Global Variables

Introduction

Global variables are first declared in an operation, after which they become available to be referenced in the same or downstream operations. Downstream operations may be within the same or downstream workflows, as linked with operation actions. Other types of variables that can be used globally throughout a project include project variables and Jitterbit variables, which are set differently.

You may want to use global variables if you have a use case that involves sharing information with subsequent parts of an operation chain, such as with these use cases:

- 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 purposes.
- Using values created in one part of a transformation at a later stage in that transformation. For example, a record number may be initialized and incremented for every record inserted into a target to identify its item number.
- Using values 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, for example, those that are linked to a prior operation within the same or different workflow 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.

Defined global variables are displayed in several places:

- From 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.
- From 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.
- From 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.
- From endpoint configuration screens, global variables can be accessed and used in any fields that have a variable icon V. As an alternative to selecting a global variable, you can manually enter the variable reference using the standard square bracket syntax.

Creating and Updating Global Variables

You can create and update global variables using Jitterbit Script in scripts and transformations, or using JavaScript in scripts created as a project component (see Script Types and Creation). In addition, 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 these characters: letters (a-z, A-Z), numbers (0-9), periods, and underscores. Other characters, such as hyphens, are not recommended and may cause issues. Global variable names are case-insensitive; for example, a variable called `GlobalVar` is treated the same as `globalvar`.

It can be a good idea to use periods or underscores to prefix global variables so that they are easy to look up later. For example, a global variable created in a Jitterbit Script named `org.account.filename` is first prefixed with `org`, then `account`, etc. to effectively organize it in a list among other global variables. However, note that for global variables created in JavaScript or for Jitterbit Script global variables that might later be used in JavaScript, it is recommended to use underscores instead of periods. 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, the code example `$serverURL=URL` creates or updates a global variable called "`serverURL`" with a value (or field in a transformation) called "`URL`".
- **`Set:`** Using the `Set` function, the code example `Set("serverURL",URL)` creates or updates the same global variable called "`serverURL`" with a value (or field in a transformation) called "`URL`".

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

Related Topics

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

Last updated: Dec 26, 2019

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.

WARNING: The `Jitterbit.GetVar` and `Jitterbit.SetVar` functions are designed to allow the use of variables that contain periods within the variable name. However, using periods in a variable name is not recommended. As the value will be converted to a string when the variable is set, these variables cannot be used with complex data types such as arrays, dictionaries, or JSON objects. Instead, it is recommended that you create Jitterbit variables without periods and instead use underscores in place of periods and use the standard dollar sign $ syntax described below.

- **Names without periods (recommended):** A global variable that does not contain any periods in its name can be created initially or updated using the command `var $`, or updated using a dollar sign $ without `var`.
    - **`var $`:** Using `var` and beginning with a dollar sign $, the code example `var $serverURL="https://www.example.com"` creates or updates a global variable called "`serverURL`" with a value of "`https://www.example.com`" . New global variables that are being initialized must precede the $ with `var`.
    - **`$`:** Prefixed with a dollar sign $, the code example `$serverURL="https://www.example.com"` updates the same global variable called "`serverURL`" with the same URL. This works only for global variables that are already initialized.
- **Names with periods (not recommended):** A global variable that contains periods in its name can be created initially or updated only with the `Jitterbit.SetVar` function.
    - **`Jitterbit.SetVar`:** Using `Jitterbit.SetVar`, the code example `Jitterbit.SetVar("$server.URL", "https://www.example.com")` creates or updates a global variable called "`server.URL`" with a value of "`https://www.example.com`" that will be treated as a string. Note that the dollar sign $ ***must*** be included within the variable name, or the variable will not be global to the Harmony system.

Variable Endpoint

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

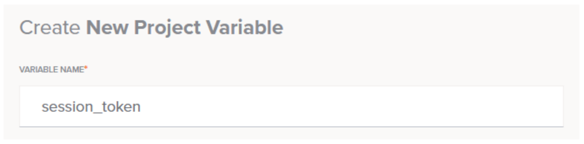
That is, if during configuration you provide the name of a variable that does not yet exist, a global variable will be created automatically. If you provide the name of a global variable that already exists, it will become associated with the variable endpoint.

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

Converting a Global Variable to a Project Variable

If you have already created a global variable that you want to be a project variable, you can convert it.

