Creating Custom Workflow Tasks

About Custom Workflow Inputs

Cisco UCS Director Orchestrator offers a list of well-defined input types for custom tasks. You can use the input type list to define the input for custom workflow tasks. Cisco UCS Director also enables you to create a customized workflow input for a custom workflow task. You can create a new input type by cloning and modifying an existing input type.

Prerequisites

Before writing custom tasks, you must meet the following prerequisites:

- Cisco UCS Director is installed and running on your system. For more information about how to install Cisco UCS Director, refer to the Cisco UCS Director Installation and Configuration Guide.
- You have a login with administrator privileges. You must use this login when you create and modify custom tasks.

Creating a Custom Workflow Input

You can create a custom input for a custom workflow task. The created input is displayed in the list of input types that you can map to custom task inputs when the custom workflow task is created.

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 Choose the **Custom Workflow Inputs** tab.

Step 3 Click the **Add** icon.

Step 4 In the **Add Custom Workflow Input** dialog box, complete the following fields:

Name	Description
Custom Input Type Name field	A unique name for the custom input type.
Input Type button	Choose a type of input. Based on the selected input, the other fields appear. For example, when you choose the **Email Address** as the input type, a list of values (LOV) appears. Use the new fields to limit the values of the custom input.

Step 5 Click **Submit**.
The custom workflow input is added to Cisco UCS Director and is available in the list of input types.

Cloning a Custom Workflow Input

You can use an existing custom workflow input in Cisco UCS Director to create a custom workflow input.

Before You Begin

A custom workflow input must be available in Cisco UCS Director.

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 Choose the **Custom Workflow Inputs** tab.

Step 3 Choose the custom workflow input that needs to be cloned.
The **Clone** icon appears at the top of the custom workflow inputs table.

Step 4 Click the **Clone** icon.

Step 5 In the **Custom Input Type Name** field, type a name for the new input.

Step 6 Use the other controls in the **Clone Custom Workflow Input** dialog box to customize the new input.

Step 7 Click **Submit**.

The custom workflow task input is cloned after confirmation and is available for use in the custom workflow task.

Creating a Custom Task

To create a custom task, do the following:

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 Choose the **Custom Workflow Tasks** tab.

Step 3 Click the **Add** icon.

Step 4 In the **Add Custom Workflow Task** dialog box, complete the following fields:

Name	Description
Task Name field	A unique name for the custom workflow task.
Task Label field	A label to identify the custom workflow task.
Activate Task check box	If checked, the custom workflow task is registered with Orchestrator and is immediately usable in workflow.
Register Under Category field	The category under which the custom workflow task is registered.
Brief Description field	A description of the custom workflow task.
Detailed Description field	A detailed description of the custom workflow task.

Step 5 Click **Next**.
The **Custom Workflow Tasks Inputs** window appears.

Step 6 Click the **Add** icon.

Step 7 In the **Add Entry to Inputs** dialog box, complete the following fields:

Name	Description
Input Field Name field	A unique name for the field. The name must start with an alphabetic character and must not contain spaces or special characters.
Input Field Label field	A label to identify the input field.
Input Field Type drop-down list	Choose the data type of the input parameter.

Name	Description
Map to Input Type button	Choose a type of input that can be mapped from another task output or global workflow input.
Mandatory check box	If checked, user must provide a value for this field.
Input Field Size drop-down list	Choose the field size for text and tabular inputs.
Input Field Help field	(Optional) A description that is shown on when you hover the mouse over the field.
Input Field Annotation field	(Optional) Hint text for the input field.
Field Group Name field	If specified, all the fields with matching group names are put into the field group.
Multiple Input check box	If checked, the input field accepts multiple values based on the input field type: • For an LOV—The input field accepts multiple input values. • For a text field—The input field becomes multi-line text field.
Text Field Attributes area—Complete the following fields when the input field type is text.	
Maximum Length of Input field	Specify the maximum number of characters that you can enter in the input field.
LOV Attributes area—Complete the following fields when the input type is List of Values (LOV) or LOV with Radio buttons.	
List of Values field	A comma-separated list of values for embedded LOVs.
LOV Provider Name field	The name of the LOV provider for non-embedded LOVs.
Table Attributes area—Complete the following fields when the input field type is Table, Popup Table, or Table with selection check box.	
Table Name field	A name of the tabular report for the table field types.
Field Input Validation area—One or more of the following fields is displayed depending on your selected data type. Complete the fields to specify how the input fields are validated.	
Input Validator drop-down list	Choose a validator for the user input.

