Workflow Designer

2022-01-28T15:53:12.000Z

> **Note:** To execute the new tasks for VMware vCenter networking operations, you should upgrade to Intersight Assist version 1.0.9-278.

{{% alert %}} #### Important: - Fixed The Orchestration option is visible in the left panel only if you have installed the Intersight Premier license. - Open Only an Account Administrator or a user with the Workflow Designer privilege can use the Workflow Designer to create and edit workflows. {{% /alert %}}

To launch the Workflow Designer, choose Orchestration > Workflows. A dashboard displays the following details under these tabs, namely, My Workflows, Sample Workflows, and All Workflows:

- Validation Status
- Last Execution Status
- Top Workflows by Execution Count
- Top Workflow Categories
- Number of System Defined Workflows
- Top Workflows by Targets

Using the dashboard, you can create, edit, clone, or delete a tab. To create your own custom view tab, click + and specify a name and then choose the required parameters that need to be displayed in the columns, tag columns, and widgets. You can rename the tabs if it does not have a Lock icon.

Below the dashboard is a tabular list of workflows displaying the following information:

- Display Name
- Description
- System Defined
- Default Version
- Executions
- Last Execution Status

- Validation Status
- Last Update
- Organization

The Actions column allows you to do the following actions:

- **Execute**—Executes the workflow.
- **History**—Displays workflow execution history.
- **Manage Versions**—create and manage versions for workflows. See Managing Versions for Workflows.
- **Delete**—Delete a workflow.
- **Retry**—Retry a failed workflow.

Creating a Workflow

Workflow creation can be broken down into the following sequence of steps:

1. **Defining a workflow**—you specify the display name, description, and other important attributes.
2. **Defining workflow inputs, workflow variables, and workflow outputs**—you can specify which input parameters are mandatory for the workflow execution, define variable(s) for workflow, and the output(s) generated on successful execution.
3. **Adding workflow tasks**—you can add one or more workflow tasks in the Workflow Designer that are needed for the workflow to carry out its function.
4. **Validate the workflow**—you can validate a workflow to ensure that there are no errors in connecting task inputs and outputs.

Defining a Workflow

1. Choose Orchestration from the left navigation pane.
2. Choose Create Workflow. The Create Workflow screen is categorized into the following areas to help you create a workflow: The following table describes the options available to you while creating a workflow:

Workflow Designer Area/Option	Description	Details
Workflow Designer Area/Option	Description	Details
General Tab	Displays workflow details, and also inputs, variables, and outputs of the workflow.	You can add a user-friendly short name, reference name, description, and assign an organization to the workflow. You can also specify a version or set a tag to the workflow. In order to categorize your workflow, use the tag key Category and provide an appropriate category value. Use the following checkboxes to: **Set as Default Version**—Sets this version as the default version for the workflow. **Retryable**—Execute the workflow from the point of a failure or retry the execution of the entire workflow. You can retry the execution of the workflow for up to two weeks after the last update to the workflow. **Enable Debug Logs**—Collects the workflow logs for each tasks. You can analyze and debug the workflow execution. **Workflow Inputs**—You can click Add Workflow Input and add workflow inputs. Each input has a user-friendly display name, a reference name, description, restrictions, and data type. Also, you can set a default value and encrypt it. For more information, see Defining Workflow Inputs, Variables, and Outputs. **Workflow Variable**—You can click Add Workflow Variable and add a

Workflow Designer Area/Option	Description	Details
Designer Tab	Displays the design area where you synthesize your workflow.	Categorized into the following areas to help you create a workflow:
Mapping Tab	Displays the relationship between the task inputs and the outputs of the selected workflow. Information on workflow inputs and workflow outputs is also displayed.	Expand and collapse the **Task Inputs** and **Task Outputs** pane.
Code Tab	Displays the code view of the workflow definition	Read-only view of the workflow. You can copy the code, use this as a sample to create a workflow using Intersight APIs.
History Tab	Status of the executed/in-progress workflows. This tab appears after executing a workflow.	View workflow execution history, retry or clone a previous execution. The system displays a maximum of 100 instances of the workflow execution. The workflow execution is populated for each version of the workflow. **Note:** When a workflow is successfully executed, the options for retrying the workflow are not displayed. You can retry a failed workflow only when the **Retryable** option is enabled in the **General** tab.

Workflow Designer Area/Option	Description	Details
Execute	Launches the Enter Workflow Inputs window.	Choose the Organization and the Workflow instance name to execute the workflow. For user-created workflows, the organization must match where it was created.
Save Workflow	Validates and saves the workflow.	Review validation errors, if any, and rectify them.
Close Designer	Closes the Workflow Designer.	Closes the Workflow Designer and displays the table view of available workflows.

Requests

Requests are closely related to workflows. You create requests by running workflows; a request is generated every time you execute a workflow in Cisco Intersight. A request is a process under the control of Cisco Intersight.

You can schedule a workflow for later execution, and Cisco Intersight stores details of completed requests. To view the detailed information of a request, choose a request. The following information is displayed in the **Requests** screen:

- **Status**—Displays the status of a workflow. Request can have one of several states depending on its execution status:
  - Running
  - Blocked (for example, awaiting an approval)
  - Completed
  - Failed (a request can fail when one of its component tasks fails to execute properly)
- **Details**—Displays the request details such as name, request ID, target name and type, source name and type, name of the user who has executed the request along with the start time and end time, and duration of the request.
- **Inputs**—Displays the workflow input details
- **Outputs**—Displays the workflow output details

- **Execution Flow**—Enable Show Additional Details to view the logs, input, and output mapping details of the user-defined workflows. Displays the workflow execution history details similar to the details displayed in the History tab.

Defining Workflow Inputs, Variables, and Outputs

Workflows can have any number of tasks, inputs, variables, and outputs.

Any task or workflow input can be either mandatory or optional. A task or workflow cannot run without all of its mandatory inputs. You define whether an input is mandatory or optional when you create the task or workflow.

After specifying the generic properties for a task or workflow, you define the input parameters and variables required for task or workflow execution. In addition, you can set workflow input Parameter Set or Progressive Disclosure rules. These rules control the availability of workflow inputs or filter the data based on the preceding selection during the workflow execution. The Task Designer uses the same parameters and information types as the workflow designer to validate the mappings of task input and output parameters when a task is used in a workflow along with other tasks. Task designer uses the same form as workflow designer to obtain the parameter and information type so as provide same user experience between workflow designer and task designer.