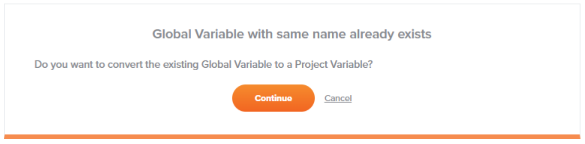
Global variables can be converted into project variables during project variable configuration. While configuring a project variable, enter the name of an existing global variable into the **Variable Name** field:


Create New Project Variable

VARIABLE NAME*

session_token

Upon clicking **Save**, a prompt will ask you to confirm that you want to convert the global variable into a project variable:


Global Variable with same name already exists

Do you want to convert the existing Global Variable to a Project Variable?

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

For more details on project variables, see Project Variables.

Using 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 where mapping variables that have a period in the variable name isn't working properly, 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`".
- **`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 will also be displayed in the **Variables** tab of the script component palette inside the **Global Variables** sub-tab:

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

- Drag the variable from the palette to the script. The appropriate syntax for the script language will be 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 syntax for Jitterbit Script.
- Manually enter the appropriate syntax for Jitterbit Script.

JavaScript

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

WARNING: The `Jitterbit.SetVar` and `Jitterbit.GetVar` functions are designed to allow the use of variables that contain periods within the variable name. However, using periods is not recommended because the value will be converted to a string when the variable is set, and thus cannot be used with complex data types such as arrays, dictionaries, and JSON objects. Instead, it is recommended to create Jitterbit variables that do not contain periods, for example by using underscores in place of periods and using the standard dollar sign $ syntax described below.

- **Names without periods:** The value of a global variable that does not contain any periods in its name can be retrieved by prefixing with a dollar sign $.
  - **`$:`** Prefixed with a dollar sign $, the code example `$serverURL` retrieves the value of the global variable "`serverURL`".
- **Names with periods:** The value of a global variable that contains periods in its name can be retrieved only with the `Jitterbit.GetVar` function.
  - **`Jitterbit.GetVar:`** Using `Jitterbit.GetVar`, the code example `Jitterbit.GetVar("$server.URL")` returns the string value of the global variable called "`server.URL`". Note that the dollar sign $ **must** be included within the variable name to read the global value from the Harmony system.

In JavaScript used within scripts created within an operation, existing global variables will also be displayed in the **Variables** tab of the component palette inside the **Global Variables** sub-tab:

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

- Drag the variable from the palette to the script. The appropriate syntax for the script language will be 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 syntax for JavaScript.

- Manually enter the appropriate syntax for JavaScript.

Using Global Variables in Configuration Screens

During configuration of various project components, including endpoint configuration using connectors, yo u can use global variables in any fields that have a variable icon V.

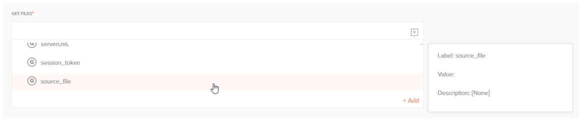
Single Variable

To use a single variable as the only value within the field, click the variable icon V to display a list of keywords and variables.

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

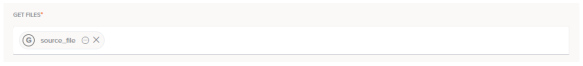
- F for filename keyword
- G for global variable
- P for project variable
- J for Jitterbit variable

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

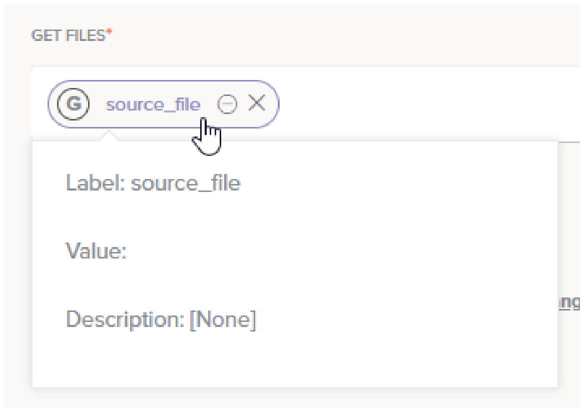


NOTE: The value for a global variable always appears to be empty, even if you have already provided a default value in an upstream script. At this time, it is not possible to add a description. More information about specifying default values is provided later on this page under Variables with Default Values.