Name	Description
Regular Expression field	A regular expression pattern to match the input value against.
Regular Expression Message field	A message that displays when the regular expression validation fails.
Minimum Value field	A minimum numeric value.
Maximum Value field	A maximum numeric value.

Step 8 Click **Submit**.
A successful entry addition message appears.

Step 9 Click **OK**.

Step 10 Click the **Add** icon to add more entry to inputs.

Step 11 Click **Next**.
The **Custom Workflow Tasks Outputs** window appears.

Step 12 Click the **Add** icon.

Step 13 In the **Add Entry to Outputs** dialog box, complete the following fields:

Name	Description
Output Field Name field	A unique name for the output field. It must start with an alphabetic character and must not contain spaces or special characters.
Output Field Description field	A description of the output field.
Output Field Type button	Choose a type of output. This type determines how the output can be mapped to other task inputs.

Step 14 Click **Submit**.
A successful entry addition message appears.

Step 15 Click **OK**.

Step 16 Click the **Add** icon to add more entry to outputs.

Step 17 Click **Next**
The **Controller** window appears.

Step 18 (Optional) Click the **Add** icon to add a controller.

Step 19 In the **Add Entry to Controller** dialog box, complete the following fields:

Name	Description
Method drop-down list	Choose either a marshalling or unmarshalling method to customize the inputs and/or outputs for the custom workflow task. The method can be one of the following: • **beforeMarshall —** Use this method to add or set an input field and dynamically create and set the LOV on a page (form). • **afterMarshall —** Use this method to hide or unhide an input field. • **beforeUnmarshall —** Use this method to convert an input value from one form to another form—for example , when you want to encrypt a password before sending it to the database. • **afterUnmarshall —** Use this method to validate a user input and set the error message on the page.
Script text area	For the method you chose from the **Method** drop-down list, add the code for the GUI customization script here. **Note** Click the **Add** icon if you want to add code for more methods.

Step 20 Click **Submit**.
A successful entry addition message appears.

Step 21 Click **Next**.
The **Script** window appears.

Step 22 From the **Execution Language** drop-down list, choose the language.

Step 23 In the **Script** field, enter the CloupiaScript code for the custom workflow task.

Step 24 Click **Save Script**.

Step 25 Click **Submit**.
The custom workflow task is created and is available for use in the workflow.

Importing Workflows, Custom Tasks, Script Modules, and Activities

To import artifacts into Cisco UCS Director, do the following:

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 In the **Orchestration** pane, click the **Workflows** tab.

Step 3 Click the **Import** action.

Step 4 In the **Import** dialog box, click **Upload**.

Step 5 In the **File Upload** dialog, click **Click and select a file from your computer**.

Step 6 Select the import file. Cisco UCS Director import and export files have a `.wfdx` file extension.
When the file is uploaded, the **File Upload** dialog displays `File ready for use`.

Step 7 Dismiss the **File Upload** dialog.

Step 8 Click **Next**.
The **Import** dialog displays a list of Cisco UCS Director objects contained in the uploaded file.