1. Choose Orchestration from the left navigation pane and click Create Workflow.
2. In the General tab, click Add Workflow Input in the Workflow Inputs tab.
3. In the Add Workflow Input screen, enter the following details:
    a. Display Name—Enter user-friendly short name.
    b. Reference Name
    c. Description—Detailed description of the input
    d. Value Restrictions—Choose Required, Collection/Multiple, or can be left blank
    e. Type—Select a data type. For example, String, Integer, Float, Boolean, Json, Enum, MoReference, Target Data Type, or a custom data type.
4. The fields displayed depend on the value you select for the Type field.
    a. If you select the data type as Enum, specify the following:
        i. Enum List—Label for the Enum value.
        ii. Widget type.

iii. Set Default Value—Select the checkbox and select the corresponding default value.

iv. Override—Allow the user to override the default value.

b. If you select the data type as String, specify the following:

i. Minimum and Maximum Range

ii. Regex—Validates the regular expression when you select String in the Type field.

iii. Secure—Indicates that the values of these properties are encrypted and stored in Cisco Intersight.

iv. Object Selector—Enable to configure the Intersight managed object selectors. This attribute is available only when the primitive data is String.

5. Specify the Intersight API Reference and selector attributes to feed the input from the table selection. Value will be passed according to the configuration set in the Value attribute.

a. If you select the data type as Integer or Float, specify the following:

i. Minimum and Maximum Range

ii. Set Default Value—Select the checkbox and the corresponding default value.

b. If you select the data type as Target Data Type or MoReference, specify the following:

i. Intersight API Reference—Enter the URL of the Intersight API configured with parameters.

ii. Selector Attribute—Enter the attribute name(s) that needs to be displayed in the input table column. You can add multiple selector attributes by clicking +.

c. If you select the data type as Composite Data Type, you can specify the field mapping for the input field. With field mapping, you can pass the value of workflow input to a composite data field.

6. To pass the value from the parent composite data type to a child composite data type, you must specify the field mapping in the composite data type field. For more information, see [Creating a Data Type]({{< relref "DataType.md" >}}).

7. To specify the field mapping, do the following:

i. Key—The input field that you want to hide.

ii. Value Attribute—The value in a template style to be passed dynamically during the execution.

8. If the data type is Object Selector, Enum, or Boolean then the input is referenced as \${workflow.input.InputReferenceName}.
9. If the data type is MoReference or Target Data Type the input is > referenced as \${workflow.input.InputReferenceName.AttributeName}.
10. The InputReferenceName is the Reference Name given to the input > field and not the Display Name or Label.
11. Click Add.
12. Repeat this procedure to create multiple inputs Sample Link.
13. To add a Parameter Set rule, click Add Rule > Parameter Set.
14. For more information, see Workflow Workflow Input Parameter Set and Progressive Disclosure Rules.
    a. Name—The name of the Parameter Set rule.
    b. If Field—Select the input field and the properties that will be used to define the Parameter Set rule.
15. The inputs are referenced as *InputName* or *InputName.Properties*. **Note:** The properties are applicable only if the data type is MoRefefence or Target Data Type. For example, for a Target Data Type input field (DeviceRegistrations), you can specify *HypervisorManager.IP Address*.
    a. **Condition**—The condition for the rule. The condition can be Equal to, Not equal to, Contains, or Matches Pattern.
    b. **Value**—Select or enter the value. The Parameter Set rule uses this value along with the Condition specified to compare the value from the If Field.
    c. **Field to be shown**—Select the input field(s) that should be made available when the Parameter Set rule condition is satisfied.
    d. Click **Save**.
16. To add a Progressive Disclosure rule, click Add Rule > Progressive Disclosure. For more information, see Workflow Input Parameter Set and Progressive Disclosure Rules.
    a. **Target Field**—The target input field on which the progressive disclosure is applied.
    b. **Attribute** —The inputs are referenced as *AttributeName.ParameterName*. For example, for an input field, Virtual Manager, you can specify the attribute name as *RegisteredDevice.Moid* or *ConnectionStatus*.
    c. **Condition**—The rule condition for the rule. The condition can be Equal to, Not equal to, or Contains.

d. **Value**—The inputs are referenced as $\backslash InputReferenceName.Attributeor$ {InputReferenceName.ParameterName}. For example, for a VM input field, you can specify the value as $\backslash HypervisorManager.Moidor$ {Target.ConnectionStatus}. **Note:** For a String input field, if you have selected the Object Selector option then you need not specify the Attribute or Parameter value.

17. In the General tab, click Add Workflow Variables in the Workflow Variables tab. For more information, see Workflow Variables.
18. In the Add Workflow Variable screen, enter the following details:
    a. **Reference Name**—Enter user-friendly short name.
    b. **Type**—Select the data type. For example, String, Integer, Float, Boolean, Json, Enum, MoReference, or Target Data Type.
19. The fields displayed depend on the value you select for the Type field.
    a. If you select the data type as String, specify the following:
        i. **Object Selector**—Enable to configure the Intersight managed object selectors. The Object Selector field is optional.
    b. Specify the Intersight API Reference and selector attributes to feed the input from the table selection. The value will be passed according to the configuration set in the Value attribute.
    c. If you select the data type as Enum, specify the following:
        i. **Enum List**—Label and value for the Enum list.
        ii. **Widget type**—The widget type as Radio or none.
    d. If you select the data type as Target Data Type or MoReference, specify the following:
        i. **Intersight API Reference**—Enter the URL of the Intersight API configured with parameters.
        ii. **Selector Attribute**—Enter the attribute name(s) that needs to be displayed in the input table column. You can add multiple selector attributes by clicking +.
    e. Select the Initial Mapping To type and specify the corresponding variable value.
    f. Click **Add**.
20. Click **Workflow Outputs** and click **Add Workflow Output**.
21. Choose a parameter and click **Add**.
22. Click **Save**.

Workflow Input Parameter Set and Progressive Disclosure Rules

When you create a workflow, you can specify the inputs for the workflow execution. After you define the workflow inputs, you can set Parameter Set or Progressive Disclosure rules. These rules control the availability of workflow inputs or filter the data based on the preceding selection during the workflow execution.

> **Note:** You can define a Parameter Set or Progressive Disclosure rule only for a workflow that has more than two input fields with the supported data type.

