

## cisco Creating Custom Workflow Tasks User Guide

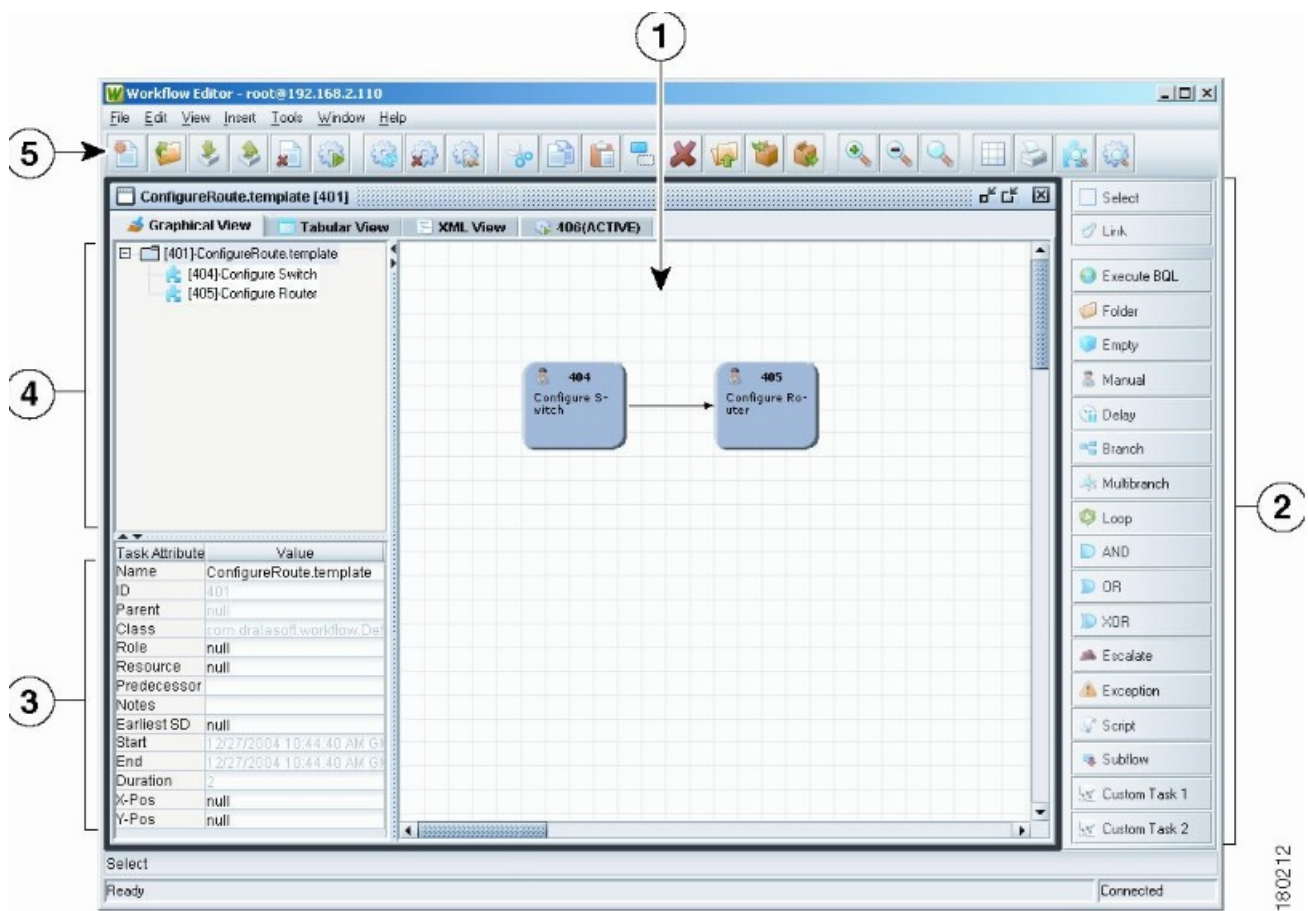
[Home](#) » [Cisco](#) » cisco Creating Custom Workflow Tasks User Guide 