Step 9 (Optional) Specify how objects are handled if they duplicate names already in the workflow folder. In the **Import** dialog box, complete the following fields:

Name	Description
Workflows drop-down list	Choose from the following options to specify how identically named workflows are handled: • **Replace—**Replace the existing workflow with the imported workflow. • **Keep Both—**Import the workflow as a new version. • **Skip—**Do not import the workflow.
Custom Tasks drop-down list	Choose from the following options to specify how identically named custom tasks are handled: • **Replace** • **Keep Both** • **Skip**
Script Modules drop-down list	Choose from the following options to specify how identically named script modules are handled: • **Replace** • **Keep Both** • **Skip**

Name	Description
Activities drop-down list	Choose from the following options to specify how identically named activities are handled: • **Replace** • **Keep Both** • **Skip**
Import Workflows to Folder check box	Check this check box to import the workflows. If you do not check the box and if no existing version of a workflow exists, that workflow is not imported.
Select Folder drop-down list	Choose a folder into which to import the workflows. If you chose **[New Folder..]** in the drop-down list, the **New Folder** field appears.
New Folder field	Enter the name of the new folder to create as your import folder.

Step 10 Click **Import**.

Exporting Workflows, Custom Tasks, Script Modules, and Activities

To export artifacts from Cisco UCS Director, do the following:

Step 1 On the menu bar, choose **Policies** > **Orchestration**.
Step 2 In the **Orchestration** pane, click the **Workflows** tab.
Step 3 On the **Workflows** tab, click **Export**.
Step 4 In the **Select Workflows** screen, select the workflows that you want to export.
Step 5 Click **Next**.
Step 6 In the **Select Custom Tasks** screen, select the custom tasks that you want to export.
Step 7 Click **Next**.
Step 8 In the **Export: Select Script Modules** screen, select the script modules that you want to export.
Step 9 Click **Next**.
Step 10 In the **Export: Select Activities** screen, select the activities that you want to export.
Step 11 In the **Export: Confirmation** screen, complete the following fields:

Name	Description
Exported By text field	Your name or a note on who is responsible for the export.

Name	Description
Comments text area	Comments about this export.
Exported File Name text field	The name of the file on your local system. Type only the base filename; the file type extension (`.wfdx`) is appended automatically.

Step 12 Click **Export**.

You are prompted to save the file.

Cloning a Custom Workflow Task from the Task Library

You can clone tasks in the task library to use in creating custom tasks.

The cloned task is a framework with the same task inputs and outputs as the original task. However, note that the cloned task is a framework only. This means that you must write all the functionality for the new task in CloupiaScript.

Note also that selection values for list inputs, such as dropdown lists and lists of values, are carried over to the cloned task only if the list values are not system-dependent. Such things as names and IP addresses of existing systems are system-dependent; such things as configuration options supported by Cisco UCS Director are not. For example, user groups, cloud names, and port groups are system-dependent; user roles, cloud types, and port group types are not.

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 Choose the **Custom Workflow Tasks** tab.

Step 3 Click **Clone From Task Library**.

Step 4 In the **Clone from Task Library** dialog box, click **Select**.

Step 5 Choose a task from the task list.

Step 6 Click **Select**.

A custom workflow task is created from the task library. The new custom task is the last custom task in the Custom Workflow Tasks report. The new custom task is named after the cloned task, with the date appended.

What to Do Next

Edit the custom workflow task.

Cloning a Custom Workflow Task

You can use the existing custom workflow tasks in Cisco UCS Director to create a custom workflow task.

Before You Begin

A custom workflow task must be available in Cisco UCS Director.

Step 1 On the menu bar, choose **Policies** > **Orchestration**.
Step 2 Choose the **Custom Workflow Tasks** tab.
Step 3 Choose the custom workflow task to clone.
The **Clone** icon appears at the top of the custom workflow tasks table.
Step 4 Click the **Clone** icon.
Step 5 In the **Clone Custom Workflow Task** dialog box, update the required fields.
Step 6 Click **Next**.
The inputs defined for the custom workflow tasks appear.
Step 7 Edit the task inputs.
Step 8 Click **Next**.
Edit the task outputs.
Step 9 Click the **Add** icon to add a new output entry.
Step 10 Click **Next**.
Step 11 Edit the controller scripts. See the following topic, Controlling Custom Workflow Task Inputs.
Step 12 Click **Next**.
Step 13 To customize the custom task, edit the task script.
Step 14 Click **Submit**.

