OCS\_URI**— Holds the OCS resource path in the format "http(s)://<ocs host>:<ocs port>". Its default value is deduced from OCS\_Record\_URI. The application can change this variable's value to use a different OCS application for all Outbound blocks in the workflow. Any downstream blocks will use the new value.
- **system.OCS\_Record**— Holds the Record Handle deduced from the value of OCS\_Record\_URI.

## Upgrading from Composer 8.1.1 or earlier

The system variables have been renamed in Composer 8.1.2 to improve the usability and to support new features. When upgrading a workflow that was initially developed with Composer 8.1.1 or earlier, the old set of system variables is kept, in addition to the new one, to ensure compatibility as some of those variables might be used in the application (for block properties or even in ECMA script code). However, users are now encouraged to use the variables that are "system." prefixed.

Composer 8.1.1	Composer 8.1.2
ANI	system.ANI
App_Language	system.language
App_Last_Error_Description	system.LastErrorDescription
App_Last_Error_Event	system.LastErrorEvent
App_Last_Error_Event_Name	system.LastErrorEventName
App_Last_Submit_Request_Id	system.LastSubmitRequestId
App_Last_Target_Component_Selected	system.LastTargetComponentSelected
App_Last_Target_Object_Selected	system.LastTargetObjectSelected
App_Last_Target_Selected	system.LastTargetSelected
App_Last_VirtualQ_Selected	system.LastVirtualQueueSelected
App_RelativePathURL	system.RelativePathURL
App_StartEvent	system.StartEvent
App_Terminate_Ixn_On_Exit	system.TerminateIxnOnExit
CallID	system.CallID
COMPOSER_WSSTUBBING	system.WebServiceStubbing
DNIS	system.DNIS
InteractionID	system.InteractionID
Tenant_Name	system.TenantName
TenantID	system.TenantID

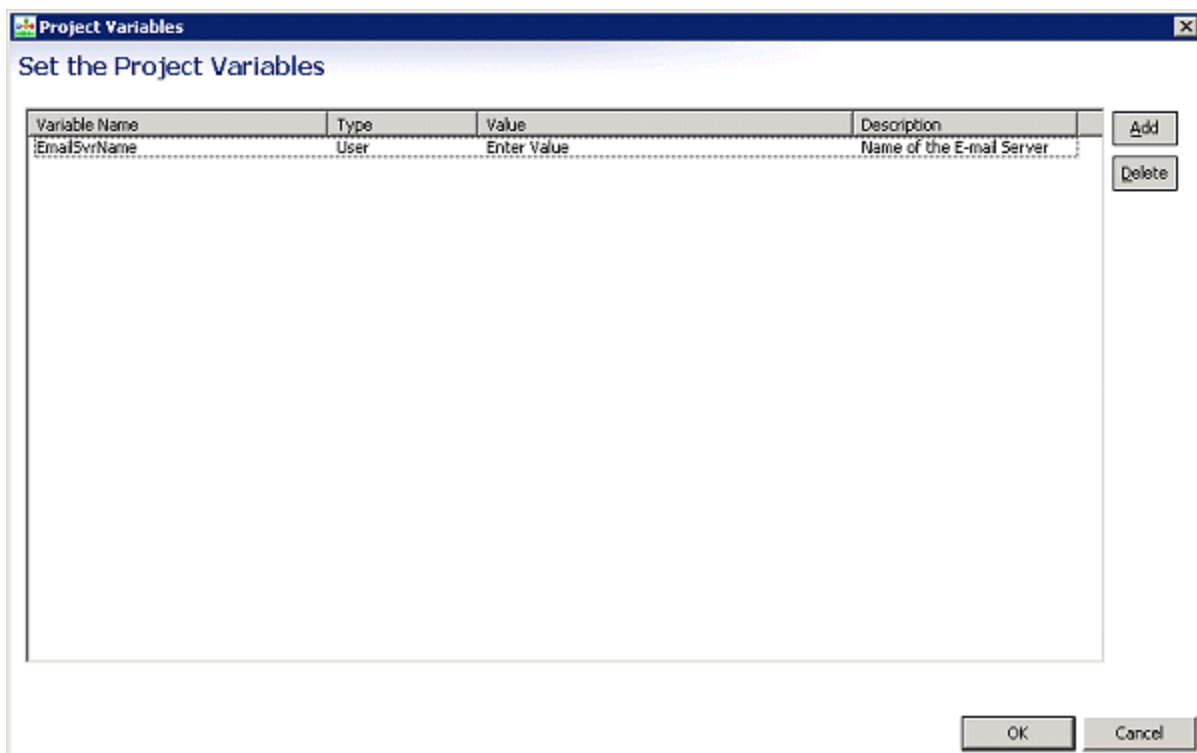
## Project Variables

Project variables encompass all the workflows in a Project. These types of variables are defined in the data model of the [interaction process diagram](#) (IPD) SCXML application and so are required to be accessed by prefixing the name of a project variable with "\_data.". For example, if you define a project variable named var1, you would access it as \_data.var1. Project variables are also accessible in [Expression Builder](#). Select Project Variables and then Variables under the [Data Subcategory](#).

Use Project variables when information needs to be shared across workflows in an IPD. For example, if you want to get the e-mail address in one workflow and would like to create and send out an email in another workflow present in the same project. Genesys suggests defining at least one appropriately named Project variable like varProjectXYZ. Any properties that accept a variable will show this variable prefixed with `_data` in their list.

To define a project variable in **Composer Design** perspective:

1. Click the **default.ixprocess** tab (or the tab for a **renamed IPD**) to bring it into focus (tab is highlighted).
2. Click the **Access Project Variables toolbar button**. This opens the Project Variables dialog box. An example dialog box with one entry is shown below.



1. Click **Add**. The variable **Name**, **Type**, **Value**, and **Description** fields become editable.
2. Name the variable.
3. Specify an initial value if appropriate.
4. Describe the variable.
5. Click **OK**.

## Internal Variables Naming

Starting with 8.1.1, Composer changes its naming policy for internal variables, which are variables that do not appear in any Variable edition dialog. They can be seen only in the generated SCXML



code. Composer uses those variables to temporarily store data during an application execution.

Most Composer users will not be affected by this change. However, it is possible that some advanced users may have written applications that use those variables although they were not available "out-of-the-box." In such cases, those users will need to upgrade their application to use the variables with the new names.

### Example

In 8.1.0, for a DB Data block named DbDataBlock Composer could declare in the SCXML application the variables:

App\_DbDataBlock, App\_DbDataBlockDBResult, App\_DbDataBlockDBResultColumnNames,  
App\_DbDataBlock\_cursor, App\_DbDataBlock\_mapping

In 8.1.1, for a DbData block named DbDataBlock Composer could declare in the SCXML application only one variable named App\_DbDataBlock.

Various properties of this variable will be used, such as App\_DbDataBlock['requestid'],  
App\_DbDataBlock['data'], App\_DbDataBlock['DBResult'],  
App\_DbDataBlock['DBResultColumnNames'], App\_DbDataBlock['cursor'],  
App\_DbDataBlock['mapping']