Managing the Custom Workflow Tasks

This chapter contains the following sections:

- About Custom Workflow Inputs, page 1
- Custom Workflow Task, page 3

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 enables you to create a customized workflow input for a custom workflow task. You can use the existing input type and create a new one by using the cloning feature.

Creating a Custom Workflow Input

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Input Type Name</td>
<td>A unique name for the custom input type.</td>
</tr>
<tr>
<td>Input Type button</td>
<td>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, the LOV entries table appear. Click the Add icon to add the label and value for the email address.</td>
</tr>
</tbody>
</table>
Cloning a Custom Workflow Input

You can use the 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 **Clone Custom Workflow Input** dialog box, update the required fields.

**Step 6** Click **Submit**.
The custom workflow task input is cloned after confirmation and is available for use in the custom workflow task.

---

Editing a Custom Workflow Input

You can update the custom workflow input details on requirement.

---

**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 edited.
The **Edit** icon appears at the top of the custom workflow inputs table.

**Step 4** Click the **Edit** icon.

**Step 5** In the **Edit Custom Workflow Input** dialog box, update the required fields.

**Step 6** Click **Submit**.
The custom workflow input is updated and is available for use in the custom workflow task creation.
Deleting a Custom Workflow Input

You can delete a custom workflow input from 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 deleted. The Delete icon appears at the top of the custom workflow input table.
Step 4: Click the Delete icon. The selected custom workflow input is deleted from Cisco UCS Director after confirmation.

Custom Workflow Task

Cisco UCS Director Orchestrator allows you to create custom workflow tasks. A task is a specific action or operation that has inputs and outputs. The activated custom workflow task is registered with orchestrator and is immediately usable in workflow. Using Orchestrator, you can organize tasks into distinct workflows to accomplish specific IT services, such as adding VMs. You can then add multiple tasks to a workflow using the workflow UI designer.

Creating a Custom Workflow Task

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Name field</td>
<td>A unique name for the custom workflow task.</td>
</tr>
<tr>
<td>Task Label field</td>
<td>A label to identify the custom workflow task.</td>
</tr>
<tr>
<td>Activate Task check</td>
<td>If checked, the custom workflow task is registered with Orchestrator and is</td>
</tr>
<tr>
<td>Register Under Category</td>
<td>The category under which the custom workflow task is registered.</td>
</tr>
<tr>
<td>Brief Description</td>
<td>A description of the custom workflow task.</td>
</tr>
</tbody>
</table>
Creating a Custom Workflow Task

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detailed Description</td>
<td>A detailed description of the custom workflow task.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Field Name</td>
<td>A unique name of the field. It must start with an alphabetic character and must not contain spaces or special characters.</td>
</tr>
<tr>
<td>Input Field Label</td>
<td>A label to identify the input field.</td>
</tr>
<tr>
<td>Input Field Type</td>
<td>Choose the type of input category.</td>
</tr>
<tr>
<td>Map to Input Type</td>
<td>Choose a type of input that can be mapped to another task output or global workflow input.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>If checked, user must provide a value for this field.</td>
</tr>
<tr>
<td>Input Field Size</td>
<td>Choose the field size for text and tabular inputs.</td>
</tr>
<tr>
<td>Input Field Help</td>
<td>(Optional) A description that is shown on when you hover the mouse over the field.</td>
</tr>
<tr>
<td>Input Field Annotation</td>
<td>(Optional) Hint text for the input field.</td>
</tr>
<tr>
<td>Field Group Name</td>
<td>If specified, all the fields with matching group would be put in to the field group.</td>
</tr>
</tbody>
</table>

**Text Field Attributes** area—Fill this area when the input field type is text.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Input</td>
<td>If checked, the input field accepts multiple values based on the input field type:</td>
</tr>
<tr>
<td></td>
<td>• For LOV—The input field accepts multiple input values.</td>
</tr>
<tr>
<td></td>
<td>• For text field—It will make it multi line text field.</td>
</tr>
<tr>
<td>Maximum Length of Input</td>
<td>Maximum length of the text that you can enter in the input field.</td>
</tr>
<tr>
<td>LOV Attributes</td>
<td>Fill this area if the input type is List of Values(LOV) or LOV with Radio buttons.</td>
</tr>
</tbody>
</table>
### Creating a Custom Workflow Task