Controlling Custom Workflow Task Inputs

Using Controllers

You can modify the appearance and behavior of custom task inputs using the controller interface available in Cisco UCS Director.

When to Use Controllers

Use controllers in the following scenarios:

- To implement complex show and hide GUI behavior including finer control of lists of values, tabular lists of values, and other input controls displayed to the user.
- To implement complex user input validation logic.

With input controllers you can do the following:

- **Show or hide GUI controls:** You can dynamically show or hide various GUI fields such as checkboxes, text boxes, drop-down lists, and buttons, based on conditions. For example, if a user selects UCSM from a drop-down list, you can prompt for user credentials for Cisco UCS Manager or change the list of values (LOVs) in the drop-down list to shown only available ports on a server.

- **Form field validation:** You can validate the data entered by a user when creating or editing workflows in **Workflow Designer**. For invalid data entered by the user, errors can be shown. The user input data can be altered before it is persisted in the database or before it is persisted to a device.
- **Dynamically retrieve a list of values:** You can dynamically fetch a list of values from Cisco UCS Director objects and use them to populate the GUI form objects.

Marshalling and Unmarshalling GUI Form Objects

Controllers are always associated with a form in the **Workflow Designer's** task inputs interface. There is a one-to-one mapping between a form and controller. Controllers work in two stages, *marshalling* and *unmarshalling*. Both stages have two substages, before and after. To use a controller, you marshall (control UI form fields) and/or unmarshall (validate user inputs) the related GUI form objects using the controller's scripts.

The following table summarizes these stages.

Stage	Sub-stage
Marshalling — Used to hide and unhide form fields and for advanced control on LOVs and tabular LOVs.	**beforeMarshall —** Used to add or set an input field and dynamically create and set the LOV on a page (form). **afterMarshall —** Used to hide or unhide an input field.
Unmarshalling - Used for form user input validation.	**beforeUnmarshall —** Used to convert an input value from one form to another form, for example, to encrypt the password before sending it to the database. **afterUnmarshall —** Used to validate a user input and set the error message on the page.

Building Controller Scripts

Controllers do not require any additional packages to be imported.

You do not pass parameters to the controller methods. Instead, the Cisco UCS Director framework makes the following parameters available for use in marshalling and unmarshalling:

Parameter	Description	Example
Page	The page or form that contains all the task inputs. You can use this parameter to do the following: • Get or set the input values in a GUI form. • Show or hide the inputs in a GUI form.	`page.setHidden(id + ".portList", true);` `page.setValue(id + ".status", "No Port is up. Port List is Hidden");`
id	The unique identifier of the form input field. An id is generated by the framework and can be used with the form input field name.	`page.setValue(id + ".status", "No Port is up. Port List is Hidden");// here 'status' is the name of the input field.`
Pojo	POJO (plain old Java object) is a Java bean representing an input form. Every GUI page must have a corresponding POJO holding the values from the form. The POJO is used to persist the values to the database or to send the values to an external device.	`pojo.setLunSize(asciiValue); //set the value of the input field 'lunSize'`

See Example: Using Controllers, on page 12 for a working code sample that demonstrates the controller functionality.