Parameter Set Rule The Parameter Set rules control the availability of specific parameters or inputs during the execution. After the first input is specified, the Parameter Set rule controls which subsequent input fields are made available during the workflow execution.

Example: The following example shows how to create a Parameter Set rule for a workflow with three input fields. The workflow input fields are:

- Enum—Type is *Enum* and Enum list is *VM* and *HX*.
- Input1—Type is *String*.
- Input2—Type is *String*.

You can create two workflow input Parameter Set rules. During the workflow execution when the Enum value is *VM* only Input1 field is made available. Alternately, when the Enum value is *HX*, only Input2 is made available. The details of the Parameter Set rules are:

- Rule1—If Field is *Enum*, Condition is *Equal to*, Value is *VM*, and Fields to be shown is *Input1.*
- Rule2—If Field is *Enum*, Condition is *Equal to*, Value is *HX*, and Fields to be shown is *Input2.*

The following is a sample of the API request.

```
\"InputParameterSet\": \[\
  {\
    \"Condition\": \"eq\",\
    \"ControlParameter\": \"Enum\",\
    \"EnableParameters\": \[\
      \"Input1\"\
    \],\
    \"Name\": \"Rule1\",\
    \"ObjectType\": \"workflow.ParameterSet\",\
    \"Value\": \"VM\"\
  },\
  {\
```

```
    \"Condition\": \"eq\",\
    \"ControlParameter\": \"Enum\",\
    \"EnableParameters\": \[\
      \"Input2\"\
    \],\
    \"Name\": \"Rule2\",\
    \"ObjectType\": \"workflow.ParameterSet\",\
    \"Value\": \"HX\"\
  },
```

The supported data types for Parameter Set rules are:

- Boolean
- Enum
- String Object Selector
- MoReference
- Target

Progressive Disclosure Rule The Progressive Disclosure rules filter the data available in an input field based on the preceding selection during a workflow execution. The first input field is populated with broadest options. The subsequent input fields are populated with options based on the previous selection.

Example 1: The following example shows how to create a Progressive Disclosure rule for a workflow with two input fields. Both these input fields are MoReference data type. The workflow input fields are:

- **Field1**—Type is *MoReference*, Intersight API Reference is */api/v1/asset/DeviceRegistrations*, and Selector attribute is ConnectionStatus.
- **Field2**—Type is *MoReference*, Intersight API Reference is */api/v1/asset/DeviceRegistrations*, and Selector attribute is ConnectionStatus.

You can create a Progressive Disclosure rule so that during the workflow execution, the data is filtered for Field2 fields based on value in the Field1 field. The details of the Progressive Disclosure rule are:

- **Field2 Rule**—Target Field is Field2, Attribute is ConnectionStatus, Condition is *Equal to*, and Value is \${Field1.ConnectionStatus}.

Example 2:: The following example shows how to create a Progressive Disclosure rule for a workflow with two input fields. Both these input fields are composite custom data type. The workflow input fields are:

- **Input1**—Composite data type with two fields (Field1 and Field2)

  - **Field1**—Type is *MoReference*, Intersight API Reference is */api/v1/asset/DeviceRegistrations*, and Selector attribute is ConnectionStatus.
  - **Field2**—Type is *MoReference*, Intersight API Reference is */api/v1/asset/DeviceRegistrations*, and Selector attribute is ConnectionStatus.
- **Input2**—Composite data type with two fields (Field3 and Field4)
  - **Field3**—Type is *MoReference*, Intersight API Reference is */api/v1/asset/DeviceRegistrations*, and Selector attribute is ConnectionStatus.
  - **Field4**—Type is *MoReference*, Intersight API Reference is */api/v1/asset/DeviceRegistrations*, and Selector attribute is ConnectionStatus.

You can create two Progressive Disclosure rules. During the workflow execution, the data is filtered for Input2 fields based on value in the Input1 fields. The details of the two Progressive Disclosure rules are:

- **Input2.Field3 Rule**—Target Field is Input2.Field3, Attribute is ConnectionStatus, Condition is *Equal to*, and Value is *${Input1.Field1.ConnectionStatus}*.
- **Input2.Field4 Rule**—Target Field is Input2.Field4, Attribute is ConnectionStatus, Condition is *Equal to*, and Value is *${Input1.Field2.ConnectionStatus}*.

The following is a sample of the API request.

```
\"UiInputFilters\": \[\
  {\
    \"Filters\": \[\
      \"ConnectionStatus eq \'\${Input1.Field1.ConnectionStatus}\'\"\
    \],\
    \"Name\": \"Input2.Field3\",\
    \"ObjectType\": \"workflow.UiInputFilter\",\
    \"UserHelpMessage\": \"\"\
  },\
  {\
    \"Filters\": \[\
      \"ConnectionStatus eq \'\${Input1.Field2.ConnectionStatus}\'\"\
    \],\
    \"Name\": \"Input2.Field4\",\
    \"ObjectType\": \"workflow.UiInputFilter\",\
    \"UserHelpMessage\": \"\"\
  }\
\]
```

The supported data types for the Progressive Disclosure rules are:

- MoReference
- Target
- String Object Selector

Workflow Variables

Workflow variables are similar to local variables within functions of a programming language. You define variables for workflows in Workflow Designer, and the scope of the workflow variables lies within the defined workflow. In a workflow, all tasks are bound to the scope of the workflow and have access to the workflow variables that are defined for the workflow. You can use the workflow variable to do the following:

- Simplify complex workflows which have multiple branches based on the conditional operator. Save the output of tasks that get executed in different branches of the workflow into a single variable. This variable can then be mapped in the downstream tasks irrespective of the branch from which it was taken.
- Evaluate the workflow input using a complex expression, save the result as a variable, and reuse the variable in the subsequent tasks.
- Transform the output of a task with the help of template functions and map the value into a workflow variable. The variables can then be used in multiple mappings without having to redo the transformation in every mapping.

To define the workflow variable and specify the corresponding value, you can do the following:

- In the General tab, click Add Workflow Variable and define the variable properties.
- In the Designer tab, click the task, and then click the Variables tab.
  - To add a new workflow variable, click Add Workflow Variable and define the variable properties.
  - To add or edit a mapping to an existing variable, click Edit Mapping and specify the new value.

  Note: To view events for a workflow variable, click the **View** icon.