### Contents

- 1 cisco Creating Custom Workflow Tasks
- 2 About Custom Workflow Inputs
  - 2.1 Prerequisites
- 3 Creating a Custom Workflow Input
  - 3.1 Custom Input Validation
  - 3.2 Cloning a Custom Workflow Input
  - 3.3 Creating a Custom Task
- 4 Custom Tasks and Repositories
- 5 Importing Workflows, Custom Tasks, Script Modules, and Activities
- 6 Exporting Workflows, Custom Tasks, Script Modules, and Activities
  - 6.1 Cloning a Custom Workflow Task from the Task Library
  - 6.2 Cloning a Custom Workflow Task
  - 6.3 Controlling Custom Workflow Task Inputs
  - 6.4 Example: Using Controllers
- 7 Using Output of a Previous Task in a Workflow
  - 7.1 Example: Creating and Running a Custom Task
- 8 Documents / Resources
- 9 Related Posts



**cisco Creating Custom Workflow Tasks**



180212

## About Custom Workflow Inputs

Cisco UCS Director Orchestrator offers a list of well-defined input types for custom 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.
- You must have the write CloupiasScript permission to write a custom task using CloupiasScript.
- You must have the execute CloupiasScript permission to execute a custom task created using CloupiasScript.

## Creating a Custom Workflow Input

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

- **Step 1** Choose Orchestration.
- **Step 2** Click Custom Workflow Inputs.
- **Step 3** Click Add.

- **Step 4** On the Add CustomWorkflow Input screen, complete the following fields:
  - Custom Input Type Name—A unique name for the custom input type.
  - Input Type—Check a type of input and click Select. Based on the selected input, other fields appear. For example, when you choose 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.

## Custom Input Validation

Customers may need to validate workflow inputs using external resources. Out of the box, Cisco UCS Director cannot meet every customer's validation needs. To fill this gap, Cisco UCS Director provides an option to validate any input at runtime using a customer-provided script. The script can flag errors in the input and can require valid input before running a service request. The script can be written in any language, can access any external resource, and has access to all the workflow input values.

You can write custom validation scripts using JavaScript, Python, a bash shell script, or any other scripting language.

The following example validation scripts can be found in Cisco UCS Director in Orchestration > Custom Workflow Inputs:

- Example-bash-script-validator
- Example-javascript-validator
- Example-python-validator

You can copy or clone the example scripted workflow inputs to create a new validated input. You can also use the example scripted workflow inputs as a guide for developing your own scripts.

## Regardless of the scripting language, the following features and rules apply to scripted custom input validation:

- All scripted validation is run in a separate process, so that a failing validation process does not affect the Cisco UCS Director process.
- Only generic text inputs can be validated using scripts.
- Validation scripts are run one at a time, in sequence, in the same order in which the inputs appear in the workflow inputs page. A separate process is launched for each validated input.
- A nonzero return value from the script indicates a failed validation. Optionally, you can pass an error message back to the workflow input form.
- All workflow inputs are passed to the validation script in two ways:
  - As arguments to the script in the form "key"="value".
  - As environment variables to the script process. The variable names are the input labels.

For example, if the workflow has an input labeled as Product-Code and the input value is AbC123, the variable is passed to the validator script as "Product-Code"="AbC123".

These input variables can be used by the script if necessary to implement the validation. Exception: Table values contain only the row number of the table selection, and are therefore probably useless.
- The Edit Custom Workflow Input page makes the script available in the Custom Task editor. Syntax is highlighted for all languages. Syntax errors are checked only for JavaScript validators.

## 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** Choose Orchestration.
- **Step 2** Click Custom Workflow Inputs.
- **Step 3** Click the row with the custom workflow input to be cloned.  
The Clone icon appears at the top of the custom workflow inputs table.
- **Step 4** Click Clone.
- **Step 5** Enter the custom input type name.
- **Step 6** Use the other controls in the Clone Custom Workflow Input screen 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** Choose Orchestration.
- **Step 2** Click Custom Workflow Tasks.
- **Step 3** Click Add.
- **Step 4** On the Add Custom Workflow Task screen, complete the following fields:
  - Task Name field—A unique name for the custom workflow task.
  - Task Label field—A label to identify the custom workflow task.
  - Register Under Category field—The workflow category under which the custom workflow task has to be registered.
  - Activate Task check box—If checked, the custom workflow task is registered with Orchestrator and is immediately usable in workflows.
  - 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 Task Inputs screen appears.
- **Step 6** Click Add.
- **Step 7** On the Add Entry to Inputs screen, complete the following fields:
  - Input Field Name field—A unique name for the field. The name must start with 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.
  - Map to Input Type (No Mapping) field—Choose a type of input to which this field can be mapped, if this field 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.