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>List of Values</strong> <em>field</em></td>
<td>A list of values for embedded LOVs that are separated by commas.</td>
</tr>
<tr>
<td><strong>LOV Provider Name</strong> <em>field</em></td>
<td>A name of the LOV provider for non-embedded LOVs.</td>
</tr>
<tr>
<td><strong>Table Attributes</strong> area</td>
<td>Fill this area if the input field type is Table or Popup Table or Table with selection check box.</td>
</tr>
<tr>
<td><strong>Table Name</strong> <em>field</em></td>
<td>A name of the tabular report for the table field types.</td>
</tr>
<tr>
<td><strong>Field Input Validation</strong> area</td>
<td>Validate the input fields.</td>
</tr>
<tr>
<td><strong>Input Validator</strong> <em>drop-down list</em></td>
<td>Choose a validator to verify the user input.</td>
</tr>
<tr>
<td><strong>Regular Expression</strong> <em>field</em></td>
<td>A pattern for the input value.</td>
</tr>
<tr>
<td><strong>Regular Expression Message</strong> <em>field</em></td>
<td>A message that displays when the regular expression validation fails.</td>
</tr>
<tr>
<td><strong>Minimum Value</strong> <em>field</em></td>
<td>A minimum number of allowed values in the input field.</td>
</tr>
<tr>
<td><strong>Maximum Value</strong> <em>field</em></td>
<td>A maximum number of allowed values in the input field.</td>
</tr>
</tbody>
</table>

#### 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Field Name</strong> <em>field</em></td>
<td>A unique name of the output field. It must start with an alphabetic character and must not contain spaces or special characters.</td>
</tr>
<tr>
<td><strong>Output Field Description</strong> <em>field</em></td>
<td>A description of the output field.</td>
</tr>
<tr>
<td><strong>Output Field Type</strong> <em>button</em></td>
<td>Choose a type of output category. This type determines how the output can be mapped to other task inputs.</td>
</tr>
</tbody>
</table>

#### 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.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method</strong> drop-down list</td>
<td>Choose either a marshalling or unmarshalling method to customize the inputs and/or outputs for the custom workflow task. It can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- <strong>beforeMarshall</strong> — Use this method to add or set an input field and dynamically create and set the LOV on a page(form).</td>
</tr>
<tr>
<td></td>
<td>- <strong>afterMarshall</strong> — Use this method to hide or unhide an input field.</td>
</tr>
<tr>
<td></td>
<td>- <strong>beforeUnmarshall</strong> — 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.</td>
</tr>
<tr>
<td></td>
<td>- <strong>afterUnmarshall</strong> — Use this method to validate an user input and set the error message on the page.</td>
</tr>
<tr>
<td><strong>Script</strong> text area</td>
<td>For the method you chose from the <strong>Method</strong> drop-down list, add the code for the GUI customization script here.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Click the <strong>Add</strong> icon if you want to add code for more methods.</td>
</tr>
</tbody>
</table>

**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 code to invoke 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.
Exporting a Custom Workflow Task

You can export a custom workflow task to your local system to update or make any changes to the task. Then, you can import the updated custom workflow task to Cisco UCS Director.

**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

Click **Export Tasks**.

The **Export Tasks** dialog box appears.

### Step 4

By default, the login username appears in the **Exported By** field. If required, the name can be changed.

### Step 5

In the **Comments** field, enter the notes about the task or group of tasks that are being exported.

### Step 6

In the **Exported File Name** field, enter the name for the exported custom workflow task file.

### Step 7

Choose one or more tasks from the list of tasks that are available in a table format.

### Step 8

Click **Submit**.

The custom workflow task is exported to your local system.

**What to Do Next**

Import the updated custom workflow task into Cisco UCS Director.

---

Importing a Custom Workflow Task

You can import a custom workflow task that was exported to your local system to Cisco UCS Director after you update the custom workflow task.

**Before You Begin**

You must export a custom workflow task to the local system from Cisco UCS Director.

### Step 1

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

### Step 2

Choose the **Custom Workflow Tasks** tab.

### Step 3

Click **Import Tasks**.

### Step 4

In the **Import Tasks** dialog box, click **Browse** and choose an exported file for upload.

### Step 5

Click **Upload**.

A successful file upload message appears.

### Step 6

Click **OK**.

### Step 7

Click **Next**.
The Save Tasks window displays the task exported time, task exported username, and comments provided when the task is exported. The imported task details appear in a table format.

**Step 8** Check the **Overwrite if the task exists with same name** check box to overwrite the existing task with the imported task.
If this check box is unchecked, a new task will be added with a number (n) to make the name unique.

**Step 9** Click **Import**.
The custom workflow task is imported and is immediately available for use in the workflow.

---

**Cloning a Custom Workflow Task from the Task Library**

You can use the tasks in the task library to create a custom task.

**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.

**What to Do Next**
Edit the custom workflow task to ensure that a unique name and a description are provided for the cloned 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 that needs to be cloned.
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**  
Click the **Add** icon to add a new input entry.

**Step 8**  
To edit the custom workflow task input, choose the entry and click the **Edit** icon to edit the required field.

