The bridge to possible

Intersight Cloud Orchestrator

Deep Dive Deck

ICO TMEs
Cloud and Compute

About This Document

This document aims to provide a detailed insight of Intersight Cloud Orchestrator (ICO) capabilities through examples.

Note: some screenshots may change over time due to the nature of Intersight, please refer to the documentation on https://intersight.com/help

Table of Content

Sample Scenario

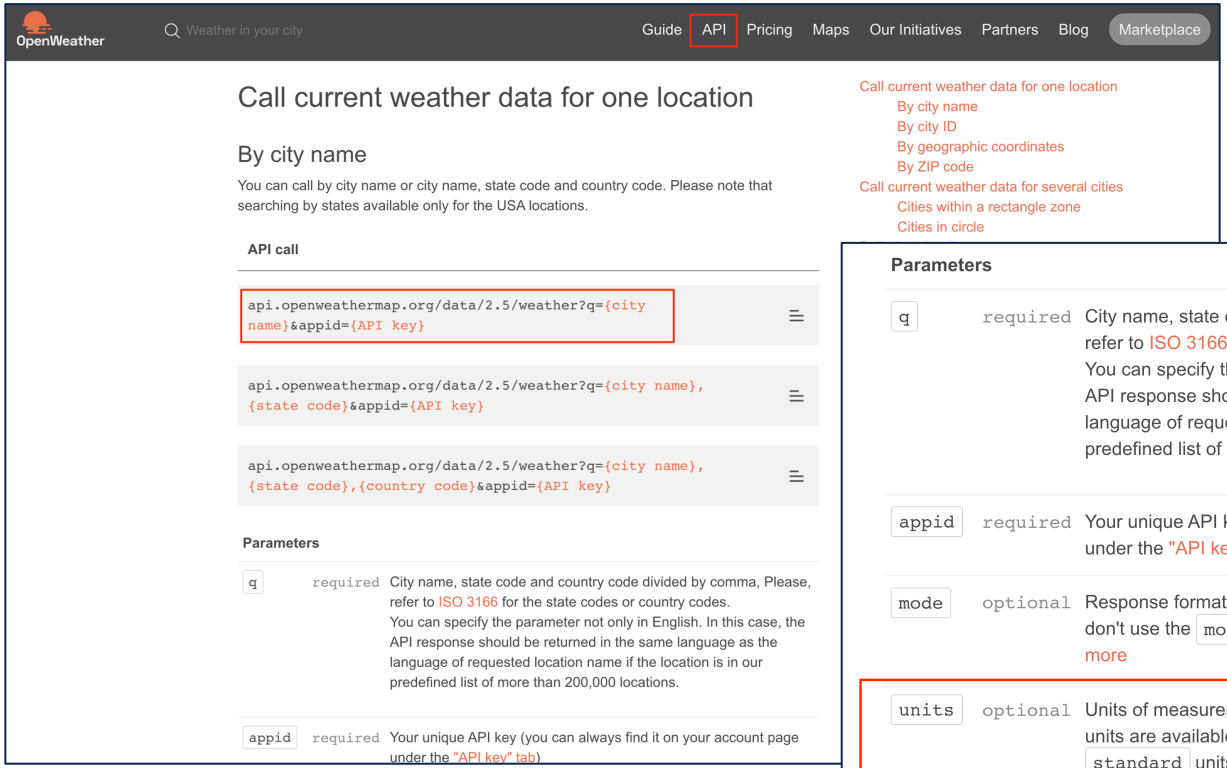
Example workflow

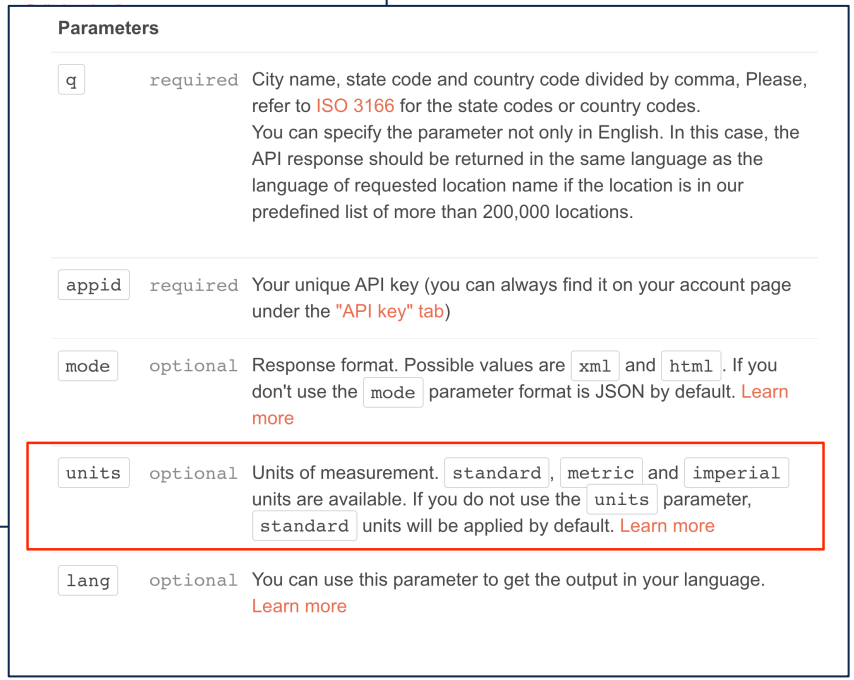
Goal:

