Customizing Workflow Components

This chapter contains the following sections:

Creating a Compound Task

A compound task is a workflow that functions as a single task. A compound task, like any other task, is atomic: its component tasks are hidden.

You create a compound task by saving a workflow as a compound task when you create or edit the workflow. Do this, for example, if you find yourself building the same series of tasks into several different workflows.

You can define a simple workflow and save it as a compound task, then define another workflow that incorporates the compound task. You can use this pattern to define increasingly complex workflows.

To save an existing workflow as a compound task, do the following:

Note To create a new compound task from scratch, see Creating a Workflow.

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Workflows**.

Step 3 Select a workflow to save as a compound task.

Step 4 Click **Edit**.

Step 5 Check **Save as Compound Task**.

Step 6 If you want all of the workflow's task outputs available as output of the compound task, click **Publish Task outputs as Compound Task outputs**.

Step 7 Click **Next**.

Step 8 On the **Add User Inputs** screen, click **Next**.

Step 9 On the **Add User Outputs** screen, click **Submit**.

The new compound task is available in the **Compound Task** folder when you open the **Workflow Designer**.

Example: Creating a Compound Task

This example demonstrates repeating a workflow task for elements in a list.

Before you begin

Create the example workflow as described in Example: Creating a Workflow.

Step 1 Navigate to **Policies** > **Orchestration**.

Step 2 Click the **Workflows** tab.

Step 3 Locate and select the PowerCycleVM workflow you created in Example: Creating a Workflow.

Step 4 Click **Edit**.

Step 5 In the **Edit Workflow Details** window, check the **Save as Compound Task** check box.

Note None of the tasks has an output, so ignore the **Publish Task Outputs as Compound Task outputs** check box. The workflow has nothing to do with system startup of Cisco UCS Director, so ignore also the **Always execute during System initialization** check box.

Step 6 Click through to the **Edit Workflow Output** page.

Step 7 Click **Submit**.

What to do next

Include the compound task in workflows. For example, you can insert this task before the **Completed (Failed)** task of a workflow that modifies a remotely hosted VM. Then, the VM restarts if a modification fails.

Creating Custom Approvals

You can create custom approval tasks that allow approvers to enter input values at the time of workflow execution.

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Custom Approval Tasks**.

Step 3 Click **Add**.

Step 4 On the **Add Inputs** screen, complete the following fields:

Name	Description
Approval Task Name	The name of the approval task as it appears in the Workflow Designer.

Name	Description
Approval Task Description	A description of the approval task (optional).

Step 5 Click **Add Input Field**.

Step 6 Under the **User Input** heading for the new input field, complete the following fields:

Name	Description
Input Label	The label for the input (supplied by the approver of the task).
Input Description	A description of the input.
Input Type	A drop-down list containing the data type of the input.
Optional Input	If **Optional Input** is checked, the administrator must provide a default value for the input. The approver is not required to provide input.

Step 7 Repeat the previous two steps to add as many inputs as needed.

Step 8 Click **Submit**.

What to do next

You can now include the custom task in a workflow.

Creating Custom Inputs

You can create custom input types to use as workflow inputs. Custom input types are based on an existing input type. They are defined by filter criteria or by a selection set that further narrows the possible values of the input.

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Custom Workflow Inputs**.

Step 3 Click **Add**.

Step 4 On the **Add Custom Workflow Input** page, complete the following fields:

Name	Description
Custom Input Type Name	The input name.
Input Type	Clicking this button brings up a list of existing input types. From the list, choose an input type on which to base your input type.

Name	Description
Filter	Depending on your choice of input type, one or more of the following filter types is available: • **Input Filter**—A text field. Type a text filter string. • **Input List**—A list of values (LOV). Choose which existing values are valid instances for this input. • **Input LOV**—Define a list of allowable name-value pairs for the input. **Note** The **Label** field and **Value** field descriptions should match. • **Input Range**—A text field. Type a range of valid character values. • **Validated Input**—Choose a validator type from the table.

