Global Variables

Introduction

Global variables are one of the types of global data elements available in Jitterbit Harmony. The other types of global data elements—project variables, Jitterbit variables, and filename keywords—are also available globally throughout a project. They are set differently, as described in Project Variables, Jitterbit Variables, and Filename Keywords, respectively. However, as all global data elements share the same namespace, their names must be unique when compared against all other global data elements.

Global variables are first declared in an operation, after which they become available to be referenced in the same or downstream operations and scripts. Downstream operations or scripts are those that are linked within an operation chain using operation actions. Downstream operations can be within the same or downstream workflows.

Global variables can also be used as a source or target within the operation as described in Variable Read and Variable Write.

You may want to use global variables if your use case involves sharing information with subsequent parts of an operation chain, such as in these examples:

- Using values created in one transformation in a later transformation. For example, a session ID (returned from a login web service) may be required when calling subsequent web services for authentication.
- Using a value created in one part of a transformation at a later stage of that same transformation. For example, a record number may be initialized and incremented for each record inserted into a target to identify its item number.
- Using a value returned in one transformation in the configuration of components in subsequent operations. For example, the URL setting returned by one transformation may be used to set the web service URL of a subsequent web service call.

Global variables pass through chained operations. These include operations that are linked to a prior operation within the operation chain using "on success" or "on failure" operation actions, as well as those that are linked through the `RunOperation()` function. Global variables can also be used within the same transformation.

Display of Global Variables

Defined global variables are displayed in several places:

- In the project pane, global variables are displayed in the **Components** tab in the **Global Variables** category. From here you can see if the global variable is referenced elsewhere in the project and view dependencies.
- In the script editor, global variables are displayed in the component palette on the right within the **Variables** tab in the **Global Variables** sub-tab. This location provides easy access for inserting global variable references in scripts, including within transformation scripts in script mode.
- In transformation mapping mode, global variables are displayed within the **Variables** tab on the left in the **Global Variables** category. This location provides easy access for inserting global variable references during transformation mapping in mapping mode.
- In endpoint configuration screens, global variables can be accessed and used in any fields that have a variable icon  . As an alternative to selecting a global variable, you can manually enter the variable reference using the standard Jitterbit Script square bracket syntax.

Creating and Updating Global Variables

Global variables are created or updated using either Jitterbit Script (in scripts and transformations) or JavaScript (only in scripts created as a project component). See Script Types and Creation for information on creating the different types of scripts. Global variables configured in a Variable connection are created or updated automatically (see Variable Connection).

Global Variable Names

Global variable names can be composed from alphanumeric characters (the letters a-z and A-Z, and the digits 0-9), periods (.), and underscores (_). (Other characters, such as hyphens, are not recommended.) Global variable names are case-insensitive; a variable called `GlobalVar` is treated the same as `global var`.

On This Page

Related Articles

- Component Palette
- General Functions
- JavaScript
- Jitterbit Script
- Mapping Variables
- Operation Actions
- Project Pane
- Project Variables
- Script Testing
- Script Types and Creation
- Variable Connection
- Variable Read Activity
- Variable Write Activity

It is a good practice to use periods or underscores when defining global variables so that they are easy to find. For example, a global variable created in a Jitterbit Script named `org.account.filename` begins with `org`, followed by `account`, etc. effectively organizing it when in a list among other similarly constructed global variables. Note that for global variables created in JavaScript (or for Jitterbit Script global variables that might later be used in JavaScript) we recommend using underscores instead of periods. Using periods in user-defined global variables causes issues at runtime. Further information is provided in the JavaScript subsection below.

Jitterbit Script

In Jitterbit Script used within scripts and transformations, a global variable can be defined by beginning with a dollar sign $ or by calling the `Set()` function:

- **`$:`** Using the dollar sign $ syntax, `$ServerURL=URL` creates or updates a global variable called `ServerURL` with the value of `URL` (the name of another variable or the name of a field in a transformation).
- **`Set:`** Using the `Set()` function, `Set("ServerURL", URL)` creates or updates a global variable called `ServerURL` with the value of `URL` (the name of another variable or the name of a field in a transformation).