- Fetch current weather in a given city
- Return a structured workflow output

How:

- Openweathermap.org
- Build and map Custom Data Types (CDTs)
- Use the Task Designer and the REST API executor and map CDTs
- Create a workflow that uses the CDT as input and the custom task





Input/Outputs

```
→  ~ > curl 'https://api.openweathermap.org/data/2.5/weather?q=rome,it&
units=metric&appid=a0dddc1224b1a18ef1346d50cc9711f1'
{"coord":{"lon":12.4839,"lat":41.8947},"weather":[{"id":802,"main":"Clo
uds","description":"scattered clouds","icon":"03d"}],"base":"stations",
"main":{"temp":27.57,"feels_like":28.65,"temp_min":24.65,"temp_max":30.
07,"pressure":1014,"humidity":58},"visibility":10000,"wind":{"speed":1.
34,"deg":342,"gust":5.81},"clouds":{"all":40},"dt":1623682231,"sys":{"t
ype":2,"id":2037790,"country":"IT",→  ~ >→  ~ >→  ~ >→  ~ >→  ~ >→  ~ >
→  ~ >
```

User Inputs:

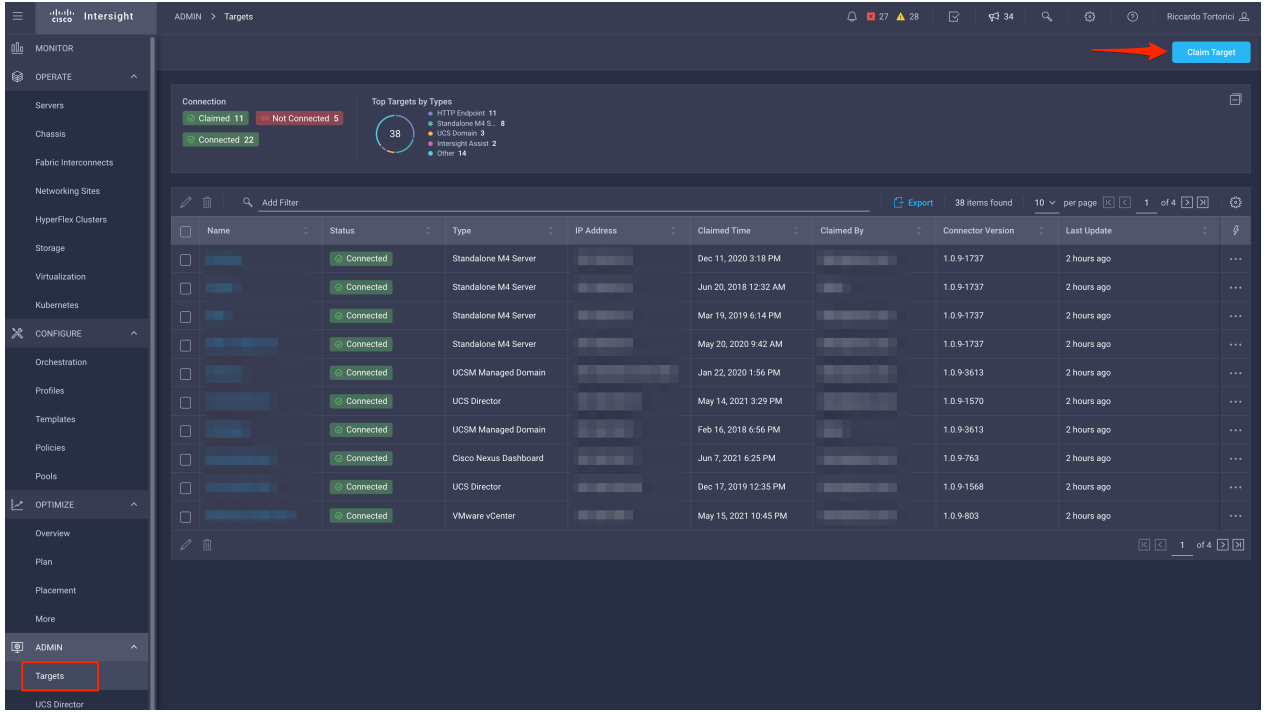
- City
- Country
- Units (metric, imperial)

Desired Outputs:

- Weather
- Temperature


```
{
  "coord": {
    "lon": 12.4839,
    "lat": 41.8947
  },
  "weather": [
    {
      "id": 802,
      "main": "Clouds",
      "description": "scattered clouds",
      "icon": "03d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 27.57,
    "feels_like": 28.65,
    "temp_min": 24.65,
    "temp_max": 30.07,
    "pressure": 1014,
    "humidity": 58
  },
  "visibility": 10000,
  "wind": {
    "speed": 1.34,
    "deg": 342,
    "gust": 5.81
```