Step 5 Click **(+) Add**.

Step 6 Click **Submit**.

The new input type is added to the **Custom Workflow Types** page. The new input type is available for selection when defining workflow and task inputs.

Generating a Task Using a OpenAPI Specification File

The OpenAPI specification file from connectors such as NetApp, VMware, and vCenter defines all APIs of connectors in the JSON or YAML format. You can upload the OpenAPI specification file to Cisco UCS Director to make use of the connector specific API for creating a task. You can use the task created based on a OpenAPIs specification file, in Cisco UCS Director workflow to perform a specific operation on the connector account.

Note Cisco UCS Director supports uploading of OpenAPI specification file of vCenter Release 6.5 or later.

Before you begin

Ensure that you have the OpenAPI specification file of a connector, such as NetApp, VMware, Intersight, and vCenter, in the JSON format.

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **OpenAPI Integration**.

A list of connectors for which the OpenAPI specification files are uploaded to Cisco UCS Director, is displayed. Expand a connector to view the API specific details of the connector.

Step 3 Upload a OpenAPI specification file of a connector to make use of the connector specific API details of the connector:

a) Click **Upload Spec**.
b) Enter a name under which you want to import the APIs in the OpenAPI specification file.
c) Either drag and drop or click **Select a file** to upload a OpenAPI specification file.

Note The OpenAPI specification file must be in JSON format. If you have the OpenAPI specification file in any other different format such as, YAML, convert the file in to JSON format by choosing **File** > **Convert and save as JSON** in the OpenAPI editor.

d) Click **Submit**.
The connector specific API details are downloaded to Cisco UCS Director.

Step 4 Create a task using the connector specific API:

a) Expand a connector folder and choose an API.
b) Click **Generate Task**.

The **Generate Task** screen appears.

c) Check the **Use Existing Account** check box to use one of the connector account available in Cisco UCS Director. If the **Use Existing Account** check box is left unchecked, Cisco UCS Director will ask for connector login credentials during task execution.

Note For importing APIs of Intersight, you must add respective Intersight DNS to the Cisco UCS Director server. The **Existing Account** and **Credential Policy** checkboxes are not applicable for Intersight APIs.

d) Check the **Use Credential Policy** check box to choose a credential policy from Cisco UCS Director for account access.
e) Check the **Use Proxy** check box to use the system proxy configuration.
f) Enter the API operation name and description. These fields may be auto-populated with details based on information available in the OpenAPI specification file.
g) Choose **http** or **https** as the protocol.

Note The VMware APIs does not support HTTP protocol.

h) Enter the HTTP request type.
i) Displays the base path of the API.
j) Choose the HTTP content type from the drop-down list.
k) Check the **Use the HTTP Basic Authentication** check box to pass the HTTP Basic Authentication parameters during API call.
l) Check the **Use selected Account/policy credentials** check box to choose authentication parameters from selected account or credential policy.

The API path is displayed.

m) Expand **API Parameters**. The default parameters that are applicable for the API are displayed. Click **Add** and complete all the fields in the **Add Entry to API Parameters** screen to add an additional API parameter.

You must add **Accept** and **Concept Type** parameters for all the accounts. In addition, for VMware account, you must add a session ID parameter.

n) Check the **Output Full Response** check box to output the full response of API.

o) Expand **Output**. Click **Add** and define the API output that can be mapped as input for other tasks.
p) Check the **Define Rollback** checkbox to define the rollback specifications for the task.
q) Click **Select** and choose the rollback workflow that needs to be associated with the task. You have to create a rollback workflow for the task in the Workflow designer.
r) Expand **Rollback Workflow Inputs**. Click **Add** and define the parameters for the rollback workflow input.

   The **Password** field must be configured as user input with the help of Macros. In the **Password** field, avoid hardcoding the password.

   The parameters defined as rollback workflow input, will get displayed under the **API Parameters** field after creating the task by clicking **Submit**.

s) Click **Submit**.