Select a variable to replace the contents of the field. Any existing input entered within the field, if present, will be replaced with the variable. A single variable used as input in a field is displayed in a pill format similar to that shown below:



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

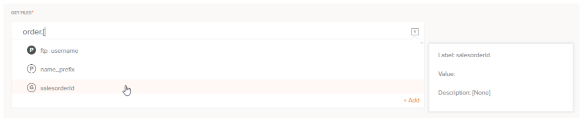


To change the pill format to text to use with additional input, with each variable displayed within square brackets [ ] , click the collapse icon ⊖.

To clear the field, click the remove icon ✕.

Variables with Additional Input

To use a variable along with other field input, including with other keywords or variables, enter an open square bracket `[` into the field. From the list, hover over the variable name to preview information about it. Select a variable to insert the variable at the location of your cursor, anywhere in the string:


GET FILES*

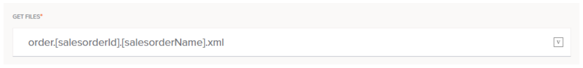
order.[

Label: salesorderId

Value:

Description: [None]

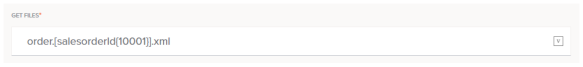
Variables entered using this method are displayed as text within square brackets `[ ]` within the field. Enter any additional input or add multiple keywords or variables using the same method described above:


GET FILES*

order.[salesorderId].[salesorderName].xml

Variables with Default Values

To define a default value, specify it within curly brackets `{ }` immediately following the global variable name, within the square brackets `[ ]`:


GET FILES*

order.[salesorderId{10001}].xml

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

TIP: When using a global variable in a WHERE clause, such as in a database or Salesforce query, we recommend specifying a default value so that script testing is possible. Otherwise, since the global variable obtains its value at runtime, the syntax may be invalidated during testing if no default value is specified.

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

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

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

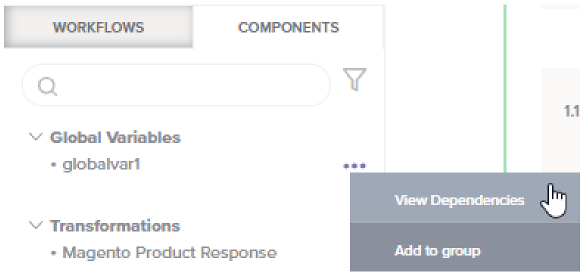
CAUTION: Within file paths that contain backslashes, a single backslash will be interpreted as initiating an escape sequence if it is followed by a square bracket set "`[`" and "`]`".

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

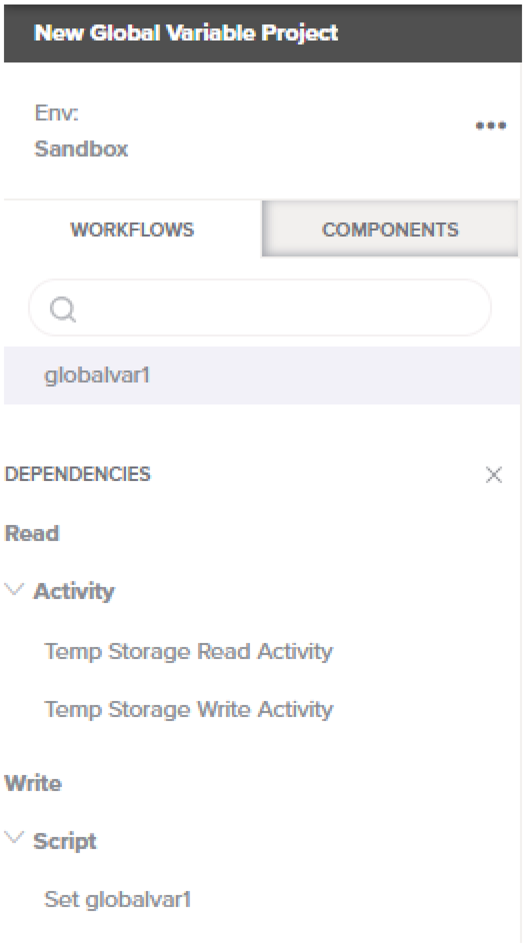
To avoid this issue, it is recommended to convert file paths to URL format (e.g. `C:/directory/path`).