JavaScript

In JavaScript used within scripts created as a project component, the syntax used for setting a global variable depends on whether the global variable name contains a period:

- **Name does not include a period (recommended):** A global variable that does not contain a period in its name can be created or updated using `var $`*`name`* or updated simply by using $*name* without `var`:
  - **`var $:`** The expression `var $ServerURL="https://www.example.com/"` creates or updates a global variable `ServerURL` with a string value of `https://www.example.com/`. A new global variable must precede the $ with `var`.
  - **`$:`** The expression `$ServerURL="https://www.example.com/"` updates the same global variable `ServerURL` with the same string URL. This works only for global variables that already exist.
- **Name includes a period (recommended only for Jitterbit variables and JavaScript object values):** A variable that contains periods in its name can be updated or retrieved in JavaScript only with the `Jitterbit.SetVar` and `Jitterbit.GetVar` functions. As these functions are not intended for user-defined global variables, see Jitterbit Variables for more information.

> **WARNING:** The JavaScript `Jitterbit.SetVar` and `Jitterbit.GetVar` functions are designed specifically to access the predefined Jitterbit variables. They are ***not*** to be used to access user-defined global variables.

Variable Endpoint

During configuration of a Variable connection, a global variable can be created or updated automatically.

That is, if during configuration you provide the name of a variable that does not yet exist, a global variable is created automatically. If you provide the name of a global variable that already exists, it becomes associated with the Variable endpoint.

For details on creating or updating a global variable using this method, see Variable Connection.

Retrieving Global Variables in Scripts or Transformations

The value of a global variable can be returned using either Jitterbit Script (in scripts or transformations) or using JavaScript (only in scripts created as a project component).

> **WARNING:** There is a known issue that mapping variables with periods in the variable name doesn't work correctly, resulting in an incorrect target field mapping in the transformation. Until this issue is resolved, rename the variable by replacing any periods with underscores or removing periods another way.

Jitterbit Script

In scripts and transformations, you can begin with a dollar sign $ or use the `Get()` function to retrieve the value of a global variable:

- **`$:`** Prefixed with a dollar sign $, the code example `$serverURL` retrieves the value (or field in a transformation) of the global variable called `"serverURL"`.

Related Topics

- Cloud Studio
- Connectors
- Operation Settings
- Operations
- Scripts
- Transformations
- Variable
- Variables

Last updated: Aug 21, 2020

- **`Get:`** Using the `Get()` function, the code example `Get("serverURL")` returns the same value (or field in a transformation).

In scripts and transformations, existing global variables are also displayed in the **Variables** tab of the script component palette inside the **Global Variables** sub-tab. See Adding a Global Variable to a Script below.

JavaScript

In JavaScript scripts within an operation, the syntax used for retrieving the value of a global variable depends on whether the global variable name contains a period.

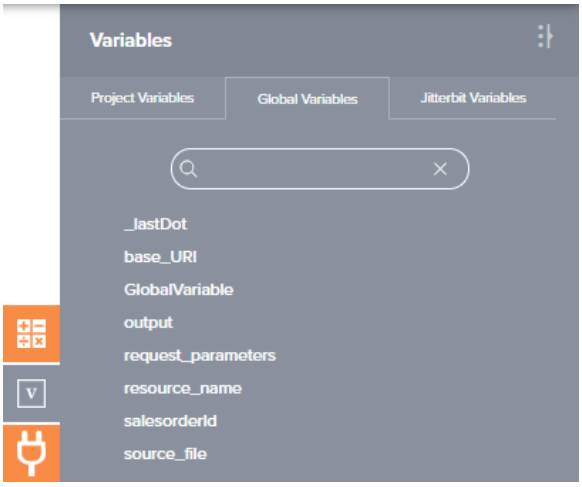
- **Name does not include a period (recommended):** The value of a global variable that does not contain a period in its name can be retrieved by beginning with a dollar sign $:
  `$:` Prefixed with a dollar sign $, the expression `$ServerURL` retrieves the value of the global variable called `ServerURL`.
- **Name with periods (recommended only for Jitterbit variables and JavaScript object values):** A variable that contains a period in its name can be updated or retrieved in JavaScript only with the `Jitterbit.SetVar` and `Jitterbit.GetVar` functions. As these functions are not intended for user-defined global variables, see the section on Jitterbit variables for more information.