- RBID field—Enter the RBID string for the 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.
- TEXT FIELD ATTRIBUTES area—Complete the following fields when the input field type is text.
- 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.
- 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.
- 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.
- HIDE ON FIELD CONDITION area—Complete the following fields to set the condition to hide the field in a form.
- Hide On Field Name field—An internal name to the field so the program that handles the form can identify the field.
- Hide On Field Value field—The value that has to be sent once the form is submitted.
- Hide On Field Condition drop-down list—Choose a condition at which the field has to be hidden.
- HTML Help field—The help instructions for the hidden field.
- **Step 8** Click Submit.  
The input entry is added to the table.
- **Step 9** Click Add to add more entry to inputs.
- **Step 10** When you are done adding inputs, click Next.  
The Custom Workflow Tasks Outputs screen appears.
- **Step 11** Click Add.
- **Step 12** On the Add Entry to Outputs screen, complete the following fields:
  - 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 field—Check a type of output. This type determines how the output can be mapped to other task inputs.

- **Step 13** Click Submit.

The output entry is added to the table.

- **Step 14** Click Add to add more entry to outputs.

- **Step 15** Click Next

The Controller screen appears

- **Step 16** (Optional) Click Add to add a controller.

- **Step 17** On the Add Entry to Controller screen, complete the following fields:

- 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:
- 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 a 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.

**See Example:** Using Controllers, on page 14.

- Script text area—For the method you chose from the Method drop-down list, add the code for the GUI customization script.

**Note** Click Add if you want to add code for more methods.

If there are any validations to the entered passwords, ensure to change the controller validation for the passwords so that you can edit the custom tasks in workflows.

**Note**

- **Step 18** Click Submit.

The controller is added to the table.

- **Step 19** Click Next.

The Script screen appears.

- **Step 20** From the Execution Language drop-down list, choose a language.

- **Step 21** In the Script field, enter the CloupiaScript code for the custom workflow task.

The Cloupia Script code is validated when you enter the code. If there is any error in the code, an error icon (red cross) is displayed next to the line number. Hover the mouse over the error icon to view the error message and the solution

- **Step 22** Click Save Script.

- **Step 23** Click Submit.

The custom workflow task is created and is available for use in the workflow

## Custom Tasks and Repositories

When you create a custom task, rather than typing in the custom task code into the script window or cutting and pasting code from a text editor, you can import the code from a file stored in a GitHub or BitBucket repository. To do this, you:

1. Create one or more text files in a GitHub or BitBucket repository, either in github.com or a private enterprise GitHub repository.  
Note Cisco UCS Director supports only GitHub (github.com or an enterprise GitHub instance) and or BitBucket. It does not support other Git hosting services including GitLab, Perforce, or Codebase.
2. Register the repository in Cisco UCS Director. See Adding a GitHub or BitBucket Repository in Cisco UCS Director, on page 7.
3. Select the repository and specify the text file that contains the custom task script. See Downloading Custom Task Script Code from a GitHub or BitBucket Repository, on page 8.

### **Adding a GitHub or BitBucket Repository in Cisco UCS Director**

To register a GitHub or a BitBucket repository in Cisco UCS Director, do the following:

#### **Before you begin**

Create a GitHub or BitBucket repository. The repository can be on any GitHub or BitBucket server, public or private that is accessible from your Cisco UCS Director.

Check in one or more files containing JavaScript code for your custom tasks into your repository.

- **Step 1** Choose Administration > Integration.
- **Step 2** On the Integration page, click Manage Repositories.
- **Step 3** Click Add.
- **Step 4** On the Add Repository page, complete the required fields, including the following:
  - In the Repository Nickname field, enter a name to identify the repository within Cisco UCS Director.
  - In the Repository URL field, enter the URL of the GitHub or BitBucket repository.
  - In the Branch Name field, enter the name of the repository branch you want to use. The default name is main branch.
  - In the Repository User field, enter the username for your GitHub or BitBucket account.
  - To add the GitHub repository, in the Password/API Token field, enter the generated API token for your GitHub.  
To generate the API token using GitHub, click Settings and navigate to Developer Setting > Personal access tokens, and click Generate new token.  
To Note add the BitBucket repository, in the Password/API Token field, enter the password for your BitBucket.
  - To default to this repository when you create a new custom task, check Make this my default repository.
  - To test whether Cisco UCS Director can access the repository, click Test Connectivity.