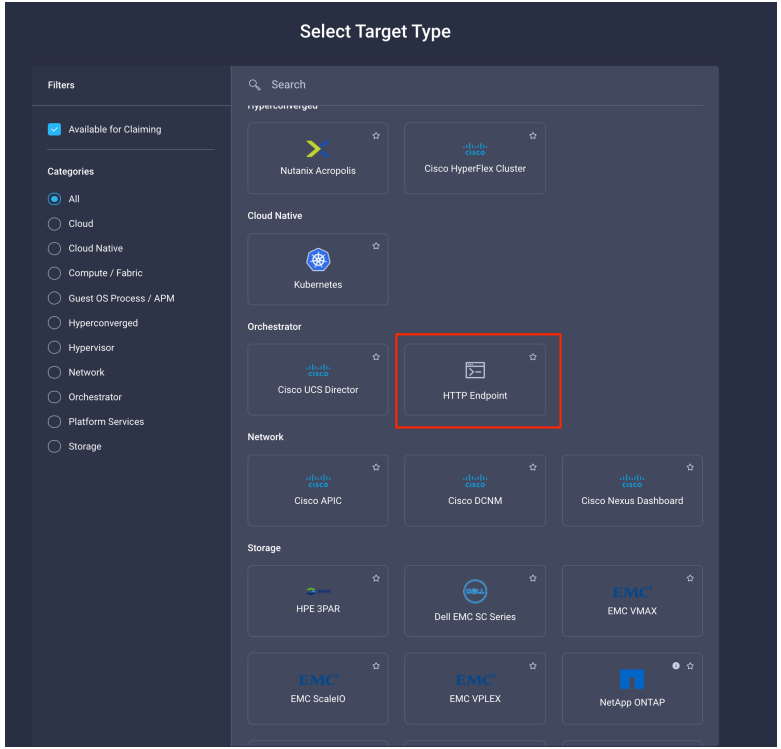
Claim the Target in Intersight

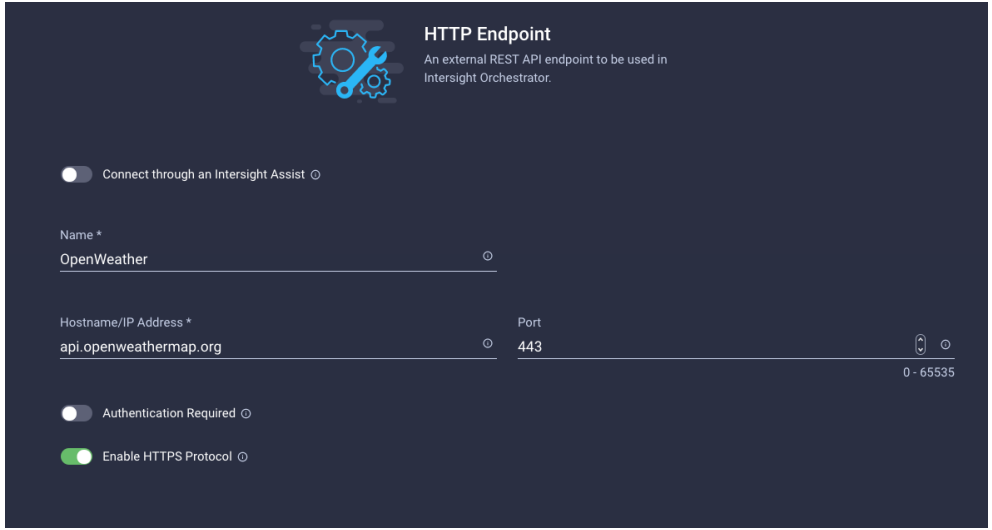


In order to execute automation against any target, you need to claim it.

In our example it will be https://api.openweathermap.org

Claim the Target in Intersight





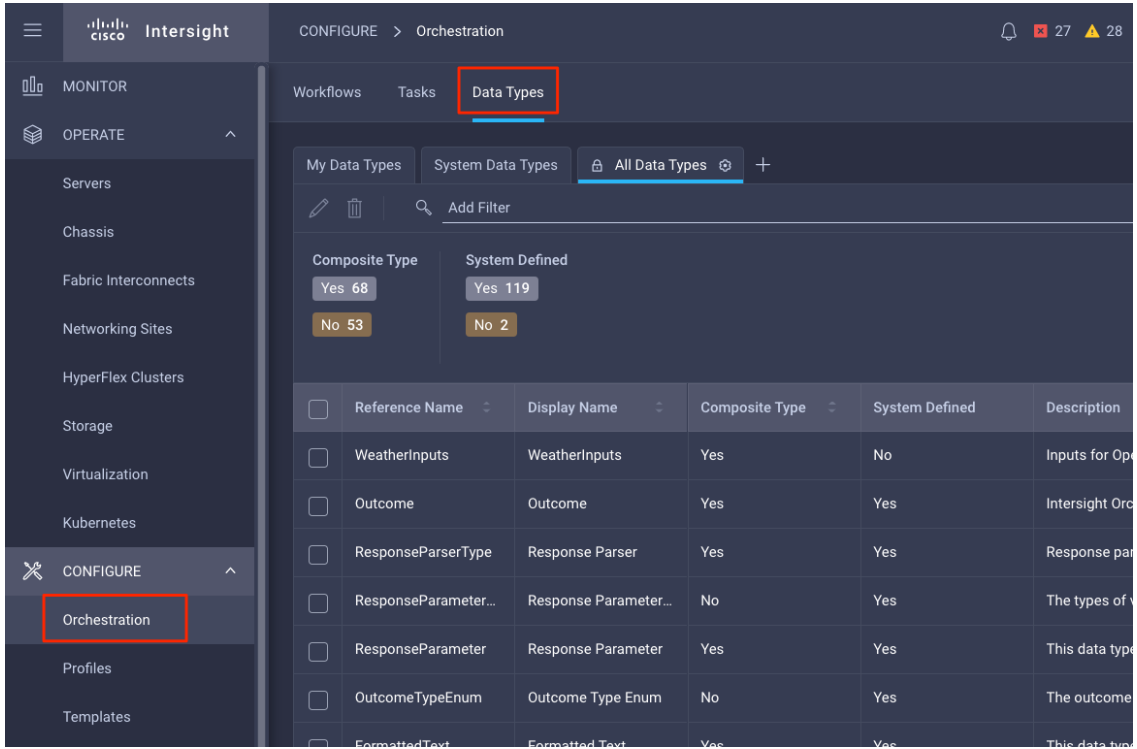
We don't need to connect through the Assist as the target endpoint is reachable from intersight.com