The workflow variables can be assigned one of the following values:

- Static value—Either the default value of the variable or the value specified in the workflow tasks.

- Direct mapping —Mapped to a workflow input, task output, or another workflow variable.
- Advanced or Transformational mapping—Evaluate a complex expression and save the result in a variable.

After you define the workflow variable, you can map the workflow variable to task input, workflow output, or as an input for another variable. For more information, see Input, Variable, and Output Mapping.

Following is a simple example which explains static mapping for a workflow variable. A workflow has two tasks *Task1* and *Task2* and a workflow variable *Variable1* is defined with a default value as *100*. For *Task1*, in the Variable tab for the task, you can specify a static mapping for the workflow variable value as 90. Similarly, for *Task2*, in the Variable tab for the task, you can specify the workflow variable value as 80 by mapping the *Variable1* value.

When you execute the workflow, the start value of *Variable1* is the default value *100* . After *Task1* is executed, the value of *Variable1* is set to *90* based on the static value that is mapped for the task. After *Task2* is executed, the value of *Variable1* is set to *80* based on the static value that is mapped for the task.

Input, Variable, and Output Mapping

Workflows inputs, variables, and task outputs can be used for mapping and they can be used as direct, transformed, or advanced mapping. The syntax to refer to these will depend on the type of mapping.

- Workflow inputs are referenced as *${workflow.input.InputName}* where InputName is the Reference Name given to the workflow input and not the Display Name or Label.
- Workflow variables are referenced as *${workflow.variable.InputName}* , where InputName is the Reference Name for the workflow variable.
- Task outputs are referenced as *${TaskName.output.OutputName}* where TaskName is the Instance Name for the task and not the Display Name or Label and OutputName is the Reference Name for the task output.

To specify an input mapping, select a task and click Edit against a task input parameter.

To specify a workflow variable mapping, select a task and click Map against a workflow variable parameter.

Following are the available options:

- **Static Value**—Enter a value that is assigned to the task input or workflow variable field.
- **Direct Mapping**—Map an existing input, workflow variable, or task output to the task input.

- **Transformed Mapping**—Allows applying data transformation through a combination of one or more transformation stages for a task\'s input. In each transformation stage, you can select a transformation function along with its inputs. The transformation stages will be converted into a template. You can then preview the template and test the transformation with sample values.

  Note: Transformed mapping is only supported for task inputs of > primitive data types, such as *string*, *integer*, *boolean*, and *float*.

- **Advanced Mapping**—Map a Golang template to the task input or workflow variable. For example, FindAllString function returns a slice of all substrings that match the given regular expression in the given string.

- FindAllString(s, regex string) ([]string, error)\ Example: {{(FindAllString .global.task.input.NumberString \\"([0-9]+)\\")}} \ returns array ["123", "456"] for input string "123 some text 456".

- For example, Atoi function converts the given number in string format to integer type.

- Atoi(s string) (int, error)\ Example: {{Atoi .global.task.input.SizeString}} \ returns integer 2048 for input string "2048"

Operations - Conditional Task

Operations can be used to control the execution path of the workflows. Orchestrator supports Condition tasks which can be used to control the tasks that need to be executed.

Tasks to Control Execution Flow of Workflows

The Conditional Task under Operations allows you to perform programmatic decisional expressions in a conditional task during a workflow execution. The conditional expression can be simple expression or a combined compound expression.

Note: Only JSON style template is supported with conditional expressions.

Conditional expressions support the following operators:

- **Comparison operators** such as === (Equal to), != (Not equal to), > (Greater than), \< (Less than), >= (Greater than or equal to), \<= (Less than or equal to)

- **Arithmetic operators** such as =, -, * (Multiplication), / (division), % (Modulo), ** (Logical AND)

- **Logical operators** such as && (Logical AND), || (Logical OR), ! ((Logical NOT)

- **Ternary operator** such as condition ? val1 : val2

The syntax for workflow input is $\backslash workflow.input.<workflowinputReferenceName>and the syntax for task$
{\

.output.\

}. You can get the values for the task name and task output name from the Code view.

Following is an example for conditional expressions:

```
json {linenos=table, linenostart=1} if ( (\${workflow.input.name} != \'test\' \|\| \${workfl
if ( (\${workflow.input.name} != \'test123' \|\| \${workflow.input.name} != \'test123')) \'t
if ( (\${workflow.input.name}.length === 12 \|\| \${workflow.input.name} !== 'test' )) \'eve
if ( (\${workflow.input.name}.length === 12 \|\| \${workflow.input.name} !== 'test' )) \'eve
if ( (\${workflow.input.name} != null && \'decisionTest\'.indexOf(\${workflow.input.name}) !
if ( (\${workflow.input.name}.toLowerCase() == 'testing')) \'even\'; else \'odd\'; \<\< used
if ( (\${workflow.input.name}.search(/\\s/g) != -1)) \'even\'; else \'odd\';\
if ( (\${workflow.input.name}.match(/\\s/g) != -1)) \'even\'; else \'odd\';\
if ( (\${workflow.input.name}.startsWith(\'Not\',0) \|\| \${workflow.input.name}.endsWith(\'
if ( (\${workflow.input.name}.length >= 3 && \${workflow.input.name}.length \<= 20)) \'even\
if ( (\${workflow.input.name}. === undefined)) 'odd'; else 'even';\
if ( (\${workflow.input.name}.search(\'Failed\') != -1)) \'even\'; else \'odd\';\
if ( (\${workflow.input.name}.toLowerCase() != \'\' \|\| \${workflow.input.name}.toUpperCase
```

The expression is executed during runtime and depending on the result, the respective path is chosen. If none of the conditions match, the default path is chosen.

For example, you can have a conditional task that follows a path of execution depending on the profile state. You can create a condition in a workflow to check the state of the profile. If the profile is in the Assigned state then you can deploy the profile, else delete the profile.

After choosing the conditional task, use the Conditions tab to specify expressions in the Condition field of the tab. The values for the condition are specified in the Value field of the Cases area. Expressions are validated for syntax accuracy when workflow is saved.