What to do next

In the **Workflow Designer**, you can drag and drop the newly created task from the **Available Tasks** > **Custom Tasks** > **OpenAPI Custom tasks** folder to the workflow area and use the task in a workflow.

To reset an API to their original form, choose an API within the connector folder and click **Reset**. Before initiating the reset process, ensure that the workflows that use the API task are deleted from Workflow designer and tasks created based on the API are deleted.

To delete a connector folder, select the folder and click **Delete Folder**. Before initiating the folder deletion process, ensure that APIs within the connector folder are not associated to any task or workflow.

Orchestration Global Variables

Global variables are variables that you can use within Cisco UCS Director Orchestrator to expose several types of variable system information in two places:

- In task input variables inside a workflow, where you can access such information as:
  - Workflow inputs and task outputs
  - Service request IDs
  - VM information such as ID, name, IP address, and power state
- In VM names, where you can access such information as:
  - User information such as group and user IDs
  - Configuration information such as catalog and system profile
  - Deployment information such as cloud name and location

You can also define your own global variables and make them available in workflows.

Note Global variables are overridden by local workflow inputs or outputs. See Input and Global Variable Scoping, on page 8.

Orchestration Variables

Global variables are variables that can be accessed by any workflow task during workflow execution. When you create a Cisco UCS Director workflow, you can use variables in any of the workflow's task, user, or admin input values.

An input field can contain any combination of text and variables. During execution of the workflow, Cisco UCS Director Orchestrator substitutes the variables' values into each task's inputs before executing the task. Furthermore, variables can be nested; nested variables are evaluated recursively from the innermost to the outermost nested variable as long as the expression still contains a valid variable name with an assigned value. See Input and Output System Variables, on page 7.

Variables from the following four categories are available:

Input and Output System Variables
Input and output variables are simply the inputs and outputs of a workflow at runtime. These inputs and outputs can be captured and used to build inputs to tasks in the workflow.

Service Request Global Variables
Service request global variables provide information about the service request containing the current running workflow.

Virtual Machine Global Variables
For workflows executed in the context of a Virtual Machine (VM), VM global variables contain information about the VM.

User-Defined Global Variables
You can create user-defined global variables that are available to any task in any service request. Unlike system-supplied variables, user-defined global variables can be modified and deleted.

The following sections describe the various types of variables.

Input and Output System Variables

Any workflow-level input or previous task output can be used as a variable in a subsequent task. For example, consider a workflow that has two inputs labeled *Enter Disk Size* and *Max Snapshots*. Suppose that the workflow has two tasks with IDs *task1* and *task2* (arranged so that *task1* executes first). Any input values to *task1* or *task2* that takes free-form input can use the following two variables:

- ${Enter Disk Size}
- ${Max Snapshots}

Note System variables consist of a dollar sign ($) followed by the variable name in curly braces ({}). A workflow level input can be used as a variable by including the label associated with the user input in the variable definition.

Also the second task, *task2*, can use the output of *task1*. If *task1* has two output variables, *OUTPUT_VOLUME_NAME* and *OUTPUT_VOLUME_SIZE*, then *task2* can use the following notation to capture their values in its inputs:

- *${task1.OUTPUT_VOLUME_NAME}*
- *${task1.OUTPUT_VOLUME_SIZE}*

Note The variable name for a task output is the task name, followed by a period, followed by the task output variable name: *${taskName.outputName}*.

Input and Global Variable Scoping

In the case where a task or workflow input variable has the same name as a global variable, the value of the input variable is substituted into the expression. The global variable is overridden by the local input variable.

For example, imagine there is a user-defined global variable BMA with the value **`titanium`**. Now suppose there is a workflow with a user input name BMA, and that during a service request a user enters **`aluminum`** for the value of the variable. If a task in the workflow includes a reference to the variable in an input expression: ${BMA}, then the reference evaluates to **`aluminum`**. The global value of **`titanium`** is "hidden" for the duration of the workflow.

Note Scoping is calculated every time a variable substitution is made in a context with more than one identically named variables. This includes substitutions in nested variables at every level.