Once done, click **Claim**

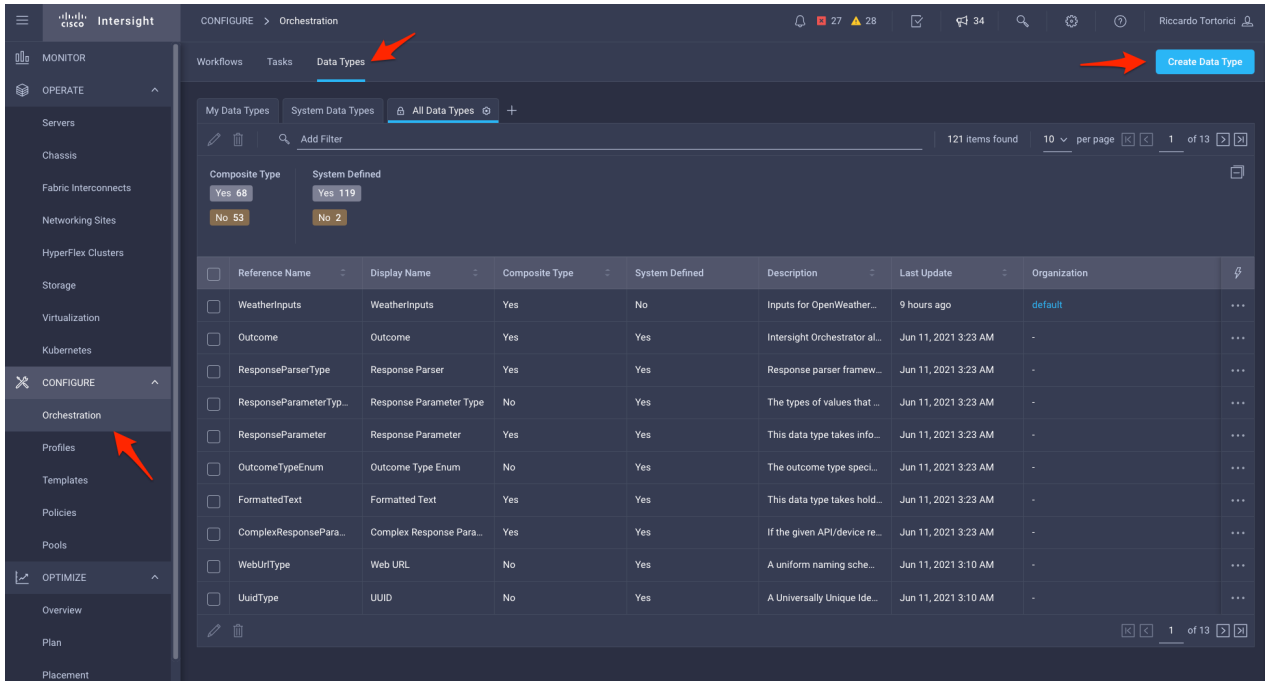
Custom Data Types (CDTs)

Custom Data Types

- Define types and associated constraints and validations
- Needed when no other data type fits (i.e. string, int, etc.)
- Can be used as workflow inputs