The following examples return the value of the condition.

```
\${workflow.input.Names}.length\
\${workflow.input.ArrayOfIds}.length              (here ArrayOfIds represent an array)\
\${workflow.input.Names}.toLowerCase()\
\${workflow.input.Names}.toUpperCase()
```

The following examples return the boolean value \‘true\’ or \‘false\’.

```
\${workflow.input.Names}.startsWith(\'Te\')\
\${workflow.input.Names}.endsWith(\'st\')\
\${workflow.input.Names} == null\
\${workflow.input.Names} != null
```

In addition to workflow, task output can also be used. Here a boolean value is used in a ternary operation. If True, branch1 executes, otherwise branch2 executes *${task1.output.IsValid} ? branch1 : branch2*.

{{< youtube nOeWB34SDFI >}}

Operations - Parallel Loop Task

Operations can be used to control the execution path of the workflows. Orchestrator supports the Parallel Loop operation task that can be used to run a single task or sub-workflow, iteratively, based on a specified count input.

The iteration count can either be a static value that is specified when the workflow is created or a dynamic value that is derived from a workflow input or task output.

When the task is executed, the count 'N' is determined, and the tasks or sub-workflows are scheduled for execution in parallel. When all the instances of the task or sub-workflow reach a final state, the parallel loop operation completes, and the workflow execution moves on to the next task.

Note:

If one of the tasks or sub-workflows fails then the entire parallel loop task fails.

The dynamic values for the count must be specified as a template function. For example, if a loop must run for a count which matches the length of a workflow input called **StringArray**, then the count must be specified using a template function *{{ len .global.workflow.input.StringArray }}*.

In addition, you can use the keyword *.iteration* in a workflow template input to control the inputs that feed into the task. For example, the task within the loop needs to take one value from the workflow input **StringArray**, then the task string input can be mapped to the template *HostGroupName{{.iteration}}* during the workflow creation.

Use the **Parallel Loop** task from the **Operations** section in the Intersight Orchestrator Workflow Designer UI to create a request.

The following table explains the task input properties:

Property	Description
Count	The iteration count value for the parallel loop. The count can either be a static value defined as a constant or a dynamic value defined as an expression that is evaluated to an integer value at execution time. The dynamic values for the count must be specified as a template function. For example, if a loop must run for a count which matches the length of a workflow input called `StringArray`, then the count must be specified using a template function `{{ len .global.workflow.input.StringArray }}`. The count must be less than or equal to 100.

Example: Create a workflow with the Parallel Loop task to create multiple storage host groups

The following example workflow has two tasks—Parallel Loop and New Storage Host Group. The first task runs the Parallel Loop task. In this task, the Count input is a static value and is set to *2*. The second task, New Storage Host Group, creates storage host groups with the host group name as a workflow input.

When the workflow is executed, the count 'N' is determined as 2, and two create storage host group tasks are executed in parallel. When all the instances of the task reach a final state, the parallel loop completes.

The workflow input field required for this example is as follows:

Property	Description
Storage Device	

The following table lists the properties for the Parallel Loop task:

Property	Description
Count	**Count**—*2*

The following table lists the properties for the New Storage Host Group task:

Property	Description
Storage Device	**Workflow Input**—*${workflow.input.StorageDevice}*
Host Group	**Custom Value**—*HostGroupName{{.iteration}}* The keyword *.iteration* in a workflow template input to control the inputs that feed into the task.

After you execute the workflow, you would see the following output:

```
New Storage Host Group
...
    Inputs
       Host Group: HostGroupName0
           Storage Device:{ 2 }
               Moid: 619efa1d6e64612d317a248a
               ObjectType: storage.PureArray
    Outputs
        ConfigResults:[ 1 ]
           Object: { 4 }
               ConfigResCtx: { 1 }
                  EntityData: { 1 }
                      task: workflow.ApiTask
               Message: Host group created successfully.
               State: Ok
               Type: Config
        Host Group: HostGroupName0
New Storage Host Group
...
    Inputs
        Host Group: HostGroupName1
            Storage Device:{ 2 }
                Moid: 619efa1d6e64612d317a248a
                ObjectType: storage.PureArray
    Outputs
        ConfigResults:[ 1 ]
            Object: { 4 }
                ConfigResCtx: { 1 }
                   EntityData: { 1 }
                       task: workflow.ApiTask
                Message: Host group created successfully.
                State: Ok
                Type: Config
        Host Group: HostGroupName1
```

