Script Editor

On This Page

Related Articles

- JavaScript
- Jitterbit Script
- Project Pane
- Project Permissions, Collaboration, and Saving
- Script Testing
- Script Types and Creation

Related Topics

- Cloud Studio
- Connectors
- Operations
- Scripts
- Transformations

Last updated: Jan 29, 2020

Introduction

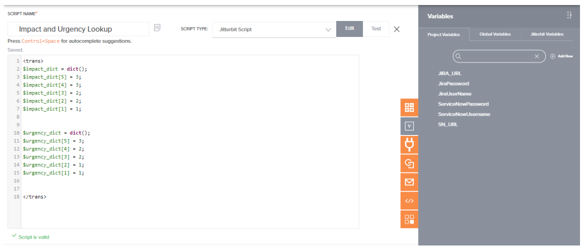
There are two versions of the script editor: (1) the full script editor that is the default for scripts created as a project component and (2) the inline script editor that is present in transformations, which can be expanded to the full script editor by clicking the popout icon ⤢ in the upper right of a script.

The full script editor contains all options covered on this page, while the inline editor offers a pared down version for quick editing.

For more information on types of scripts, see Script Types and Creation.

Accessing the Script Editor

The full script editor is the default script editor for scripts created as a project component:

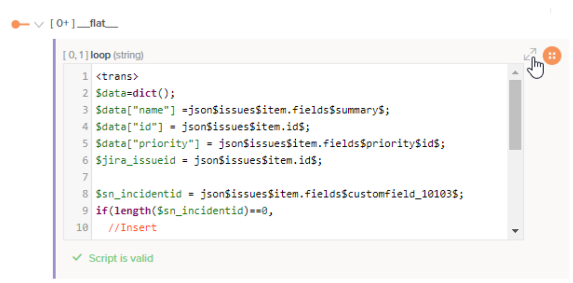


```
<trans>
$impact_dict = dict();
$impact_dict[5] = 3;
$impact_dict[4] = 3;
$impact_dict[3] = 2;
$impact_dict[2] = 2;
$impact_dict[1] = 1;


$urgency_dict = dict();
$urgency_dict[5] = 3;
$urgency_dict[4] = 2;
$urgency_dict[3] = 2;
$urgency_dict[2] = 1;
$urgency_dict[1] = 1;


</trans>
```

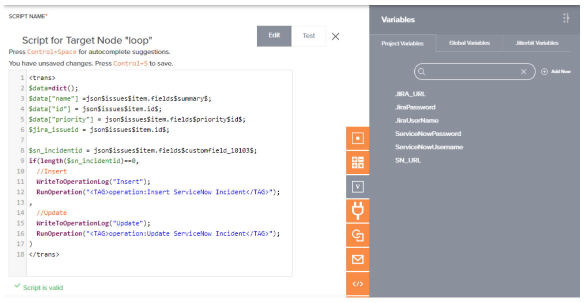
To access the full script editor from within the inline version present within transformations, click the popout icon ⤢ in the upper right of the script:



```
<trans>
$data=dict();
$data["name"] =json$issues$item.fields$summary$;
$data["id"] = json$issues$item.id$;
$data["priority"] = json$issues$item.fields$priority$id$;
$jira_issueid = json$issues$item.id$;

$sn_incidentid = json$issues$item.fields$customfield_10103$;
if(length($sn_incidentid)==0,
  //Insert
```

The full script editor interface will open:



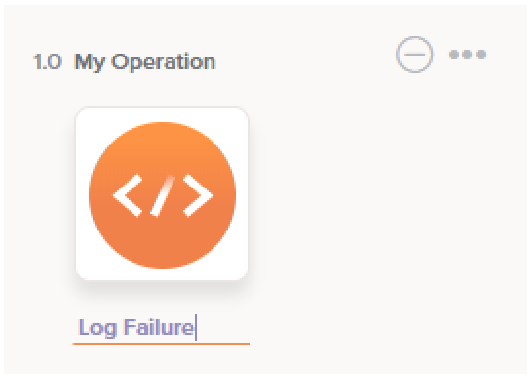
```
<trans>
$data=dict();
$data["name"] =json$issues$item.fields$summary$;
$data["id"] = json$issues$item.id$;
$data["priority"] = json$issues$item.fields$priority$id$;
$jira_issueid = json$issues$item.id$;

$sn_incidentid = json$issues$item.fields$customfield_10103$;
if(length($sn_incidentid)==0,
  //Insert
  WriteToOperationLog("Insert");
  RunOperation("<TAG>operation:Insert ServiceNow Incident</TAG>");
,
  //Update
  WriteToOperationLog("Update");
  RunOperation("<TAG>operation:Update ServiceNow Incident</TAG>");
)
</trans>
```

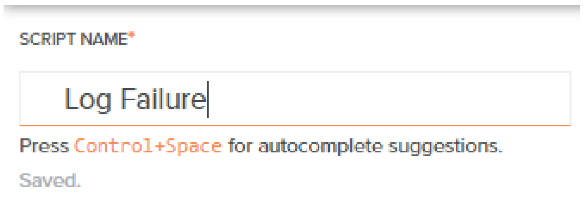
Renaming a Script

By default, a new script is named "New Script," with a number appended if there are additional scripts with the same name. The script name must be unique for each script within the project, regardless of the scripting language used, and must not contain forward slashes (`/`) or colons (`:`).