Viewing 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:


New Global Variable Project

Env:
Sandbox

globalvar1

DEPENDENCIES

Read

Activity

Temp Storage Read Activity

Temp Storage Write Activity

Write

Script

Set globalvar1

For example:

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

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 the global variable with that name 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

An array is a collection of variables. Each member in the collection can be of any supported type, including arrays. The members of an array can be accessed using the `Get()` and `Set()` methods or using the `[] array` syntax. Arrays are zero-indexed, and indices are numeric, sequential, and cannot be skipped.

You can also create arrays of global variables. An array global variable is an array of other global variables that in turn can be arrays.

Setting an Array

You can set values in an array using the `Set()` function with this syntax:

```
type Set(string name, type value [, int index1, int index2, ...])
```

This sets the value of the global variable with the given name to value, and returns the value. If the first argument is an array or the name of an array global variable, you can set the value of an array variable by specifying its index (or indices for multi-dimensional arrays) as the third argument.

Not all the items in an array have to be of the same type. For example, you can create an array that holds a date, an integer, and a string. You can even create arrays inside other arrays.

This example creates an array with three variables of different types where each entry represents the current date and time:

```
<trans>
$right_now = Now();
Set($now, $right_now, 0);
Set($now, Long($right_now), 1);
Set($now, String($right_now), 2);
</trans>
```

As arrays are zero-indexed, the first element is at index 0 and the last element is at index (size-1). To append data to an array, pass either a negative index value or the size of the array (`Length($arr)`). Setting an element with an index larger than the size of the array results in an index out-of-range error. Setting non-array data elements can also be done using the `$de_name` syntax.

Here are examples of setting values in an array:

```
// Set the n:th entry in an array to the string "value"
Set($arr, "value", n-1);

// Another way to set the n:th entry of an array
Set($arr, "value", Length($arr));

// Sets the value to a new variable at the end of the array
Set($arr, "value", -1);

// Set the n:th entry of the m:th array
Set($record_set, "value", m-1, n-1);
```

TIP: For additional syntax that can be used to define values in an array, see Dictionary and Array Functions.

Accessing an Array

You can access the items of an array using the `Get()` method with this syntax:

```
type Get(string name[, int index1, int index2, ...])
```

This returns the value of the global variable with the given name. If the first argument is an array or the name of an array global variable, you can get a specific variable by specifying its index (or indices for a multi-dimensional array such as a record-set) as the second argument.

Some Functions have array return values. For example, `SelectNodesFromXMLAny()` returns the results of an XPath query as an array. As another example, `DbExecute()` returns a record set as a two-dimensional array: rows first, then columns. This returns the selected data as an array of arrays (representing the selected rows and columns):

```
<trans>
$resultSet = DbExecute("<TAG>endpoint:database/My Database</TAG>", "select
Result from SimpleCalculatorResults");
$firstRow = Get($resultSet, 0);
$thirdColumnOfSecondRow = $resultSet[2][3];
$secondColumnOfThirdRow = Get($resultSet, 3, 2);
</trans>
```

Arrays are zero-indexed. To access the n:th variable of an array called `"arr"`, use `Get("arr", n-1)`. For multi-dimensional arrays you need to specify all the indices. To access the `n:th` column of the `m:th` row in an array called `ResultSet` you would use `Get("ResultSet", m-1, n-1)`. Attempting to get a variable beyond the end of the array will result in an array out-of-range error.

These are examples of retrieving values from an array:

```
// Return the third array variable
Get($arr, 2);

// Another way to return the third array variable
Get("arr", 2);

// Get the n:th variable of the m:th array in arr
Get($arr, m-1, n-1);
```

This example shows how you can first create a script that builds and returns an array. The second block shows running that script and assignings its returned value to a variable.

Build Array

```
<trans>
// Script to build and return an array
a = Array();
a[Length(a)] = "A";
a[Length(a)] = "B";
a[Length(a)] = "C";
a;
</trans>
```

Call Script to Get Array

```
<trans>
// Call the script to retrieve the array
$Arr = RunScript("<TAG>script:Build Array</TAG>");
</trans>
```

TIP: For additional syntax that can be used to access values in an array, see Dictionary and Array Functions.

Other Types of Global Variables

Project variables and Jitterbit variables, which are also referred to as predefined global variables, are also considered to be global variables because they are available to use globally across a project. Together, these types of variables were formerly referred to as global data elements in Jitterbit Harmony. For additional information, refer to documentation on project variables and Jitterbit variables.