The following is a sample code view:


```
{
  "Catalog": {
    "Moid": "5e6109bc696f6e2d31f856e5",
    "ObjectType": "workflow.Catalog",
    "link": "https://www.intersightcom/api/v1/workflow/Catalogs/5e6109bc696f6e2d31f856e5"
  },
  "DefaultVersion": true,
  "Description": "",
  "InputDefinition": [
    {
      "CustomDataTypeProperties": {
        "CatalogMoid": "",
        "CustomDataTypeId": "",
        "CustomDataTypeName": "",
        "ObjectType": "workflow.CustomDataProperty"
      },
      "Default": {
        "IsValueSet": false,
        "ObjectType": "workflow.DefaultValue",
        "Override": false,
        "Value": null
      },
      "Description": "",
      "DisplayMeta": {
        "InventorySelector": true,
        "ObjectType": "workflow.DisplayMeta",
        "WidgetType": "None"
      },
      "InputParameters": null,
      "IsArray": false,
      "Label": "Storage Device",
      "Max": 0,
      "Min": 0,
      "Name": "StorageDevice",
      "ObjectType": "workflow.TargetDataType",
      "Properties": [
        {
          "ConnectorAttribute": "RegisteredDevice.Moid",
          "ConstraintAttributes": [
            "ObjectType"
          ],
          "DisplayAttributes": [
            "Name",
            "Vendor"
```

```
          ],
          "ObjectType": "workflow.TargetProperty",
          "Selector": "/api/v1/storage/PureArrays",
          "SelectorProperty": {
            "Body": null,
            "Method": "GET",
            "ObjectType": "workflow.SelectorProperty"
          },
          "SupportedObjects": [
            "storage.PureArray"
          ]
        }
      ],
      "Required": true
    }
  ],
  "InputParameterSet": [],
  "Label": "Parallel Loop",
  "Name": "ParallelLoop",
  "OutputDefinition": [],
  "OutputParameters": null,
  "Properties": {
    "Cloneable": true,
    "EnableDebug": false,
    "ExternalMeta": true,
    "ObjectType": "workflow.WorkflowProperties",
    "Retryable": false,
    "SupportStatus": "Supported"
  },
  "Tags": [],
  "Tasks": [
    {
      "Description": "",
      "Label": "",
      "Name": "StartTask",
      "NextTask": "parallelLoop1",
      "ObjectType": "workflow.StartTask"
    },
    {
      "Description": "",
      "Label": "",
      "Name": "SuccessEndTask",
      "ObjectType": "workflow.SuccessEndTask"
    },
    {
      "Description": "",
```

```
      "Label": "",
      "Name": "FailureEndTask",
      "ObjectType": "workflow.FailureEndTask"
    },
    {
      "Count": "2",
      "Description": "A Parallel Loop is a control task that runs one task or one sub-workfl
      "Label": "Parallel Loop",
      "LoopStartTask": "NewStorageHostGroup1",
      "Name": "parallelLoop1",
      "NumberOfBatches": 1,
      "ObjectType": "workflow.LoopTask",
      "OnSuccess": "SuccessEndTask",
      "Parallel": true
    },
    {
      "CatalogMoid": "5c2fc884696f6e2d316c5d59",
      "Description": "Create a host group with host group name as input. On successful execu
      "InputParameters": {
        "HostGroupName": "HostGroupName{{.iteration}}",
        "StorageDevice": "${workflow.input.StorageDevice}"
      },
      "Label": "New Storage Host Group",
      "Name": "NewStorageHostGroup1",
      "ObjectType": "workflow.WorkerTask",
      "OnFailure": "",
      "OnSuccess": "",
      "RollbackDisabled": false,
      "TaskDefinitionId": "5f7ed8a2696f6e2d30eb388b",
      "TaskDefinitionName": "NewStorageHostGroup",
      "UseDefault": false,
      "Version": 2
    }
  ],
  "UiInputFilters": [],
  "UiRenderingData": {
    "Positions": [
      {
        "Name": "StartTask",
        "X": 269.5,
        "Y": 45.5
      },
      {
        "Name": "SuccessEndTask",
        "X": 269.5,
        "Y": 303
```

```
      },
      {
        "Name": "FailureEndTask",
        "X": 412.5,
        "Y": 303
      },
      {
        "Name": "parallelLoop1",
        "X": 149.01800537109375,
        "Y": 116.75314331054688
      },
      {
        "Name": "NewStorageHostGroup1",
        "X": 185,
        "Y": 177
      }
    ]
  },
  "Version": 1,
  "WorkflowMetadata": null
}
```


Managing the Lifecycle of a Workflow

Workflow management consists of organizing, creating, updating, and deletion of workflows. The following table describes the set of actions that you can perform with workflows in Cisco Intersight.

Actions

Description

View Workflows

Choosing Orchestration from the left navigation pane will display all workflows in a tabular format. You can view information such as the workflow name and description, the number of versions available, the number of times the workflow has been executed, the last execution status, validation information and the last update time and date.

Create a New Workflow

Choose Orchestration from the left navigation pane and click Create Workflow. This option opens up the Workflow Designer, using which you can create a new workflow.

Create a Version for a Workflow

You can create multiple versions for a workflow. In addition, you can set a specific version of the workflow as the default version for the workflow. See

Managing Versions for Workflows.

Delete a Workflow

From the tabular list of workflows, you can select a workflow and click Delete. This will remove the workflow and all versions of the workflow.

You cannot delete a workflow when :

workflow is used as sub-workflow task in another workflow.

workflow is in running state.

Execute a Workflow

Select a workflow from the tabular list of workflows and choose Execute. Alternatively, you can click the workflow name, and then choose Execute in the Workflow Designer.

Important: Workflow execution includes privilege-based validation for tasks. With the introduction of privilege-based validation for virtualization, compute and storage tasks, a user must have all the required privileges to execute all the domain tasks within a workflow. For example, a user can successfully execute a workflow that includes storage and virtualization tasks only if the user has both Storage and Virtualization Administrator privileges. In the absence of either one of these privileges, the Execute button will not be displayed and the user cannot execute the workflow.

However, an Account Administrator can execute all workflow.

While configuring the properties of a workflow, if you selected the Retryable option, then in the event of a workflow failure, you can either choose to execute the workflow again from the point of failure, or you can execute the entire workflow again.

Clone a Workflow

From the tabular list of workflows, you can select a workflow and click Clone. This will clone the selected workflow.

Save a Workflow as

From the tabular list of workflows, you can choose a workflow and click. By default, the workflow details are displayed in the Designer tab. You can choose Save As option from the Action drop-down list to clone or create a new version of the existing workflow.

You can also perform this operation in the General tab.

In the Save As screen, you can choose one of the following:

New Version—Specify a new version and related description. Click Save.

This option is not available for a system generated workflow.

Clone—Specify user-friendly short name, reference name, description, and tag key. Click Save. For more information, see Cloning a Workflow.

Cloning a Workflow

You can clone a workflow. The cloned workflow is identical to the original workflow. You can edit the new workflow immediately. You might do this, for example, to create a workflow that is similar to the source workflow that can be edited based on your requirements. The new workflow has a new, separate version history.

To clone a workflow, complete the following steps:

1. Choose Orchestration from the left navigation pane.
2. From the tabular list of workflows, select a workflow and choose Clone.
3. In the Clone Workflow screen, edit the user-friendly short name, reference name, description, and organization details. The organization mappings are displayed only for pre-canned workflows and not for the custom workflows.
4. Check the Open Clone in Editor check box if you want to open the cloned workflow in the workflow designer and edit the details.

   Note: Open Clone in Editor check box is not displayed when you clone a workflow using the Save As option.
5. Click Clone.

Managing Versions for Workflows

Cisco Intersight allows you to create and manage versions for workflows. In addition to creating versions, you can also set a specific version as the default version for the workflow. You can create a version of a workflow and execute it only if you are one of the following users:

- Account Administrator
- User with Workflow Designer privileges

Read-Only users can only view versions of a workflow. They cannot create, edit, execute or delete versions. Users with Storage Administrator and Virtualization Administrator privileges can only view and execute specific versions of a workflow.

To create a version for a workflow, complete the following steps:

1. Choose Orchestration from the left navigation pane.
2. From the tabular list of workflows, select a workflow and choose Manage Versions.