To rename a script from the design canvas, click the name below the script block to bring the name into edit mode, then enter a name:



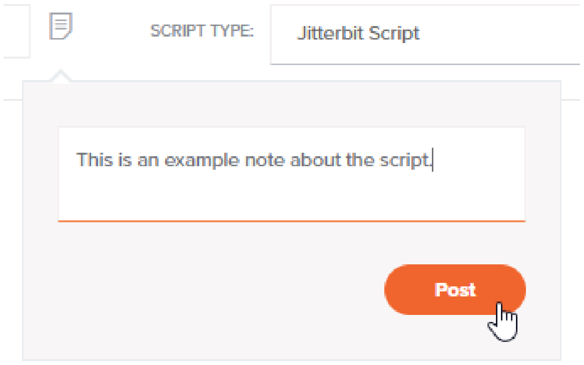
To rename a script from within the script editor, click into the **Script Name** text box along the top left and enter a name:



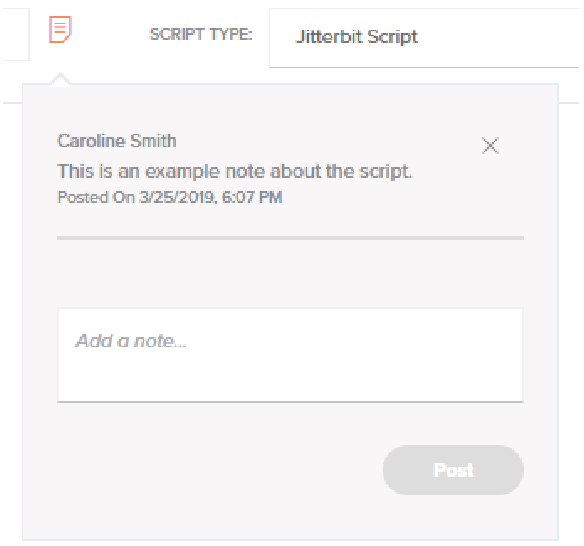
Adding Notes

You can add custom notes to a script that serve as documentation or reminders for yourself or project collaborators. To add a note, click the note icon next to the script name:

An editable area will open where you can enter the text of your note. Then click **Post** to add the note on the script:



Your posted note will be displayed, along with your name and a timestamp. Anyone with edit access to the script can remove notes by clicking the remove icon ✕. To close the note, simply click on another part of the interface.


SCRIPT TYPE:
Jitterbit Script
Caroline Smith
This is an example note about the script.
Posted On 3/25/2019, 6:07 PM
Add a note...
Post

When one or more notes are present, the note icon color is displayed in orange:

Additional notes can be added using the same method described above.

Saving and Viewing Save Status

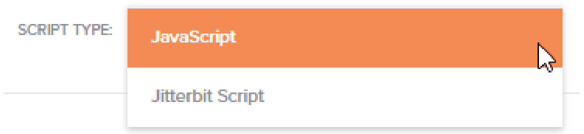
Upon making changes to a script, the script configuration saves automatically. To manually save the script, press `Control+S` (or `Command+S` on Mac).

The save status is displayed along the top of the script below the script name:

To learn more about auto-save, see Project Permissions, Collaboration, and Saving.

Selecting the Script Language

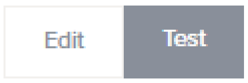
Scripts created as a project component open in Jitterbit Script language by default. The language can be changed to JavaScript using the dropdown:



Scripts created within a transformation, either on a target field or on a node, are limited to Jitterbit Script la nguage.

Toggling Edit and Test Modes

Scripts are opened in edit mode by default. To open the script in test mode, use the toggle in the top right to select **Test**:



For further details, see Script Testing.

Closing the Script

To exit the script and return to the previous screen, click the close icon ✕ in the top right.

Building the Script

The script area is the text area where you enter a script using the appropriate language: Jitterbit Script or JavaScript.

In Jitterbit Script, scripts must be enclosed within a `<trans>` opening tag and `</trans>` closing tag, unless using functions that specifically call for code to be placed outside of these tags, such as several Database Functions.

In JavaScript, scripts must be enclosed within a `<javascript>` opening tag and `</javascript>` closing tag.

