Designer User's Guide

Variables

4/11/2021

Contents

- 
- Administrator

Learn how to use variables in Designer.

Related pages:

-
-

You can use two types of variables in Designer:

- User Variables - These are variables that you create. You can use these variables throughout the application and in all phases.
- System Variables - These variables are created with the application and cannot be deleted.

Tips

Variable names must be alphanumeric, but not start with a numeric character. For example:

- Valid variable names = **a**`bcdef123` or **c**`123badef`
- Invalid variable names = **1**`23abcdef` or **3**`abcdef21`

Variable values may be:

- ECMAScript objects, such as `Date()`.
- Valid ECMAScript expressions. Do not add an ending semi-colon (;) as typically required by ECMAScript.
- Simple values, such as numeric or string.
  - If the value is a string, it must be surrounded by single quotes (for example, `'value'`). If the value also uses a single quote, you can use a backslash to escape the quote character (for example `'Joe\'s Pizza'`).

> **Important**
> The block properties page will indicate if single quotes are required.

User Variables

You can add user variables in the **Initialize** phase. You can assign initial values to these variables in the **Initialize** phase, or by setting values in an **Assign** block in the **Initialize** phase.

You can also assign a system variable as the default value of a user variable. For example, you might assign the system variable **DNIS** to a user variable you have created. (If the system variable does not have a value at the time of the call, the default values are used.) This is also supported for Self Service Shared Modules.

Important

User variables may not be initialized correctly if their value is set to one or more system variables in the **Initialize** phase itself. This phase should be used to declare variables, but their values should be assigned later using an **Assign** block if the value or the value expression involves a system variable.

Warning

You can delete a variable even if it is required by the application. Designer validates the application when you click **Publish**, at which time it checks for deleted variables.

Securing Variables

Warning

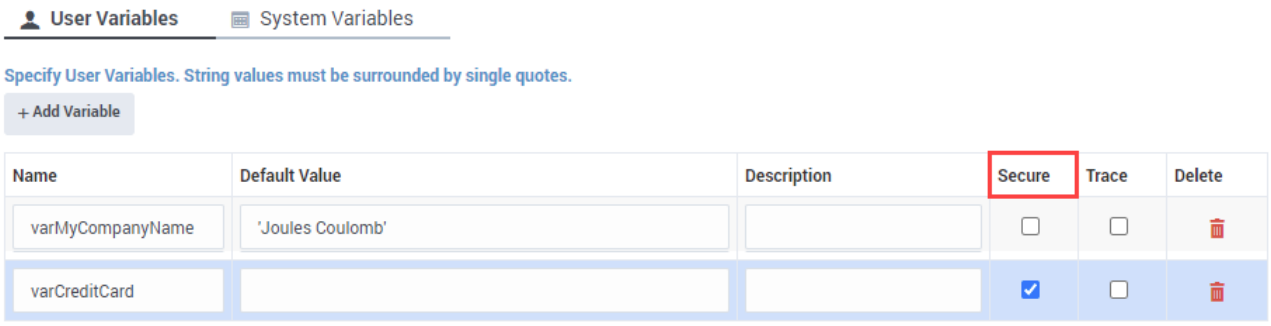
Variable values can be captured in a variety of data sources when Designer applications run on the Genesys platform. **To prevent the values of variables from being exposed as plain text in Designer and the Genesys platform, they must be marked as Secure.**

If a variable contains sensitive or personally-identifiable information (PII), you can enable the **Secure** check box to prevent the value from being logged or recorded as plain text.

Secure variables function as follows:

- Secure variable values are not logged in application logs.
- When you use secure variables to store the results of a user input, the user input is masked in platform logs (i.e. Media Control Platform).
- When you use secure variables to play back prompts, the prompt message is masked in platform logs.
- Secure variables cannot be selected in blocks that record reporting information, such as Call Data, Activity, and Milestone blocks.

- Secure variables are not reported in Analytics.



If you secure variables in an application that has already been published, you'll need to re-publish the application for the new settings to take effect.

System Variables

The **Initialize** phase has a second tab that lists system variables — these variables are created with the application and cannot be deleted.

Most system variables are initialized when the application starts and can be used throughout the application, such as the **Last Milestone** variable. When your application starts, the initial value of **Last Milestone** is an empty string. While your application runs, the **Last Milestone** value is set to the last milestone that your application reaches.