The state of connectivity with the repository is displayed in a banner at the top of the page.

If you are unable to connect and communicate with the GitHub or BitBucket repository from Cisco UCS Director, update Cisco UCS Director to access the Internet through a proxy server. See the Cisco UCS Director Administration Guide.

#### **Note**

- **Step 5** When you are satisfied that the repository information is correct, click Submit.

### **Downloading Custom Task Script Code from a GitHub or BitBucket Repository**

To create a new custom task by importing text from a GitHub or BitBucket repository, do the following:

## Before you begin

Create a GitHub or BitBucket repository and check in one or more text files containing the JavaScript code for your custom tasks into your repository.

Add the GitHub repository to Cisco UCS Director. See Adding a GitHub or BitBucket Repository in Cisco UCS Director, on page

- **Step 1** On the Orchestration page, click Custom Workflow Tasks.
- **Step 2** Click Add.
- **Step 3** Complete the required fields on the Custom Task Information page. See Creating a Custom Task, on page 3.
- **Step 4** Complete the required fields on the Custom Task Inputs page. See Creating a Custom Task, on page 3.
- **Step 5** Complete the required fields on the Custom Task Outputs page. See Creating a Custom Task, on page 3.
- **Step 6** Complete the required fields on the Controller page. See Creating a Custom Task, on page 3.
- **Step 7** On the Script page, complete the required fields:
  - From the Execution Language drop-down list, select JavaScript.
  - Check Use Repository for Scripts to enable the custom task to use a script file from a repository. This enables you to select the repository and specify the script file to use.
  - From the Select Repository drop-down list, select the GitHub or BitBucket repository containing the script files. For details on how to add repositories, see Adding a GitHub or BitBucket Repository in Cisco UCS Director, on page 7.
  - Enter the full path to the script file in the Script filename text field.
  - To download the script, click Load Script.  
The text from the file is copied in the Script text edit area.
  - Optionally, make changes to the downloaded script in the Script text edit area.
  - To save the script as it appears in the Script text edit area, click Save Script.  
When you press Save Script, the script is saved to your current work session. You must click Submit to save the script to the custom task that you are editing.

### Note

- **Step 8** To save the custom task, click Submit.  
If you made changes to the downloaded script in the Script text edit area, the changes are saved to the custom task. No changes are saved to the GitHub or BitBucket repository. If you would like to discard the loaded script and enter your own script, click Discard Script to clear the script window.

## What to do next

You can use the new custom task in a workflow.

## Importing Workflows, Custom Tasks, Script Modules, and Activities

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

**Note** Global variables associated with a workflow will be imported while importing a workflow if the global variable is not available in the appliance.

- **Step 1** Choose Orchestration.



- **Step 2** On the Orchestration page, click **Workflows**.
- **Step 3** Click **Import**.
- **Step 4** On the Import screen, click **Select a File**.
- **Step 5** On the Choose File to Upload screen, choose the file to be imported. Cisco UCS Director import and export files have a .wfdx file extension.
- **Step 6** Click **Open**.  
When the file is uploaded, the File Upload/Validation screen displays **File ready for use** and **Key**.
- **Step 7** Enter the key that was entered when exporting the file.
- **Step 8** Click **Next**.  
The Import Policies screen displays a list of Cisco UCS Director objects contained in the uploaded file.
- **Step 9** (Optional) On the Import Policies screen, specify how objects are handled if they duplicate names already in the workflow folder. On the Import screen, complete the following fields

Name	Description
<b>Workflows</b>	<p>Choose from the following options to specify how identically named workflows are handled:</p> <ul style="list-style-type: none"> <li>• <b>Replace</b>—Replace the existing workflow with the imported workflow.</li> <li>• <b>Keep Both</b>—Import the workflow as a new version.</li> <li>• <b>Skip</b>—Do not import the workflow.</li> </ul>
<b>Custom Tasks</b>	<p>Choose from the following options to specify how identically named custom tasks are handled:</p> <ul style="list-style-type: none"> <li>• Replace</li> <li>• Keep Both</li> <li>• Skip</li> </ul>