Using Input Variables to Filter Admin Inputs

You can use input variables to populate an admin input, admin input list, or admin input filter. For example, suppose you have a workflow with the following two custom task inputs:

- Host Pool, of type gen_text_input (generic text), with an input list of values (LOV) containing several pool labels **`Pool 1`**, **`Pool 2`**, etc.
- Host, of type gen_text_input, with an input LOV containing several host/pool label combinations **`Host 1 Pool 1`**, **`Host 2 Pool 1`**, etc., and with an admin filter that refers to the Host Pool input as follows: `CONTAINS ${Host Pool}`.

When you run the workflow and select a pool from the Host Pool LOV, the Host input LOV shows only those entries containing the pool that you selected. For example, if you select **`Pool 2`** from the Host Pool LOV, the Host LOV is limited to **`Host 1 Pool 2`**, **`Host 2 Pool 2`**, and so on. If you select a different pool, the input screen refreshes again and only hosts from the new pool appear in the Host LOV.

Note In this example, when you select an entry from the Host Pool LOV, the workflow input screen refreshes, updating all controls on the page, including the Host LOV. This refresh behavior cannot be triggered by simple text input controls. Therefore, filtering on workflow inputs does not work for free-form, non-list inputs on the same screen.

For example, if Host Pool were of type gen_text_input (rather than a custom input type with an LOV), the Host drop-down would never update no matter what you typed in the Host Pool field.

Nested Variables

You can nest variables in input values. Cisco UCS Director resolves nested variables to any arbitrary level. Put another way, the result of every variable substitution is evaluated to resolve variables until there are no more variables to evaluate.

For example, assume that a workflow has two tasks, **GetStore** and **GetData**. The **GetData** task has an admin input **GetData.DataStore** defined as **`${GetStore.${GetStore.StoreRef}}`**.

The **GetStore** task has the following two outputs:

- StoreRef, with value **`LUN`**
- **LUN**, with value **`7`**

When **GetStore** runs, then evaluation of **GetData.DataStore** proceeds as follows:

`${GetStore.StoreRef}`

evaluates to:

`LUN`

This value is substituted into the value of **GetData.DataStore**:

`${GetStore.${GetStore.StoreRef}}`

which then evaluates to:

`${GetStore.LUN}`

Which in turn is evaluated to:

`7`

The following rules apply to nested variables:

- Nested input values can contain any combination of nested variables and text. For example, this is a valid input expression: **`VM name is ${VM_${VMIndex}}`**. If **`${VMIndex}`** = **`23`** and **`${VM_23}`** = **`DNA`**, then **`${VM_${VMIndex}}`** evaluates to **`VM name is DNA`**.
- Nested input values are evaluated any number of times until the resulting expression no longer contains any variable references. The innermost variable expression is evaluated first, then the resulting expression is evaluated, and so on. For example, this is a valid input expression:

  `${${${Earth}}}`.

  If the following variables are set:

  `${Earth}` = **`Sol`**

  `${Sol}` = **`MilkyWay`**

  `${MilkyWay}` = **`Universe`**

  then evaluation of the expression cascades to its final value as follows:

  `${${${Earth}}}`

  `${${Sol}}`

  `${MilkyWay}`

  `Universe`

- A user-defined global variable can contain references only to other global variables. See User-Defined Global Variables, on page 10.

Service Request Global Variables

In addition to workflow inputs and task outputs, the following variables representing service requests are available:

- *${SR_ID}*—The ID of the current service request
- *${PARENT_SR_ID}*—The ID of the service request that is the parent of the current service request.(Available only if the current service request has a parent.)
- *${ROOT_SR_ID}*—The ID of the top most parent service request.

For the full list of VM system variables, see List of VM Macros and VM Annotations, on page 13.

Virtual Machine System Variables

For workflows that are executed in the context of a VM, more VM system variables are available. VM system variables cannot be used in a non-VM context workflow.

For the full list of VM system variables, see List of VM Macros and VM Annotations, on page 13.