> ### Important
> Do not update system variables in the Assisted Service phase while an asynchronous Start Treatment is running. Instead, update system variables *before* the Start Treatment starts or within the Self Service treatment itself.

The following system variables are available:

Variable Name	Description
DNIS	Specifies the dialed number.
ANI	The number associated with the calling party.
MaxTime	Maximum time (in minutes) to keep this session alive.
Timezone	The timezone used for this application, unless this value is overridden in other blocks.
Language	The default language for this application that is used for announcements.
AppLanguageName	The name of the default language for this application that is used for announcements.

Variable Name	Description
RoutingSkills	A set of skills that might be specified in some blocks, such as **Menu Option** child blocks, that determine how the call is routed. For example, if you select a Skill in the **Call Handling** tab of a **Menu Option** block, this selection is stored in the **RoutingSkills** variable. Then, in a subsequent **Route Call** block, you can enable the **Use system variables RoutingSkills and RoutingVirtualQueue set already in Menu Options** check box to use the value of the **RoutingSkills** variable.
RoutingVirtualQueue	A virtual queue that might be specified in some blocks, such as **Menu Option** child blocks, that is used for routing unless a different queue is specified in Routing blocks. For example, if you select a Virtual Queue in the **Call Handling** tab of a **Menu Option** block, this selection is stored in the **RoutingVirtualQueue** variable. Then, in a subsequent **Route Call** block, you can enable the **Use system variables RoutingSkills and RoutingVirtualQueue set already in Menu Options** check box to use the value of the **RoutingVirtualQueue** variable.
EstimatedWaitTime	The estimated wait time for the call to be routed to an agent.
IVRProfileName	This application's calls will be associated with the given IVRProfile for VAR reporting.
GVPTenantID	This application's calls will be associated with the given tenant for VAR reporting.
SelectedTarget	This is the DN and the switch name of the target to which the interaction was routed or should be routed to definitively. The target format is `Name@SwitchName.Type`.
SelectedVirtualQueue	The virtual queue that was selected.
SelectedComponent	The agent-level target to which the interaction was routed or should be routed to definitively. If the target selected for routing is of type **Agent**, **Place**, **Queue**, or **Routing Point**, this variable contains the target. If the desired target type is **Agent Group**, **Place Group**, or **Queue Group**, the function returns the agent, place, or queue from the corresponding group to which the interaction was sent. The target format is `Name@StatServerName.Type`.
SelectedTargetObject	This is the high-level target to which the interaction was routed or should be routed to definitively. If a skill expression is used, the function returns: `?:SkillExpression@statserver.GA` or `?GroupName:SkillExpression@statserver.GA`. The target format is `Name@StatServerName.Type`.
SelectedAgent	This is the Employee ID of the agent to which the interaction was routed.

<table>
<tr><th>Variable Name</th><th>Description</th></tr>
<tr><td>Access</td><td>(Optional) When present, this is an ECMAScript object that represents a switch access code. The table below show its properties and the corresponding switch access code fields:<table><tr><th>Access property</th><th>Switch access code field</th></tr><tr><td>prefix</td><td>Code</td></tr><tr><td>rtype</td><td>Route Type</td></tr><tr><td>destination</td><td>Destination Source</td></tr><tr><td>location</td><td>Location Source</td></tr><tr><td>dnis</td><td>DNIS Source</td></tr></table></td></tr>
<tr><td>CustomerSegment</td><td>The segment to which the customer belongs, based on information that the customer has provided.</td></tr>
<tr><td>CustomerId</td><td>A unique identifier for the customer, based on information that the customer has provided.</td></tr>
<tr><td>EnableSSRecording</td><td>Enable call recording in the Self Service phase.</td></tr>
<tr><td>CallbackReporting</td><td>An object containing key-value pairs for callback reporting.</td></tr>
<tr><td>PositionInQueue</td><td>The call's position in queue while waiting to be routed to an agent.</td></tr>
<tr><td>AppCountry</td><td>The country code for this call (can be specified by the application).</td></tr>
<tr><td>AppRegion</td><td>The region for this call (can be specified by the application).</td></tr>
<tr><td>AppCallType</td><td>The type of call (can be specified by the application).</td></tr>
<tr><td>AppUserDisposition</td><td>A custom disposition that the application can use to specify a user-specific outcome.</td></tr>
<tr><td>AppUserDispositionCategory</td><td>A custom disposition category that the application can use to categorize user-specific outcomes.</td></tr>
<tr><td>AppDeflectionMessage</td><td>The application can use this variable to track deflections by specifying the message played when a caller disconnected their call.</td></tr>
<tr><td>AppLastMilestone</td><td>The last milestone that the application achieved.</td></tr>
<tr><td>AppStrikeoutMilestone</td><td>The last milestone that the application achieved before strikeout.</td></tr>
<tr><td>AppBailoutMilestone</td><td>The last milestone that the application achieved before the caller bailed out to an agent.</td></tr>
<tr><td>AppDeflectionMilestone</td><td>The last milestone that the application achieved before the caller was deflected.</td></tr>
<tr><td>ScriptID</td><td>The ScriptID as reported by the routing engine.</td></tr>
<tr><td>AppSelfHelpedMilestone</td><td>Used to contain a self help milestone.</td></tr>
</table>