Create a Custom Data Type

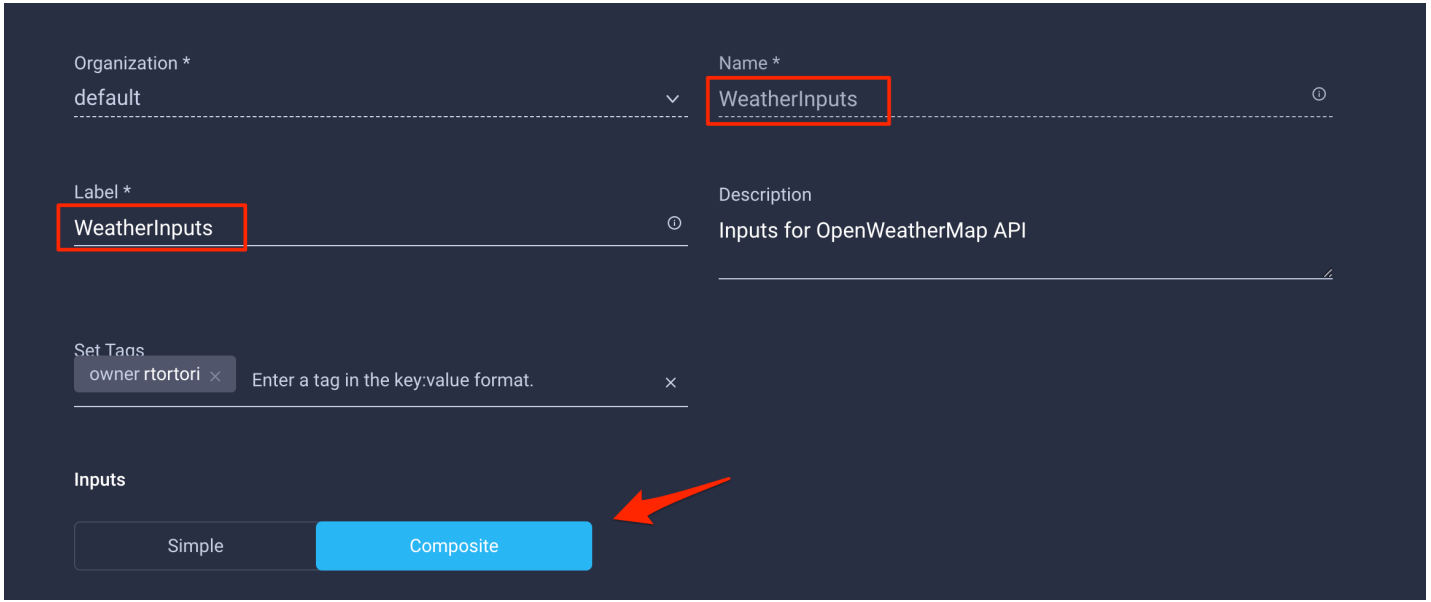


We create a new data type that can be used in our main workflow input.

In this case, this new custom data type will include the necessary data to built the API call to openweathermap.org:

- City
- Country
- Units

Create a Custom Data Type

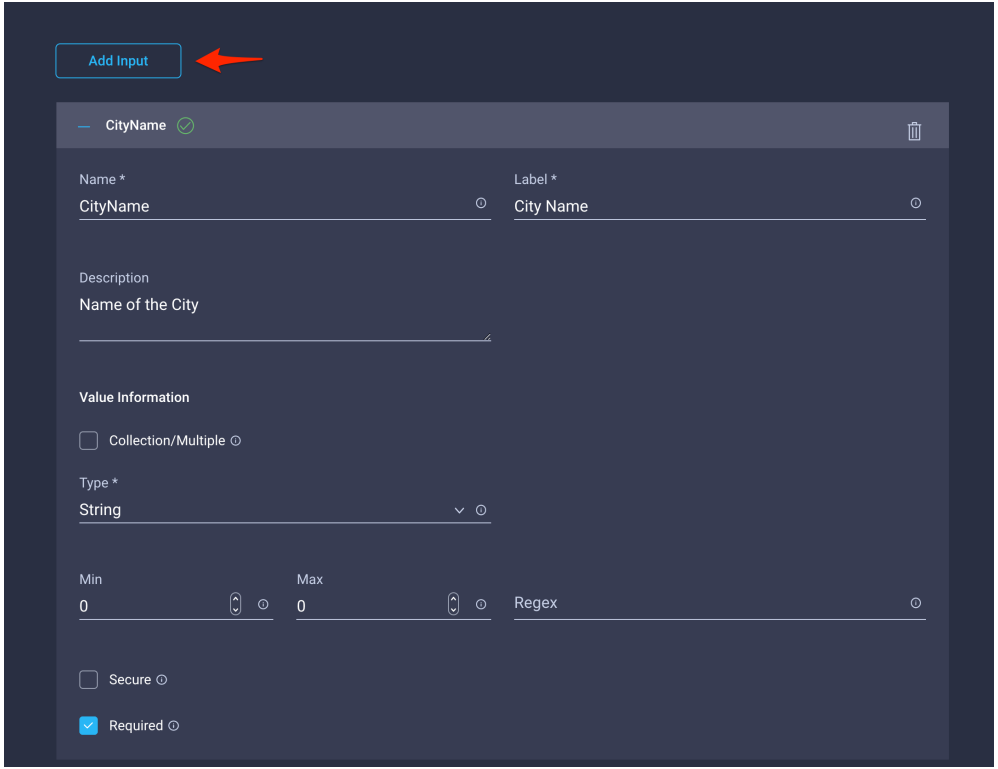


We specify a name, a label and a description.