User-Defined Global Variables

You can create, clone, modify, and delete global variables. Global variables are available to any task in any workflow in Cisco UCS Director.

User-defined global variables can be used only in generic text input types. They can be called in nested input variables. They can contain references to other global variables, but cannot contain references to input variables.

Note It is possible to delete a user-defined global variable that is used by a task in an existing workflow. When such a workflow is run, any task with a missing global variable runs without resolving the variable and might cause unanticipated results. It is up to administrators to ensure that service requests do not contain deleted global variables.

Creating Global Variables

To create a global variable, do the following:

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Global Variables**.

Step 3 Click **(+) Add**.

Step 4 On the **Add Global Variable** screen, complete the following fields:

a) In the **Name** field, enter a unique name for the global variable.

b) In the **Description** field, enter a description of the global variable (optional).

c) In the **Value** field, enter a value for the global variable. This is the text that is inserted when the global variable is used in a task.

User-defined global variables have the following restrictions:

- A global variable **Name** may not contain any of the following 19 characters: `"%&'*+,./:;<=>?^|}{`

  A global variable **Description** or **Value** may contain any character.
- A global variable **Name**, **Description**, or **Value** may contain spaces, but may not start or end with a space. Space characters are stripped from the beginning and end of a name, value, or description.
- Global variables are usable only in generic text type inputs.

Step 5 Click **Submit**.

Cloning Global Variables

To clone a global variable, do the following:

Before you begin

You have created a global variable. See Creating Global Variables, on page 10.

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Global Variables**.

Step 3 Choose the global variable you want to clone. You can choose a system-defined or a user-defined global variable.

Step 4 Click **Clone**.

Step 5 On the **Clone Global Variable** screen, complete the following fields:

a) In the **Name** field, enter a unique name for the global variable. You must change the global variable's name.
b) In the **Description** field, enter a description of the global variable (optional).
c) In the **Value** field, enter a value for the global variable. This is the text that is inserted when the global variable is used in a task.

User-defined global variables have the following restrictions:

- A global variable **Name** may not contain any of the following 19 characters: `"%&'*+,./:;<=>?^|}{`

  A global variable **Description** or **Value** may contain any character.
- A global variable **Name**, **Description**, or **Value** may contain spaces, but may not start or end with a space. Space characters are stripped from the beginning and end of a name, value, or description.
- Global variables are usable only in generic text type inputs.

Step 6 Click **Submit**.

Editing Global Variables

To modify a global variable, do the following:

Before you begin

You have created a global variable. See Creating Global Variables, on page 10.

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Global Variables**.

Step 3 From the `User Defined` folder, choose the global variable you want to edit.

Step 4 Click **Edit**.

Step 5 On the **Edit Global Variable** screen, complete the following fields:

a) You cannot change the **Name** field of an existing global variable.

b) In the **Description** field, change or add a description of the global variable (optional).

c) In the **Value** field, change the value for the global variable. This is the text that is inserted when the global variable is used in a task.

User-defined global variables have the following restrictions:

- A global variable **Name** may not contain any of the following 19 characters: `"%&'*+,./:;<=>?^|}{`

  A global variable **Description** or **Value** may contain any character.
- A global variable **Name**, **Description**, or **Value** may contain spaces, but may not start or end with a space. Space characters are stripped from the beginning and end of a name, value, or description.
- Global variables are usable only in generic text type inputs.

Step 6 Click **Submit**.

Deleting Global Variables

To delete a global variable, do the following:

Step 1 Choose **Orchestration**.

Step 2 On the **Orchestration** page, click **Global Variables**.

Step 3 From the `User Defined` folder, choose the global variable you want to delete.

Step 4 Click **Delete**.

Step 5 On the **Delete Global Variable** screen, click **Delete**.

VM Annotations

VM annotations represent information about a VM. You add these variables to a VMware system policy if you choose to define VM annotations in that policy. The output from these variables displays in the Annotations field of the VM in VMware vCenter.