Variable Name	Description
SdrTraceLevel	Enables users to set the recording level. This variable accepts the following values: • `100` — Debug level and up. Currently, there are no Debug messages. • `200` — Standard level and up. This setting shows the existing blocks array in the SDR. • `300` — Important level and up. This setting filters out all blocks except those containing data that can change from call to call (such as the **User Input** block).
AppSessionType	The type of the session. The default value is *inbound'* for inbound calls. Survey applications must use the value **survey**.
EnableRouteCallRecording	Set to `true` or `false` to enable or disable call recording for routed calls in the **Assisted Service** phase. Leave blank to use platform defaults.
GmsCallbackServiceName	The GMS Callback Service name.
GmsCallbackServiceID	The unique identifier that GMS assigns to a scheduled callback.
survey_sOffer	Set by the Setup Survey block to specify if a survey was offered, setup, or rejected.
survey_iAgentScore	Holds the user satisfaction score for the agent, if this question is asked by the survey.
survey_iCompanyScore	Holds the user satisfaction score for the company, if this question is asked by the survey.
survey_iCallScore	Holds the user satisfaction score for the overall call, if this question is asked by the survey.
survey_iProductScore	Holds the user satisfaction score for the product, if this question is asked by the survey.
survey_iRecommendScore	Holds the user's rating score (on a scale of 0-10) of the company, product, or service. Used to calculate Net Promoter Score (NPS).
ApplicationRevisionSerialID	A read-only variable that increments by 1 each time an application is revised.
ApplicationPath	The absolute path to the application.
DefaultPartition	The default partition used to provide access control in GIR. This variable can be overridden by settings in the **Record** block.
AutoStopInteraction	(Digital only) Specifies whether or not the interaction is to be automatically terminated when the session ends. The default value is `auto` (to automatically terminate a *chat* interaction if it exceeds 20 application runs or exceeds the specified expiration time); other valid settings are `true` or `false`.

Variable Name	Description
ChatEntryPoint	(Digital only) Holds the point of entry for a chat interaction. Can be used in application logic at runtime to provide alternative processing or to facilitate the use of parallel testing environments.
TreatmentIterationCount	Keeps track of how many times a treatment has been executed.
AgentsTotalSize	The total number of agents that are possibly available. For example, the total number of agents in a specified Agent Group.
AgentsLoginSize	The number of agents that are actually logged in.
enableSSML	Set as `true` to enable the interpretation of Speech Synthesis Markup Language (SSML) tags in TTS (Text-to-Speech) prompts.
removeSSMLInChat	Set as `true` to remove SSML tags from chat messages if an omnichannel application is using the same messages for both voice and chat channels.

VAR Metrics

> **Important**
> VAR action IDs are stripped of spaces and pipe characters (`|`). This includes implicit actions that are generated when a caller enters a shared module.

Use the **IVRProfileName** variable (User Data Key: **gsw-ivr-profile-name**) to associate the application VAR metrics with an IVR Profile. Use a value of `auto` to auto-detect the IVR Profile.

Use the **GVPTenantID** variable (User Data Key: **gvp-tenant-id**) to associate the application VAR metrics with a tenant. Designer attaches the value to user data. Use a value of `auto` to auto-detect the tenant.

These variables are listed in the properties of blocks once they are defined.