Simple Input:
A simple definition of input. Can be one or a collection of String, Integer, Float, Boolean, JSON or enum

Composite Input:
Multiple combinations of the above

Create a Custom Data Type

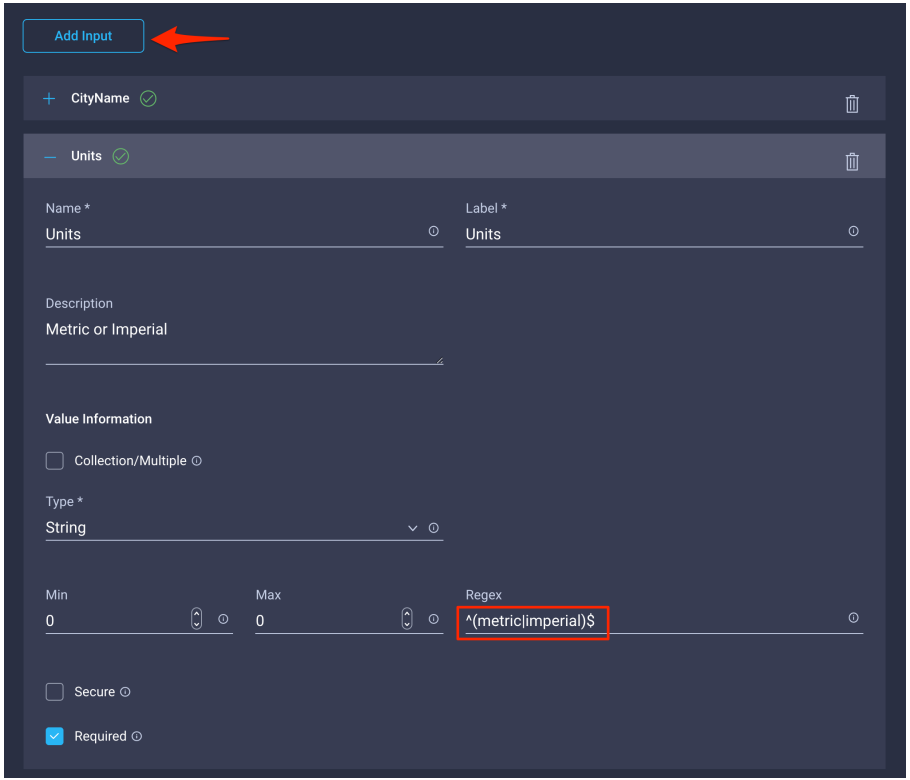


The first input will be the name of the city we want the current weather of (CityName)

Type is ‘String’, we do not enforce any constraints/validations (empty regex)

Marked as ‘Required’

Create a Custom Data Type

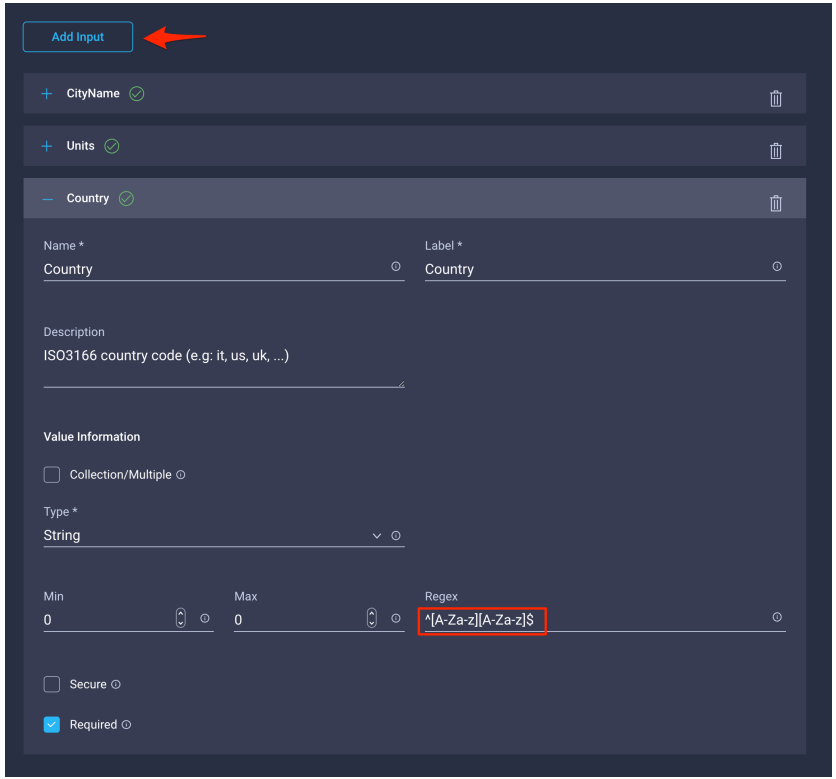


The second input will be the units we want to use for the temperature (**Units**)

Type is ‘String’.
As units can be either ‘metric’ or ‘imperial’ we enforce validation with a regex. Any other value will be invalid.