For the full list of VM annotations, see List of VM Macros and VM Annotations, on page 13.

List of VM Macros and VM Annotations

The syntax that you use for VM macros can be different from the syntax used for VM annotations. In addition, more variables are available for VM macros than for VM annotations. The following table shows the correct syntax for VM macros and for VM annotations. If a cell contains N/A, the variable is not available in that context.

For information about the variables that you can use in the **VM Name Template** and **VM Host Name Template** fields of the system policy, see the Cisco UCS Director Administration Guide.

Note This table does not include a complete list of CloupiaScript macros. For information on using CloupiaScript macros, see the Cisco UCS Director CloupiaScript Cookbook.

Variable	VM Macros for Orchestration Workflows	VM Annotations for System Policies
VM name	${*VM_NAME*}	${*VMNAME*}
VM IP address	${*VM_IPADDRESS*}	N/A
VM state (can be either *on* or *off*)	${*VM_STATE*}	N/A
VM state details (can be either *power-on* or *power-off*)	${*VM_STATE_DETAILS*}	N/A
ESX server or host node that hosts the VM	${*VM_PARENT*}	N/A
Cloud used for VM provisioning	${*VM_CLOUD*}	${*CLOUD_NAME*}
Type of cloud	N/A	${*CLOUD_TYPE*}
VM hostname	${*VM_HOSTNAME*}	N/A
Short VM hostname	${*VM_HOSTNAME_SHORT*}	N/A
VM hostname and domain	${*VM_HOSTNAME_DOMAIN*}	N/A
Name of the group to which the VM belongs	${*VM_GROUP_NAME*}	${*GROUP_NAME*}
Full name of the group	N/A	${*FULL_GROUP_NAME*}
ID of the group	${*VM_GROUP_ID*}	N/A
Name of the parent group, if one exists	N/A	${*GROUP_PARENT*}
ID of the catalog used to provision the VM	${*VM_CATALOG_ID*}	N/A
Name of the catalog used to provision the VM	N/A	${*CATALOG_NAME*}

Variable	VM Macros for Orchestration Workflows	VM Annotations for System Policies
VM ID	${*VM_ID*}	N/A
Service request ID for the VM	${*M_SR_ID*}	${*SR_ID*}
Comments from the user who requested the VM	${*VM_COMMENTS*}	${*COMMENTS*}
Virtual data center name	${*VM_VDC_NAME*}	N/A
Virtual data center ID	${*VM_VDC_ID*}	N/A
Type of VM	${*VM_TYPE*}	N/A
Scheduled termination of the VM	${*VM_SCHED_TERM*}	N/A
Location specified in the account	N/A	${*LOCATION*}
Cost center for the VM	N/A	${*COST_CENTER*}
Current time in milliseconds converted to a unique ID for the VM	N/A	${*UNIQUE_ID*}
User who requested the VM	N/A	${*USER*}
Full name of the user who requested the VM	N/A	${*FULL_USER_NAME*}
Appcode from the catalog	N/A	${*APPCODE*}
Name of the system policy associated with the application category	N/A	${*PROFILE_NAME*}
Name of the user who initiated the request	N/A	${*INITIATING_USER*}
Simple name of the user who initiated the request	N/A	${*INITIATING_USER_SIMPLE_NAME*}
Email address of the user who submitted the request	N/A	${*SUBMITTER_EMAIL*}
The ID of the user who submitted the request	N/A	${*SUBMITTER_USERID*}
First name of the user who submitted the request	N/A	${*SUBMITTER_FIRSTNAME*}
Last name of the user who submitted the request	N/A	${*SUBMITTER_LASTNAME*}

Variable	VM Macros for Orchestration Workflows	VM Annotations for System Policies
The role of the user who submitted the request	N/A	${*SUBMITTER_ROLE*}
Name of the group to which the user who submitted the request belongs	N/A	${*SUBMITTER_GROUPNAME*}
The ID of the group to which the user who submitted the request belongs	N/A	${*SUBMITTER_GROUPID*}