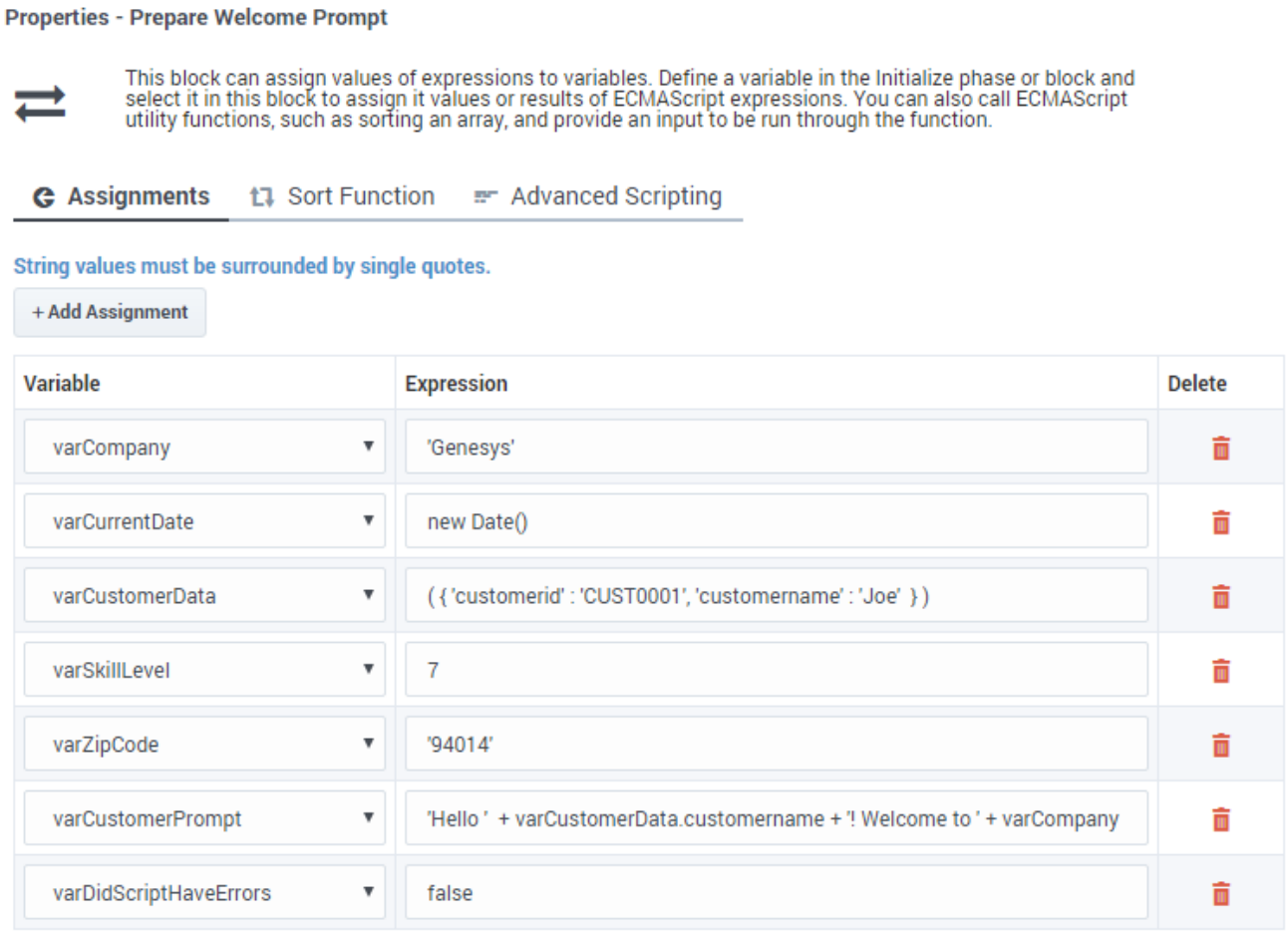
Assigning Values to Variables

Designer lets you assign values to variables in different ways. These examples show a few of the methods you can use to assign different types of values to a variable, including a JSON value.

Example 1: Simple Assignment

The easiest (and recommended) way is to assign a value to a variable using the **Assignments** tab on the Assign Variables block.

Click **Add Assignment** to add an assignment slot to the block, then choose a variable from the **Variable** column. For the **Expression**, you can use the name of another variable whose value should be copied in to the **Variable** column, a literal value (for example, "Sales Channel"), or an expression whose value should be calculated first and the results assigned to the **Variable**.