3. In the Manage Versions screen, you can perform the following tasks:
    - Create a new version for the workflow
    - Execute a specific version of the workflow
    - Delete a specific version. You cannot directly delete the default version of a workflow. To delete the default version, perform the following:
    - Make another version the default.
    - Change the workflow version to be deleted as non-default.
    - Delete the workflow version that was changes as non-default.
    - Set a specific version of the workflow as the default version
4. Choose Create a New Version and enter the following details:
    - Source Version—Choose a specific version that will form the source for the new version that you are creating.
    - Version—Specify a version number for the workflow.
    - Description—Enter a description for the version so that you can identify it.
    - Set as Default Version—Check this check box to set this new version as the default version for the workflow.
5. Click Create.
6. After creating a version for a workflow, clicking on the version number on the Manage Versions screen opens the workflow in Workflow Designer.

Note: Deleting a workflow will delete all versions created for the workflow.

Exporting a Workflow

Intersight Cloud Orchestrator enables you to export workflows from an account to your system and then import them to another account.

To export a workflow, complete the following steps:

1. Choose Orchestration from the left navigation pane.
2. Click the Workflow tab.
3. From the tabular list of workflows, do one of the following:

- Select a workflow, click the Ellipsis (…) icon in the same row, and then choose Export Workflow.
- Select multiple workflows, click the Ellipsis (…) icon from the header or footer of the tabular list, and then choose Export Workflow.

Note: You can also export workflows from the Actions menu in the Workflow Designer window.

4. In the Export Workflow screen:
    a. In the JSON File Name field, use the default filename or enter a filename of your choice for the JSON file that stores the workflow components.
    b. Use the Export Tags toggle button to include or exclude the user-defined tags. ICO does not export the system-defined tags.
    c. Click Export.
    d. Save a local copy of the JSON file.

{{< youtube _5J8Q0lyC00 >}} ## Importing a Workflow

Intersight Cloud Orchestrator enables you to import workflows to your account by importing a JSON file that contains the workflow components. You can create the JSON file by exporting the workflow components from another account.

To import a workflow, complete the following steps:

1. Choose Orchestration from the left navigation pane.
2. Click Import.
3. The Import wizard appears.
4. In the Select File screen:
    a. From the Organization drop-down list, choose the organization to which you want to import the workflow(s).
    b. Click Browse, and then select the JSON file that contains the Workflow(s).

    Note: Ensure that the file size of the JSON file is not more than 1MB. If the file size is more than 1MB, export the workflow(s) in batches, and then try import.

    c. Click Next.

16. Intersight Cloud Orchestrator validates the JSON file and displays > the workflow(s) in the Details screen.
17. In the Details screen:
    a. To associate an additional tag to the components listed in the table, enter the tag in the Set Tags field. **Note:** Set Tags is an optional field. You must enter the tag in the key:value format.
    b. If one or more workflow components are already available in the system, choose a rule to replace or skip the duplicate components.

c. Click Import.

5. In the Import Result screen:

    a. Verify the status of the imported workflow.

    b. If you want to view the details of the import request:

- Click the link displayed above the table.
- Alternatively, click the Requests icon displayed in the menu bar.
- For more information, see Requests.

    a. Click Close.

You can execute the imported workflow from the Workflows tab.

Defining Workflow Input Rules

Intersight Cloud Orchestrator allows you to configure input rules that control the display of fields in the workflow execution screen. You can define these input rules for a workflow by using the Intersight API and while creating a composite data type. By default, all input fields for a workflow are displayed in the workflow execution screen. By configuring input rules, the fields in this screen are displayed or hidden based on the value selected for a previous input field.

After configuring input rules in the workflow definition using the Intersight API, you can view the input rules using the Code tab in the Workflow Designer.

Following is a sample of a simple input rule set for a workflow:

```
\"InputParameterSet\":\[\
{\
\"ObjectType\": \"workflow.ParameterSet\",\
\"Name\": \"rule-1\",\
\"ControlParameter\": \"input-1\",\
\"Condition\": \"eq\",\
\"Value\": \"true\",\
\"EnableParameters\": \[\
\"input-2\"\
\]\
},\
{\
\"ObjectType\": \"workflow.ParameterSet\",\
\"Name\": \"rule-2\",\
\"ControlParameter\": \"input-1\",\
\"Condition\": \"ne\",\
\"Value\": \"true\",\
\"EnableParameters\": \[\
```

```
\"input-3\",\
\]\
}\
\]
```