Example: Using Controllers

The following code example demonstrates how to implement the controller functionality in custom workflow tasks using the various methods — beforeMarshall, afterMarshall, beforeUnmarshall and afterUnmarshall.

```
/*
Method Descriptions:

Before Marshall: Use this method to add or set an input field and dynamically create and
set the LOV on a page(form).
After Marshall: Use this method to hide or unhide an input field.
Before UnMarshall: Use this method to convert an input value from one form to another form,
 for example, when you want to encrypt the password before sending it to the database.
After UnMarshall: Use this method to validate a user input and set the error message on the
 page.

*/

//Before Marshall:

/*
Use the beforeMarshall method when there is a change in the input field or to dynamically
create LOVs and to set the new input field on the form before it gets loaded.
 In the example below, a new input field 'portList' is added on the page before the form
is displayed in a browser.
*/
importPackage(com.cloupia.model.cIM);
```

```
importPackage(java.util);
importPackage(java.lang);

var portList = new ArrayList();
var lovLabel = "eth0";
var lovValue = "eth0";

var portListLOV = new Array();
portListLOV[0] = new FormLOVPair(lovLabel, lovValue);//create the lov input field
//the parameter 'page' is used to set the input field on the form
page.setEmbeddedLOVs(id + ".portList",  portListLOV);// set the input field on the form
```

```
//After Marshall :
/*
Use this method to hide or unhide an input field.
*/
page.setHidden(id + ".portList",  true); //hide the input field 'portList'.
page.setValue(id + ".status", "No Port is up.  Port List is Hidden");
page.setEditable(id + ".status", false);
```

```
//Before Unmarshall :

/*
  Use the beforeUnMarshall method to read the user input and convert it to another form
before inserting into the database. For example, you can read the password and store the
password in the database after converting it into base64 encoding, or read the employee
name and convert to the employee Id when the employee name is sent to the database.

In the code example below the lun size is read and converted into an ASCII value.

*/
importPackage(org.apache.log4j);
importPackage(java.lang);
importPackage(java.util);

var size = page.getValue(id + ".lunSize");
var logger = Logger.getLogger("my logger");

if(size != null){
                logger.info("Size value "+size);
                if((new java.lang.String(size)).matches("\\d+")){
                                var byteValue = size.getBytes("US-ASCII"); //convert the
lun size and get the  ASCII character array
                                var asciiValueBuilder = new StringBuilder();
                                for (var i = 0; i < byteValue.length; i++) {
                                                asciiValueBuilder.append(byteValue[i]);
                                }
                                var asciiValue = asciiValueBuilder.toString()+" - Ascii
value"
                                //id + ".lunSize" is the identifier of the input field
                                page.setValue(id + ".lunSize",asciiValue); //the parameter
 'page' is used to set the value on the input field .
                                pojo.setLunSize(asciiValue); //set the value on the pojo.
This pojo will be send to DB or external device.
                }
}
```

```
// After unMarshall :

/*
Use this method to validate and set an error message.
*/
importPackage(org.apache.log4j);
importPackage(java.lang);
importPackage(java.util);

//var size = pojo.getLunSize();
var size = page.getValue(id + ".lunSize");
var logger = Logger.getLogger("my logger");
```

```
logger.info("Size value "+size);
if (size > 50) { //validate the size
                page.setError(id+".lunSize", "LUN Size can not be more than 50MB "); //set
 the error message on the page
                page.setPageMessage("LUN Size can not be more than 50MB");
                //page.setPageStatus(2);
}
```

Example: Creating and Running a Custom Task

To create a custom task, do the following:

Step 1 Go to **Policies** > **Orchestration** > **Custom Workflow Tasks**.

Step 2 Click **Add** and key in the custom task information.

Step 3 Click **Next**.
The **Cloupia Script Interpreter** dialog box appears.

Step 4 Click+ and add the input details.

Step 5 Click **Submit**.

Step 6 Click **Next**.
The custom task output window is displayed.

Step 7 Click **Next**.
The custom task controller window is displayed.

Step 8 Click **Next**.
The script window is displayed.

Step 9 Select JavaScript as the execution language and enter the following script to execute.

```
logger.addInfo("Hello World!");
logger.addInfo("Message "+input.message);
```

where **message** is the input field name.

Step 10 Click **Submit**.
The custom task is defined and added to the custom tasks list.

Step 11 Go to **Workflows** tab.

Step 12 Click **Add** and add a workflow.

Step 13 Drag and drop the 'Hello world custom task' to the workflow designer once the workflow is created.

Step 14 Add 'Hello World custom task' to the designer.

Step 15 Click **Validate workflow**.

Step 16 Click **Execute Now** and click **Submit**.

Step 17 See the log messages in the **Service Request** log window.