Name	Description
<b>Script Modules</b>	<p>Choose from the following options to specify how identically named script modules are handled:</p> <ul style="list-style-type: none"> <li>• Replace</li> <li>• Keep Both</li> <li>• Skip</li> </ul>
<b>Activities</b>	<p>Choose from the following options to specify how identically named activities are handled:</p> <ul style="list-style-type: none"> <li>• Replace</li> <li>• Keep Both</li> <li>• Skip</li> </ul>
<b>Import Workflows to Folder</b>	<p><b>Check Import Workflows to Folder to import the workflows. If you do not check Import Workflows to Folder and if no existing version of a workflow exists, that workflow is not imported.</b></p>
<b>Select Folder</b>	<p>Choose a folder into which to import the workflows. If you chose <b>[New Folder..]</b> in the drop-down list, the <b>New Folder</b> field appears.</p>
<b>New Folder</b>	<p>Enter the name of the new folder to create as your import folder.</p>

- **Step 10** Click Import.

## Exporting Workflows, Custom Tasks, Script Modules, and Activities

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

Note Global variables associated with a workflow will be automatically exported while exporting a workflow.

- **Step 1** Click Export.
- **Step 2** On the Select Workflows screen, choose the workflows that you want to export.  
Custom workflows, tasks, and scripts created in Cisco UCS Director before version 6.6 can fail to import if they contain XML data.

## Note

- **Step 3** Click Next.
- **Step 4** On the Select Custom Tasks screen, choose the custom tasks that you want to export.  
Note The exported custom task contains all custom inputs that are used by that custom task.
- **Step 5** Click Next.
- **Step 6** On the Export: Select Script Modules screen, choose the script modules that you want to export.
- **Step 7** Click Next.
- **Step 8** On the Export: Select Activities screen, choose the activities that you want to export.
- **Step 9** Click Next.
- **Step 10** On the Export: Select Open APIs screen, choose the APIs that you want to export.
- **Step 11** On the Export: Confirmation screen, complete the following fields:

Name	Description
Exported By	Your name or a note on who is responsible for the export.
Comments	Comments about this export.
Encrypt the exported file	Check the Encrypt the exported file check box to encrypt the file to be exported. By default, the check box is checked.
Key	Enter the key for encrypting the file.  This field is displayed only when the Encrypt the exported file check box is checked.  Preserve the key as it is needed while importing the workflow for decryption.
Confirm Key	Enter the key again for confirmation.  This field is displayed only when the Encrypt the exported file check box is checked.
Exported File Name	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. You can also clone a custom task to create a custom task.

The cloned task is a framework with the same task inputs and outputs as the original task. However, 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** Choose Orchestration.
- **Step 2** Click Custom Workflow Tasks.
- **Step 3** Click Clone From Task Library.
- **Step 4** On the Clone from Task Library screen, check the row with the task that you want to clone.
- **Step 5** 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.

- **Step 6** Click Submit

### **What to do next**

Edit the custom workflow task to ensure that the proper name and description are in place for the cloned task.

### **Cloning a Custom Workflow Task**

You can use an existing custom workflow task 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** Choose Orchestration.
- **Step 2** Click Custom Workflow Tasks.
- **Step 3** Click the row with the custom workflow task that you want to clone.  
The Clone icon appears at the top of the custom workflow tasks table.
- **Step 4** Click Clone.
- **Step 5** On the Clone Custom Workflow Task screen, update the required fields.
- **Step 6** Click Next.

The inputs defined for the custom workflow tasks appear.

- **Step 7** Click the row with the task input that you want to edit and click Edit to edit the task inputs.
- **Step 8** Click Add to add a task input entry.
- **Step 9** Click Next.  
Edit the task outputs.
- **Step 10** Click Add to add a new output entry.
- **Step 11** Click Next.
- **Step 12** Edit the controller scripts. See Controlling Custom Workflow Task Inputs, on page 13.
- **Step 13** Click Next.
- **Step 14** To customize the custom task, edit the task script.
- **Step 15** 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

## 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 the 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 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 a controller. Controllers work in two stages, marshalling and unmarshalling. Both stages have two substages, before and after. To use a controller, you marshal (control UI form fields) and/or unmarshal (validate user inputs) the related GUI form objects using the controller's scripts.

The following table summarizes these stages.

Stage	Sub-stage
<b>Marshalling</b> — Used to hide and unhide form fields and for advanced control of LOVs and tabular LOVs.	<b>before Marshall</b> — Used to add or set an input field and dynamically create and set the LOV on a page (form).  <b>after Marshall</b> — Used to hide or unhide an input field.