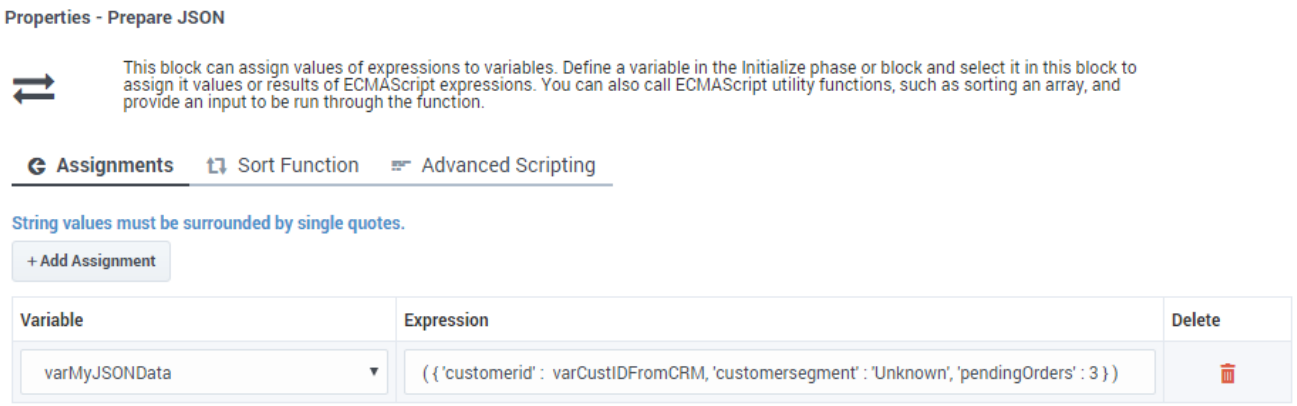
You must use single quotes (' ') when specifying string values, but you can assign numeric values without quotes. Note that the *varZipCode* above is a string data type (the single quotes tell JavaScript to treat it as a string), but it contains only numbers. It's important to remember that JavaScript treats string and numeric data types differently. For example, 1 + 2 = 3, but '1' + 2 = '12'.

JSON data (for example, *varCustomerData*) is typically retrieved from an external data source, but you can also form a JSON string in the application. JSON strings must be enclosed in brackets ( ) and should follow the rules and syntax for JSON strings as defined by JavaScript. Note also that variables can easily be used to form part of the JSON string (as represented by *varCustIDFromCRM*, in the example shown below).

The *varCustomerPrompt* above shows a simple string expression, with the different string segments linked together by a **+**. If used in a **Play Message** block, it will play "Hello Joe! Welcome to Genesys." It accesses a property of the *varCustomerData* object using the "." notation and combines it with the welcome message.

> **Important**
>
> Although the terms ECMAScript and JavaScript are used interchangeably throughout this Help, Designer technically supports ECMAScript and does not support JavaScript functions that are typically used for web-browser based applications, such as pop-up windows, alerts, and so on.

Here is another example of how you could build a JSON expression. It contains mostly hard-coded strings, but also uses a variable to form part of the JSON string:



Example 2: Advanced Scripting

If your application requires you to go beyond simple assignments and use JavaScript constructs like loops or multiple nested conditions, you can use the **Advanced Scripting** tab, which allows you to compose valid ECMAScript or JavaScript.

> **Important**
>
> Advanced Scripting is an optional feature and might not be enabled on your system. To enable this functionality, contact Genesys.

To use this feature, you need a basic level of familiarity and understanding of ECMAScript syntax and rules. Any errors in the script can cause erratic behavior, so test your changes to make sure that your script works correctly.

> **Warning**
>
> Use caution! Designer can check your script for syntax errors, but cannot validate it nor check for runtime errors that might occur when the script runs during a call.

In this example, the script sets the variable *varOrdersPrompt* to "3 Laptop bags, 2 Phone chargers, 1 Super rare fish". Here's how it works:

The sample script below first initializes JSON data in `varOrderDetails` so that it becomes an array of three JSON objects. Each JSON object has properties — item, quantity, backordered. The script then proceeds to loop through orders and forms a string to play back to the caller to notify them of their order status.

Note that this script uses variables in two scopes:

- A scope exclusive or local to this script itself ("`i`"). This variable remains available only while this script runs, and then it disappears.
- Top level variables that were defined in the **Initialize** phase — these remain available throughout this application flow, but not in any modules this application calls (such as `varOrdersPrompt`).