> **WARNING:** The JavaScript `Jitterbit.SetVar` and `Jitterbit.GetVar` functions are designed specifically to access the predefined Jitterbit variables. They are ***not*** to be used to access user-defined global variables.
>
> In JavaScript, ***do not*** mix and match your usage of `SetVar` (and `GetVar`) with $-prefixing when setting (and retrieving) a variable. Use only one syntax. Mixing the two different syntaxes for the same variable can cause issues at runtime.
>
> If a JavaScript fails, any changes made to the value of a global variable are lost. Only if the JavaScript successfully completes are modified global variable values available outside the script.

In scripts and transformations, existing global variables will also be displayed in the **Variables** tab of the script component palette inside the **Global Variables** sub-tab. See Adding a Global Variable to a Script below.

Adding a Global Variable to a Script

In either Jitterbit Script or JavaScript used within scripts created within an operation, existing global variables are also displayed in the **Variables** tab of the component palette inside the **Global Variables** sub-tab:



Add a variable to a script using one of these methods:

- Drag the variable from the palette to the script. The appropriate syntax for the script language is inserted.
- Begin typing the variable name and then press `Control+Space` to display a list of autocomplete suggestions. Select a variable to insert the appropriate script language syntax.
- Manually enter the appropriate syntax for Jitterbit Script or JavaScript.

Using Global Variables in Configuration Screens

During configuration of various project components, including endpoint configuration using connectors, you can use global variables in any fields that have a variable icon V. Variables can be used in fields along with other input, including with other variables or keywords. These actions are covered below:

- Selecting a Global Variable
- Toggling Formats Between Pill and Text
- Defining a Default Value
- Removing a Variable

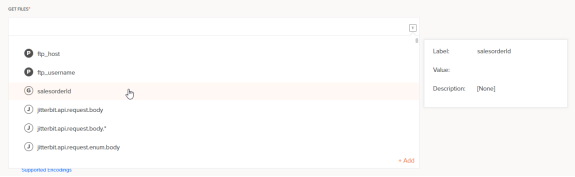
Selecting a Global Variable

To access global variables, you can either click the variable icon V or enter an open square bracket [to display a list of variables and keywords (if available for the current field).

Within the list, each variable or keyword type is indicated by the icon next to its name:

- F for filename keyword
- G for global variable
- P for project variable
- J for Jitterbit variable

In the list, hover over the variable name to preview information about it:



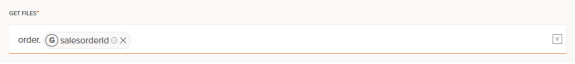
The **Description** in the information popup is *[None]* for a global variable. A **Value** is populated only if the global variable has a default value specified in this configuration field.

The icon's background is solid G if the global variable has a default value specified in this configuration field, or empty G if it does not. The icon's background is unaffected if the same global variable has a value specified in another configuration field, as only a default value that is specified in the current field is used.

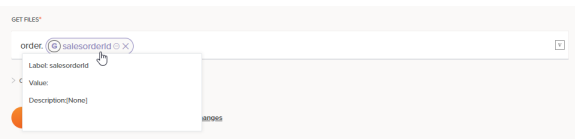
> **WARNING:** If a global variable obtains its value in a script that is upstream of the component, its *evaluated* value overrides the default value specified in the configuration field. Note that evaluated values that are obtained in a script are not reflected in the **Value** in the information popup.