Marked as ‘Required’

Create a Custom Data Type



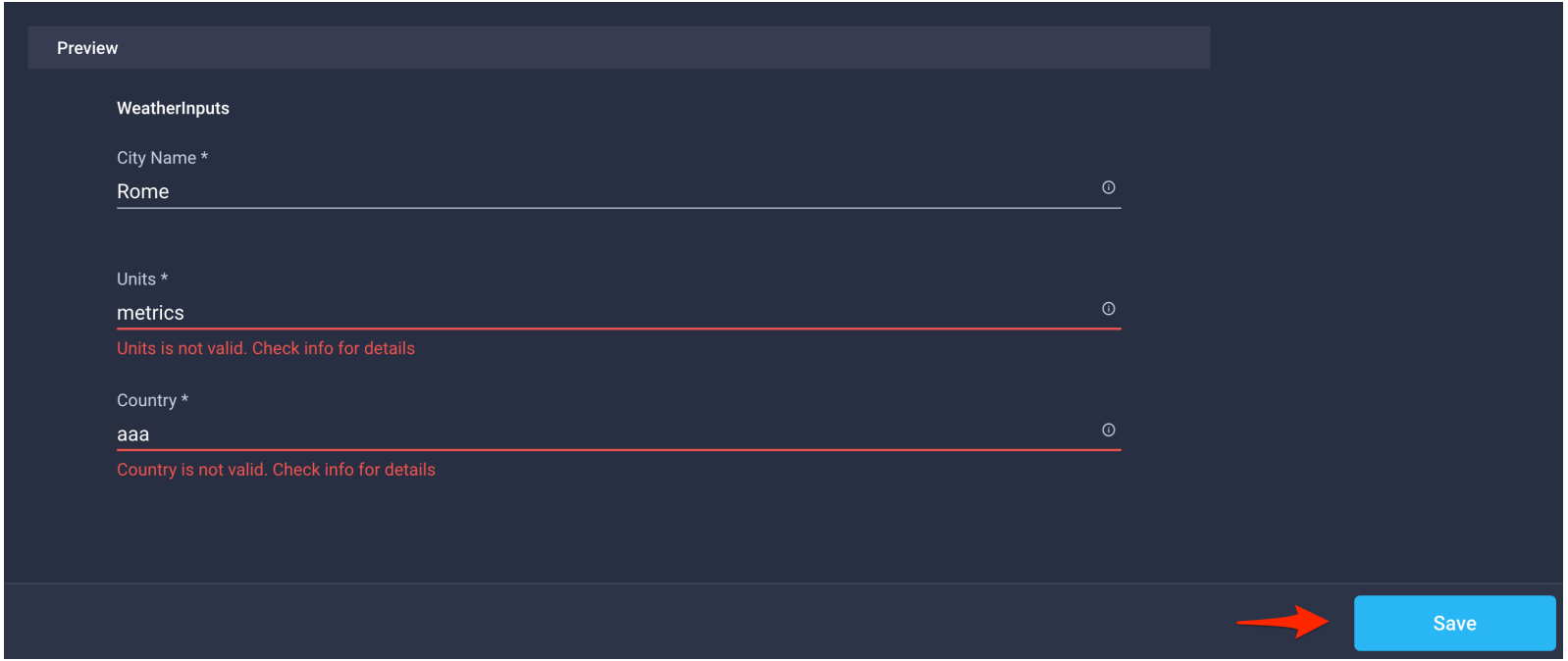
The third input will be the country code of the city we are requesting the weather of (Country)

Type is ‘String’.
As we require the country code to be ISO3166 compliant, we enforce validation with a regex. Only two alphabetic characters (case insensitive) will be accepted.

Marked as ‘Required’