**Step 9**  
Click **Next**.
The outputs defined for the custom workflow tasks appears.

**Step 10**  
Click the **Add** icon to add a new output entry.

**Step 11**  
To edit the custom workflow task output, select the entry and click the **Edit** icon to edit the required field.

**Step 12**  
Click **Next**.
The **Script** window appears.

**Step 13**  
Update the script, and click **Save Script**.

**Step 14**  
Click **Submit**.
The custom workflow task is cloned.

---

**What to Do Next**

Edit the custom workflow task to ensure that a unique name and a description are provided for the cloned custom workflow task.

---

**Editing a Custom Workflow Task**

You can update the custom workflow task details.

---

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

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

**Step 3**  
Choose the custom workflow task that needs to be edited.  
The **Edit** icon appears at the top of the custom workflow tasks table.

**Step 4**  
Click the **Edit** icon.

**Step 5**  
In the **Modify Custom Workflow Task** dialog box, update the required fields.

**Step 6**  
Click **Next**.
The inputs defined for the custom workflow tasks appear.

**Step 7**  
Click the **Add** icon to add a new input entry.

**Step 8**  
To edit the custom workflow task input, choose the entry and click the **Edit** icon to edit the required field.

**Step 9**  
Click **Next**.
The outputs defined for the custom workflow tasks appear.

**Step 10**  
Click the **Add** icon to add a new output entry.

**Step 11**  
To edit the custom workflow task output, choose the entry and click the **Edit** icon to edit the required field.

**Step 12**  
Click **Next**.
The Script window appears.

**Step 13**  Update the script, and click **Save Script**.

**Step 14**  Click **Submit**.

The custom workflow task is updated and is available for use in the workflow.

---

### Deleting a Custom Workflow Task

You can delete a custom workflow task from 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 that needs to be deleted.

The **Delete** icon appears at the top of the custom workflow task table.

**Step 4**  Click the **Delete** icon.

The selected custom workflow task is deleted from Cisco UCS Director after confirmation.

---

### Controlling Custom Workflow Task Inputs

**Using Controllers**

You can control the inputs of a custom workflow task using the **controller** interface available in Cisco UCS Director. As a result, for every controlled user input, the corresponding GUI field in **Workflow Designer**, appears as modified. Therefore, when you create, or edit a workflow in **Workflow Designer** and add the controlled custom workflow task, you will find modified GUI fields that were not present in the out-of-the-box functionality.

**When to use Controllers**

A general rule of thumb is to use controllers in the following scenarios:

- **Complex** show and hide GUI behavior including finer control on list of values, tabular list of values etc, that are being shown to the user.
- **Complex** user input validation logic.

Below are aspects of customization that can be achieved using controllers:

- **Show or hide GUI controls**: You can dynamically show or hide various GUI fields such as check boxes, 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 up ports in 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 sent to device.

• **Dynamically retrieve a list of values:** You can dynamically fetch a list of values from Cisco UCS Director objects and populate the GUI form objects.

### Marshalling and Unmarshalling GUI form objects

Controllers are always associated with a form in *Workflow Designer’s* task inputs interface. There is a one-to-one mapping between a form and the controller. For controlling, you need to **marshall (control UI form fields)** and/or **unmarshall (validate user inputs)** the related GUI form objects using **controller scripts**. Using these controller scripts, you can implement the marshalling and unmarshalling stages and sub-stages.

The following table summarizes these stages.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Sub-stage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Marshalling</strong> - This stage is used for controlling form fields, such as, show and hide behavior, or advanced control on LOVs and tabular LOVs.</td>
<td><strong>beforeMarshall</strong> — Use this method to add or set an input field and dynamically create and set the LOV on a page(form).  &lt;br&gt; <strong>afterMarshall</strong> — Use this method to hide or unhide an input field.</td>
</tr>
<tr>
<td><strong>Unmarshalling</strong> - This stage is used for form user input validation.</td>
<td><strong>beforeUnmarshall</strong> — 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.  &lt;br&gt; <strong>afterUnmarshall</strong> — Use this method to validate an user input and set the error message on the page.</td>
</tr>
</tbody>
</table>

### Building Controller Scripts

Before you begin building a controller script, note the following:

• **Required Packages:** To be able to use controllers, you will not be required to import any packages.

• **Parameters:** The beforeMarshall, afterMarshall, beforeUnmarshall and afterUnmarshall methods are not passed any parameters, however, the Cisco UCS Director framework will pass the following parameters:
### 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.

```java
/* 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 an user input and set the error message on the page.
*/
```

See Example: Using Controllers, on page 12 for a working code sample that demonstrates the controller functionality.
//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 with the employee Id when the employee name send to the database.
In the code example below the lun size is read and converted into 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);//set the value on the pojo.
        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);
}