These are the main features of the script area:

- **Syntax highlighting:** Different colors are used to distinguish the different parts of an expression, such as function names, strings, and variables.
- **Line numbering:** Line numbers are displayed along the left margin of the script area.
- **Auto-validation:** Harmony provides basic syntax checking and highlighting of lines with errors. If the script passes basic syntax validation, a line below the script area will read: "Script is valid." If the script does not pass basic syntax validation, specific error information will be provided here one line at a time. That is, after resolving an error on one line, additional syntax errors to resolve may be reported for subsequent lines.
- **Drag-and-drop:** Dragging an item from the component palette to the script area will automatically insert the item in the appropriate syntax for use within the script.
- **Auto-completion:** As you type, you can display autocomplete suggestions that begin with the entered string by using `Control+Space`. When only one suggestion is available, it will be inserted automatically. When more than one suggestion is available, you can navigate through the list using the keyboard arrow keys and press `Enter` or `Tab` to insert the suggestion into the script area.
- **Block indentation:** When a complete line or lines are selected, pressing `Tab` indents the selected lines one tab stop to the right. Similarly, `Shift+Tab` moves out the selected lines one tab-stop to the left. If no text is selected, `Shift+Tab` moves the line left to where the cursor is.

In addition, standard browser features for undo, controlling font size, and searching within the script may be used.

Adding Components from the Palette

The script component palette provides access to various components that can be used within a script. Each tab is summarized below, with additional details provided in Jitterbit Script or JavaScript depending on the language.

⚠ **CAUTION:** If a script calls other project components that have not yet been deployed, those components must be deployed before you can run the script successfully. This is true even if those components are not added to any operations (that is, if they appear only within the **Components** tab of the project pane).

▣	**Source Objects:** This tab is present only for Jitterbit Scripts created within a transformation. Within the script, you can reference source data by the path of the field. Add a source object to the Jitterbit Script using one of these methods: • Drag the object from the palette to the script. The full reference path to the source object using the appropriate syntax will be inserted. • Manually enter the full reference path to the source object. For an explanation of the node/field notation, refer to Nodes and Fields.
⊞	**Functions:** This tab provides a list of functions, in the appropriate language, available to use in a script. Add a function to the Jitterbit Script or JavaScript using one of these methods: • Drag the function from the palette to the script. The appropriate syntax for the script language will be inserted. • Begin typing the function name and then press `Control+Space` to display a list of autocomplete suggestions. Select a function to insert the appropriate syntax for the script language. • Manually enter the appropriate syntax for the script language. For additional documentation, see Functions.

Variables: This tab provides access to variables that are available to use globally throughout a project, including global variables, project variables, and Jitterbit variables.

Add a variable to the Jitterbit Script or JavaScript 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 the script language.
- Manually enter the appropriate syntax for the script language.

For additional documentation, see Variables.

NOTE: Local variables are not listed because they are not available globally; however you can still use them locally within a script.

Plugins: This tab provides a list of plugins that can be run inside the script.

Add a plugin to the Jitterbit Script using one of these methods:

- Drag the plugin from the palette to the script. Both the `RunPlugin()` function and the plugin reference will be inserted into the script.
- Begin typing the plugin name and then press `Control+Space` to display a list of autocomplete suggestions. Select a plugin to insert the plugin reference into the script.
- Manually enter the plugin reference.

TIP: Only plugins that are available to use within a script will be listed. Additional plugins can be applied on an activity within an operation. For documentation on using plugins, including how to configure a plugin at the activity level and set global variables, see Plugins.

Operations: This tab provides a list of operations from the project that are available to use in the script.

Add an operation to the Jitterbit Script using one of these methods:

- Drag the operation from the palette to the script. Both the `RunOperation()` function and the operation reference will be inserted into the script.
- Begin typing the operation name and then press `Control+Space` to display a list of autocomplete suggestions. Select an operation to insert the operation reference into the script.
- Manually enter the operation reference.

For additional documentation, see Operations.

Notifications: This tab provides a list of notifications from the project that are available to use in the script.

Add a notification to the Jitterbit Script using one of these methods:

- Drag the notification from the palette to the script. Both the `SendEmailMessage()` function and the notification reference will be inserted into the script.
- Begin typing the notification name and then press `Control+Space` to display a list of autocomplete suggestions. Select a notification to insert a notification reference into the script.
- Manually enter the notification reference.

For additional documentation, see Notifications.

	Scripts: This tab provides a list of all other scripts that were created as project components, including both Jitterbit Scripts and JavaScripts, that are available to use in the script. Add another script within the Jitterbit Script using one of these methods: • Drag the script from the palette to the script. Both the `RunScript()` function and the script reference will be inserted into the script. • Begin typing the script name and then press `Control+Space` to display a list of autocomplete suggestions. Select a script to insert the script reference into the script. • Manually enter the script reference. For additional documentation, see Scripts. **CAUTION:** While Jitterbit Scripts can call JavaScripts, the reverse is not true at this time. Jitterbit JavaScripts cannot call other scripts of any language.
	Endpoints: This tab provides a list of endpoints from the project that are available to use in the script. Add a connection or activity to the Jitterbit Script or JavaScript using one of these methods: • Drag the connection or activity from the palette to the script. The connection or activity reference will be inserted into the script. • Begin typing the connection or activity name and then press `Control+Space` to display a list of autocomplete suggestions. Select a connection or activity to insert the appropriate reference into the script. • Manually enter the connection or activity reference. Depending on the endpoint, you may then use the **Functions** tab to add functions for which to use the connection or activity reference as the argument. For additional documentation, see Connectors.