Following is an example for the parameter sets defined at workflow level:

```
{\
  \"Label\": \"Provision VM\",\
  \"Name\": \"ProvisionVM\",\
  \"DefaultVersion\": true,\
  \"Description\": \"\",\
  // Parameter set defintion \
  \"InputParameterSet\": \[\
    {\
      // rule-1 enables \'vmwarehost\' input when value of \'vmtype\' is \'vmware\'\
      \"Name\": \"rule-1\",\
      \"ControlParameter\": \"vmtype\",\
      \"Condition\": \"eq\",\
      \"Value\": \"vmware\",\
      \"EnableParameters\": \[\
        \"vmwarehost\"\
      \]\
    },\
    {\
      // rule-2 enables \'hxcluster\' input when value of \'vmtype\' is \'hx\'\
      \"Name\": \"rule-2\",\
      \"ControlParameter\": \"vmtype\",\
      \"Condition\": \"eq\",\
      \"Value\": \"hx\",\
      \"EnableParameters\": \[\
        \"hxcluster\"\
      \]\
    }\
  \],\
  \"InputDefinition\": \[\
    {\
      \"ObjectType\": \"workflow.PrimitiveDataType\",\
      \"InputParameters\": null,\
      \"Label\": \"VM Type\",\
      \"Name\": \"vmtype\",\
      \"Required\": false,\
      \"Properties\": {\
        \"ClassId\": \"workflow.PrimitiveDataProperty\",\
        \"ObjectType\": \"workflow.PrimitiveDataProperty\",\
```

```
    \"Constraints\": {\
      \"ClassId\": \"workflow.Constraints\",\
      \"ObjectType\": \"workflow.Constraints\",\
      \"EnumList\": \[\
        {\
          \"ClassId\": \"workflow.EnumEntry\",\
          \"ObjectType\": \"workflow.EnumEntry\",\
          \"Label\": \"VMWare\",\
          \"Value\": \"vmware\"\
        },\
        {\
          \"ClassId\": \"workflow.EnumEntry\",\
          \"ObjectType\": \"workflow.EnumEntry\",\
          \"Label\": \"HX\",\
          \"Value\": \"hx\"\
        }\
      \],\
      \"Max\": 0,\
      \"Min\": 0,\
      \"Regex\": \"\"\
    },\
    \"InventorySelector\": \[\],\
    \"Secure\": false,\
    \"Type\": \"enum\"\
  }\
},\
{\
  \"Name\": \"vmname\",\
  \"Label\": \"VM Name\",\
  \"Required\": false,\
  \"Properties\": {\
    \"Type\": \"string\",\
    \"Constraints\": {\
      \"Min\": 0,\
      \"Max\": 0\
    },\
    \"Secure\": false\
  },\
  \"ObjectType\": \"workflow.PrimitiveDataType\"\
},\
{\
  \"Name\": \"vmwarehost\",\
  \"Label\": \"Host\",\
  \"Required\": false,\
  \"Properties\": {\
    \"Type\": \"string\",\
```

```
      \"Constraints\": {\
        \"Min\": 0,\
        \"Max\": 0\
      },\
      \"Secure\": false\
    },\
    \"Default\": {\
      \"Override\": false,\
      \"Value\": null\
    },\
    \"ObjectType\": \"workflow.PrimitiveDataType\"\
  },\
  {\
    \"Name\": \"hxcluster\",\
    \"Label\": \"Cluster\",\
    \"Required\": false,\
    \"Properties\": {\
      \"Type\": \"string\",\
      \"Constraints\": {\
        \"Min\": 0,\
        \"Max\": 0\
      },\
      \"Secure\": false\
    },\
    \"ObjectType\": \"workflow.PrimitiveDataType\"\
  }\
\],\
\"OutputDefinition\": \[\],\
\"OutputParameters\": null,\
\"Tasks\": \[\
  {\
    \"ClassId\": \"workflow.StartTask\",\
    \"Description\": \"\",\
    \"Label\": \"\",\
    \"Name\": \"StartTask\",\
    \"NextTask\": \"NewProfile1596244596840\",\
    \"ObjectType\": \"workflow.StartTask\"\
  },\
  {\
    \"ClassId\": \"workflow.SuccessEndTask\",\
    \"Description\": \"\",\
    \"Label\": \"\",\
    \"Name\": \"SuccessEndTask\",\
    \"ObjectType\": \"workflow.SuccessEndTask\"\
  },\
  {\
```

```
      \"ClassId\": \"workflow.FailureEndTask\",\
      \"Description\": \"\",\
      \"Label\": \"\",\
      \"Name\": \"FailureEndTask\",\
      \"ObjectType\": \"workflow.FailureEndTask\"\
    },\
    {\
      \"CatalogMoid\": \"5dd45ab6696f6e2d301e2df6\",\
      \"ClassId\": \"workflow.WorkerTask\",\
      \"Description\": \"Task to create a Profile\",\
      \"InputParameters\": null,\
      \"Label\": \"Provision VM\",\
      \"Name\": \"ProvisionVM1596244596840\",\
      \"ObjectType\": \"workflow.WorkerTask\",\
      \"OnFailure\": \"\",\
      \"OnSuccess\": \"SuccessEndTask\",\
      \"TaskDefinitionId\": \"5efd7c84696f6e2d309e6877\",\
      \"TaskDefinitionName\": \"NewProfile\",\
      \"Version\": 1\
    }\
  \],\
  \"Version\": 1\
}
```

Following is an example for the parameter sets defined at custom data type level:

```
{\
  \"Label\": \"ClusterType\",\
  \"Name\": \"ClusterType\",\
  \"CompositeType\": true,\
  \"Description\": \"Type definition for a cluster information.\",\
  \"TypeDefinition\": \[\
    {\
      \"ObjectType\": \"workflow.PrimitiveDataType\",\
      \"Label\": \"Cluster Type\",\
      \"Name\": \"clustertype\",\
      \"Properties\": {\
        \"ObjectType\": \"workflow.PrimitiveDataProperty\",\
        \"Constraints\": {\
          \"ObjectType\": \"workflow.Constraints\",\
          \"EnumList\": \[\
            {\
              \"ObjectType\": \"workflow.EnumEntry\",\
              \"Label\": \"VM Ware\",\
              \"Value\": \"vmware\"\
```

```
          },\
          {\
            \"ObjectType\": \"workflow.EnumEntry\",\
            \"Label\": \"HX AP\",\
            \"Value\": \"hxap\"\
          }\
        \]\
      },\
      \"Type\": \"enum\"\
    }\
  },\
  {\
    \"ObjectType\": \"workflow.PrimitiveDataType\",\
    \"Label\": \"VMware Cluster\",\
    \"Name\": \"vmwarecluster\"\
  },\
  {\
    \"ObjectType\": \"workflow.PrimitiveDataType\",\
    \"Label\": \"HX Cluster\",\
    \"Name\": \"hxcluster\"\
  },\
  {\
    \"ObjectType\": \"workflow.PrimitiveDataType\",\
    \"Label\": \"Cluster Description\",\
    \"Name\": \"clusterdescr\"\
  }\
\],\
\"ParameterSet\": \[\
  {\
    \"Name\": \"rule-1\",\
    \"Field\": \"clustertype\",\
    \"Condition\": \"eq\",\
    \"Value\": \"vmware\",\
    \"ShowFields\": \[\
      \"vmwarecluster\"\
    \]\
  },\
  {\
    \"Name\": \"rule-2\",\
    \"Field\": \"clustertype\",\
    \"Condition\": \"eq\",\
    \"Value\": \"hxap\",\
    \"ShowFields\": \[\
      \"hxcluster\"\
    \]\
  }\
```

```
  \]\
}
```

Supported Workflows for Storage Targets

The following table lists workflows supported in the various storage targets available on Intersight:

List of Storage Workflows	Pure Storage	NetApp	Hitachi
New Storage Host	Y	Y	Y
New Storage Host Group	Y	N	N
New VMFS Datastore	Y	Y	Y
New NAS Datastore	N	Y	N
Remove NFS Datastore	N	Y	N
Remove Storage Host	Y	Y	Y
Remove Storage Export Policy	N	Y	N
Remove Storage Host Group	Y	N	N
Remove VMFS Datastore	Y	Y	N
Update Storage Host	Y	Y	Y
Update NAS Datastore	N	Y	N
Update VMFS Datastore	Y	N	N
New Storage Interface	N	Y	N
New Storage Export Policy	N	Y	N
New Storage Virtual Machine	N	Y	N