Stage	Sub-stage
<b>Unmarshalling</b> – Used for form user input validation.	<p><b>before Unmarshall</b> — Used to convert an input value from one form to another form, for example, to encrypt the password before sending it to the database.</p> <p><b>after Unmarshall</b> — Used to validate a user input and set the error message on the page.</p>

### 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
<b>Page</b>	<p>The page or form that contains all the task inputs. You can use this parameter to do the following:</p> <ul style="list-style-type: none"> <li>• Get or set the input values in a GUI form.</li> <li>• Show or hide the inputs in a GUI form.</li> </ul>	<pre>page.setHidden(id + ".portList", true); page.setValue(id + ".status", "No Port is up. Port List is Hidden");</pre>
<b>id</b>	<p>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.</p>	<pre>page.setValue(id + ".status", "No Port is up. Port List is Hidden");// here 'status' is the name of the input field.</pre>
<b>Pojo</b>	<p>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.</p>	<pre>pojo.setLunSize(asciiValue); //set the value of the input field 'lunSize'</pre>

**See Example:** Using Controllers, on page 14 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 — before Marshall, after Marshall, before Unmarshall and after Unmarshall.

/\*

#### 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.get Value(id + ".lunSize");
var logger = Logger .get Logger("my logger");
logger.info("Size value "+size);
if (size > 50) { //validate the size
page. set Error(id+".lunSize", "LUN Size can not be more than 50MB "); //set
the error message on the page
page .set Page Message("LUN Size can not be more than 50MB");
//page. set Page Status(2);
}

```

## Using Output of a Previous Task in a Workflow

You can use the output of a previous task as an input for an another task in a workflow directly from the script of a custom task and an Execute Cloupia Script task of the task library.  
To access this output, you can use one of the following ways:

- Retrieve the variable from the workflow context using the get Input() method.
- Refer to the output using system variable notation.

To retrieve an output using the context getInput() method, use:  
var name = ctxt.getInput("PreviousTaskName.outputFieldName");

### For example:

```

var name = ctxt.getInput("custom_task1_1684.NAME"); // NAME is the name of the task1 output
field that you want to access
To retrieve an output using system variable notation, use:
var name = "${Previous Task Name. output Field Name}";

```

### For example:

```

var name = "${custom_task1_1684.NAME}"; // NAME is the name of the task1 output field that you want to
access

```

## Example: Creating and Running a Custom Task

To create a custom task, do the following:


- **Step 1** Choose Orchestration.
- **Step 2** Click Custom Workflow Tasks.
- **Step 3** Click Add and key in the custom task information.

- **Step 4** Click Next.
- **Step 5** Click + and add the input details.
- **Step 6** Click Submit.
- **Step 7** Click Next.  
The Custom Task Outputs screen is displayed.
- **Step 8** Click + and add the output details for the custom task.
- **Step 9** Click Next.  
The Controller screen is displayed.
- **Step 10** Click + and add the controller details for the custom task.
- **Step 11** Click Next.  
The Script screen is displayed.
- **Step 12** 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 13** Click Save Script.
- **Step 14** Click Submit.  
The custom task is defined and added to the custom tasks list.
- **Step 15** On the Orchestration page, click Workflows.
- **Step 16** Click Add to define a workflow, and define the workflow inputs and outputs.  
Once the workflow inputs and outputs are defined, use the Workflow Designer to add a workflow task to the workflow.
- **Step 17** Double-click a workflow to open the workflow in the Workflow Designer screen.
- **Step 18** On the left side of the Workflow Designer, expand the folders and choose a custom task (for example, 'Hello world custom task').
- **Step 19** Drag and drop the chosen task to the workflow designer.
- **Step 20** Complete the fields in the Add Task (<Task Name>) screen.
- **Step 21** Connect the task to the workflow. See Cisco UCS Director Orchestration Guide.
- **Step 22** Click Validate workflow.
- **Step 23** Click Execute Now and click Submit.
- **Step 24** See the log messages in the Service Request log window.

## Documents / Resources

	<p><a href="#">cisco Creating Custom Workflow Tasks</a> [pdf] User Guide          Creating Custom Workflow Tasks, Custom Workflow Tasks, Creating Workflow Tasks, Workflow Tasks, Tasks</p>
---	---