> **CAUTION:** It is currently a known issue that the global variable icon's background is solid G if the global variable is set in a script anywhere in the project. Global variables set in an upstream script use the evaluated value at runtime and are therefore not considered to be default values.

Select a variable to add to the field at the location of your cursor, anywhere in the string. The variable is displayed in a pill format similar to that shown below:

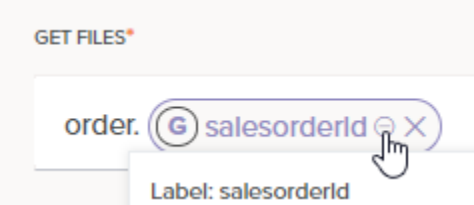


To review information about the variable, hover over the variable pill:



Toggling Formats Between Pill and Text

To change the default variable pill format to text format, click the collapse icon ⊖:



This toggles the display of the pill format to a text format, with the variable name enclosed within square brackets `[ ]`:

In text format, you can define a default value.

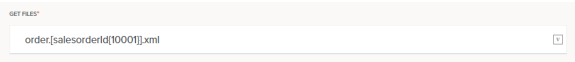
To change from a text format back to pill format, change focus off of the field, such as by clicking into another configurable field. The field input automatically returns to the default pill format.

Defining a Default Value

At operation runtime, a default value defined within a configuration field, as shown below, is used only if the variable value has not been set during runtime. This is different during script testing, where—as any upstream script values will not yet have been instantiated—any default value defined within a configuration field is used instead.

To define a default value within a configuration field, first toggle to text format.

Immediately following the global variable name within the square brackets `[ ]`, specify the default value within curly brackets `{ }`:



When default values are specified, the global variable value is used if it is defined, else the default value is used.

TIP: When using a global variable in a WHERE clause, such as in a database or Salesforce query, you can specify a default value so that script testing is possible. Otherwise, as a global variable obtains its value at runtime, during testing the syntax may be invalid if no default value has been specified.

If you don't want the global variable to be interpreted, use a backslash `\` to escape the square bracket set `[ ]`.

For example, the following does not use the value of `serverURL` even if it is defined, but instead always uses `http://server/index.asp`:

```
\[serverURL{http://server/index.asp}]
```

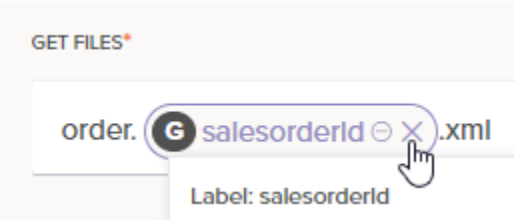
CAUTION: Within file paths that contain backslashes, a single backslash is interpreted as initiating an escape sequence if it is followed by a square bracket set `[ ]`.

Additional backslashes can be used to achieve the desired result. For example, `\\server\share\\[Directory{testing}]` is interpreted as `\\server\share\testing` if the variable `Directory` is not defined, else `\\server\share\"value of Directory"` is used.

To avoid this issue, convert file paths to URL format (e.g. `C:/directory/path`).

Removing a Variable

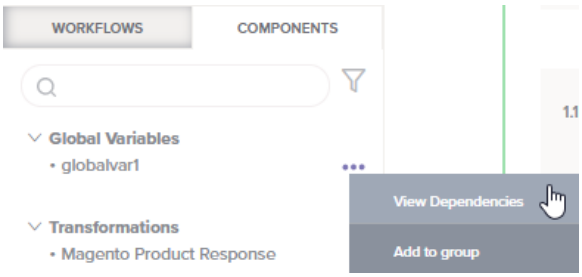
To remove the variable, click the remove icon ✕:



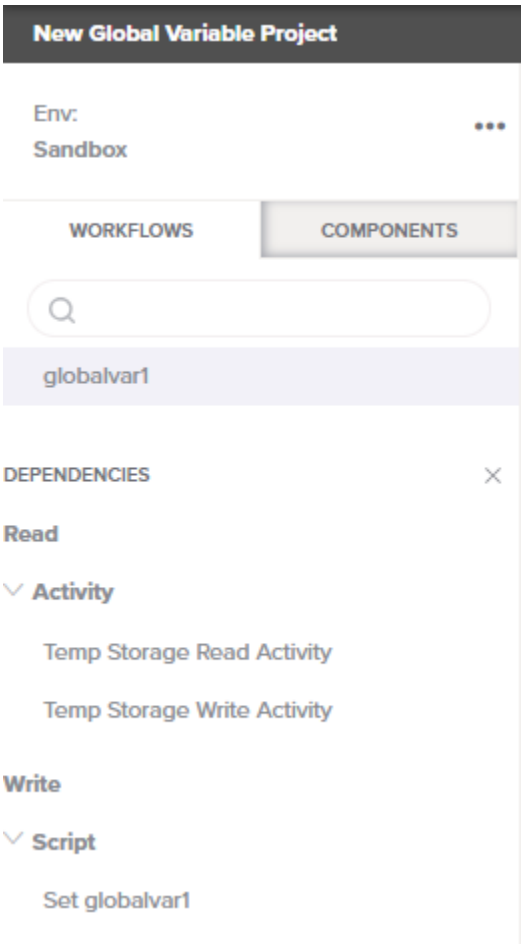
Viewing Global Variable Dependencies

The dependencies of a global variable are other components that depend on the variable because they are either reading from—or writing to—the global variable. Once a global variable has been created and used in a project, you can view these dependencies.

After a global variable is created, menu actions for that global variable are accessible from the project pane. In the **Components** tab, hover over the global variable name and click the actions menu icon ••• to open the actions menu. From the menu, select **View Dependencies**:



This changes the view in the project pane to display other parts of the project that the global variable is dependent on. In dependency view, the name of the selected global variable appears along the top, below the existing search and filter capabilities. The global variable name is followed by a list of **Dependencies** that the variable is dependent on. This list is organized by categories such as **Activity** and **Script** and further classified under **Read** or **Write** to indicate the access type of a particular variable reference:



For example:

- If a script contains `$myVar1='abc'`, it is writing a value to the variable and referencing `myVar1` for write access.
- If a script contains `$myVar2=$myVar1`, it is referencing `myVar2` for write access and `myVar1` fo r read access.
- If a script contains `If(myVar1=='abc', TRUE, FALSE)`, it is referencing `myVar1` for read access.

Each category can be expanded or collapsed using the disclosure triangles ⌄ ›.

To close out of dependency view, click the close icon ×.

Global Variable How-tos

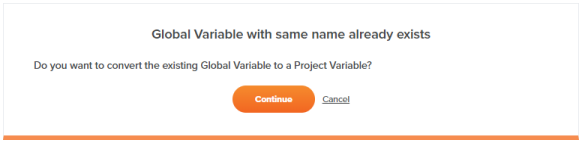
See these sections for details on using global variables in projects:

- Converting a Global Variable to a Project Variable
- Checking for Null or Undefined Values
- Setting and Accessing Array Variables

Converting a Global Variable to a Project Variable

You can convert an already-existing global variable into a project variable.

Global variables can be converted into project variables during project variable configuration. While configuring a project variable, enter the name of an existing global variable into the name field. On attempting to save, a prompt asks you to confirm that you want to convert the global variable into a project variable:



On clicking **Continue**, all references to and dependencies of the global variable are transferred to the project variable. This includes references to the former global variable within scripts, which now reference the project variable. (Keep in mind that a value for a project variable set in a script during execution of the operation chain overrides the default value specified in the project variable configuration.)

For more details on project variables, see Project Variables.

Checking for Null or Undefined Values

A global variable that has not been defined has a null value.

For example, `IsNull(Get("GlobalVariableName"))` returns true if a global variable with the name `GlobalVariableName` has not yet been defined. This can be used to check if a global variable has been defined or not.

Setting and Accessing Array Variables

You can create arrays of global variables, also known as *array variables*. See Arrays for information on setting and retrieving values from array variables.