Ref. (Alpha2): https://www.iso.org/obp/ui/#iso:pub:PUB500001:en

Preview a Custom Data Type

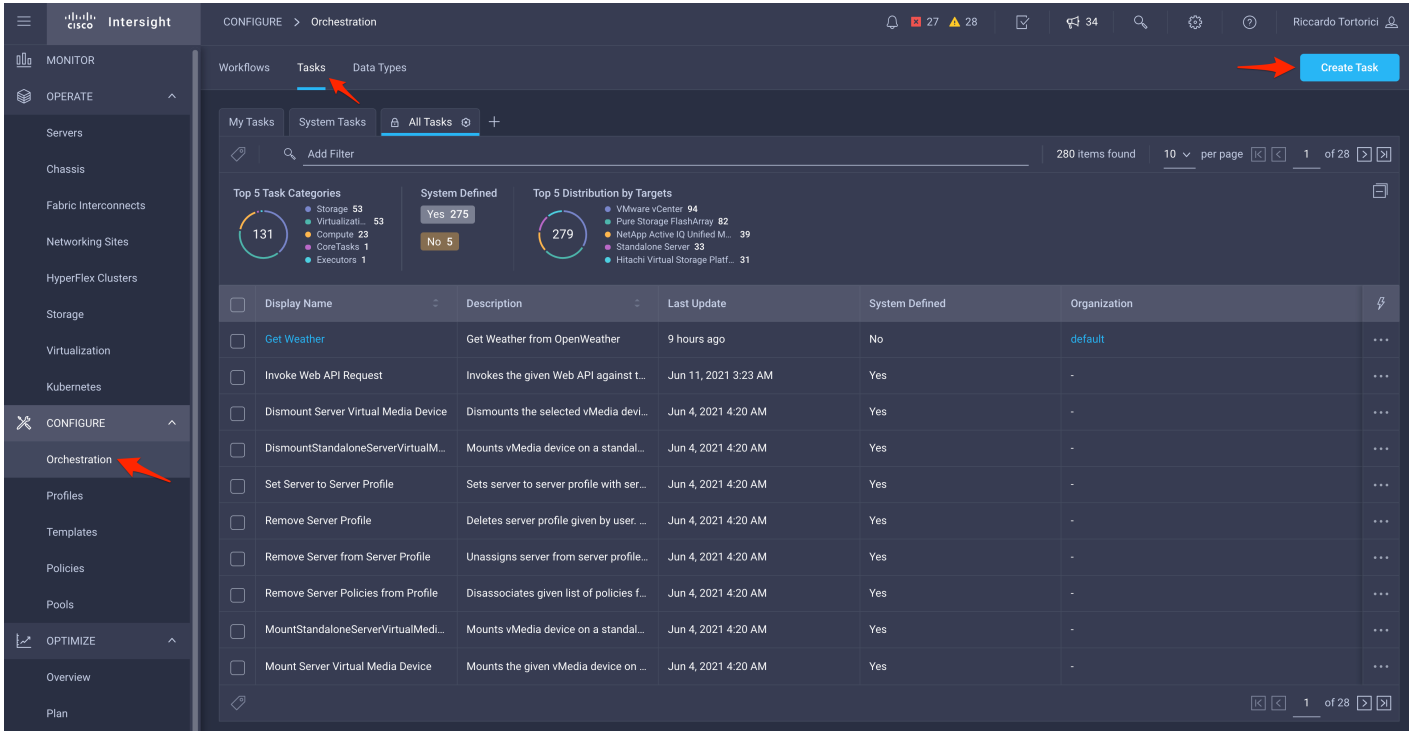


The Preview section allows the user to test their validation regex. Hit ‘Save’ when done

Task Designer

Web API Executor and Outcomes

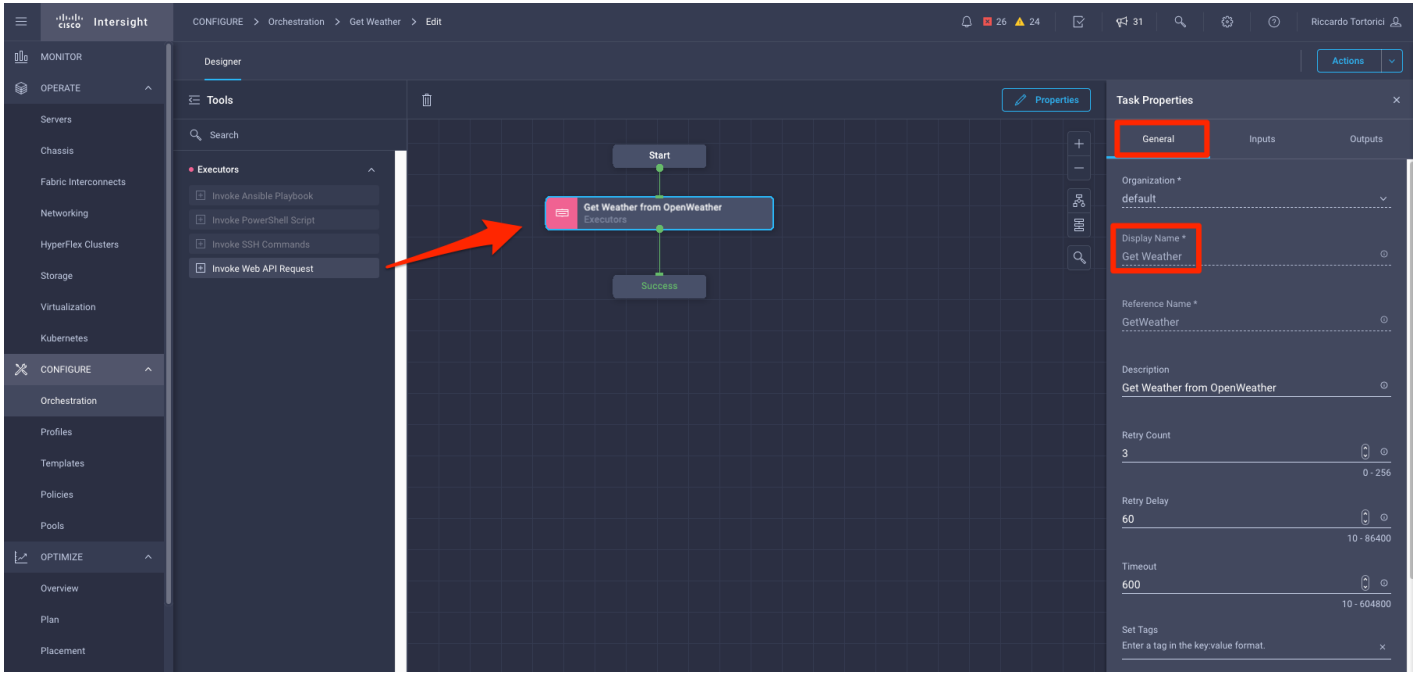
Create a Custom Task in the Task Designer



This custom task will be responsible to query openwheather and fetch the weather details.

It will use the WeatherDetails CDT and can be reused in any workflow

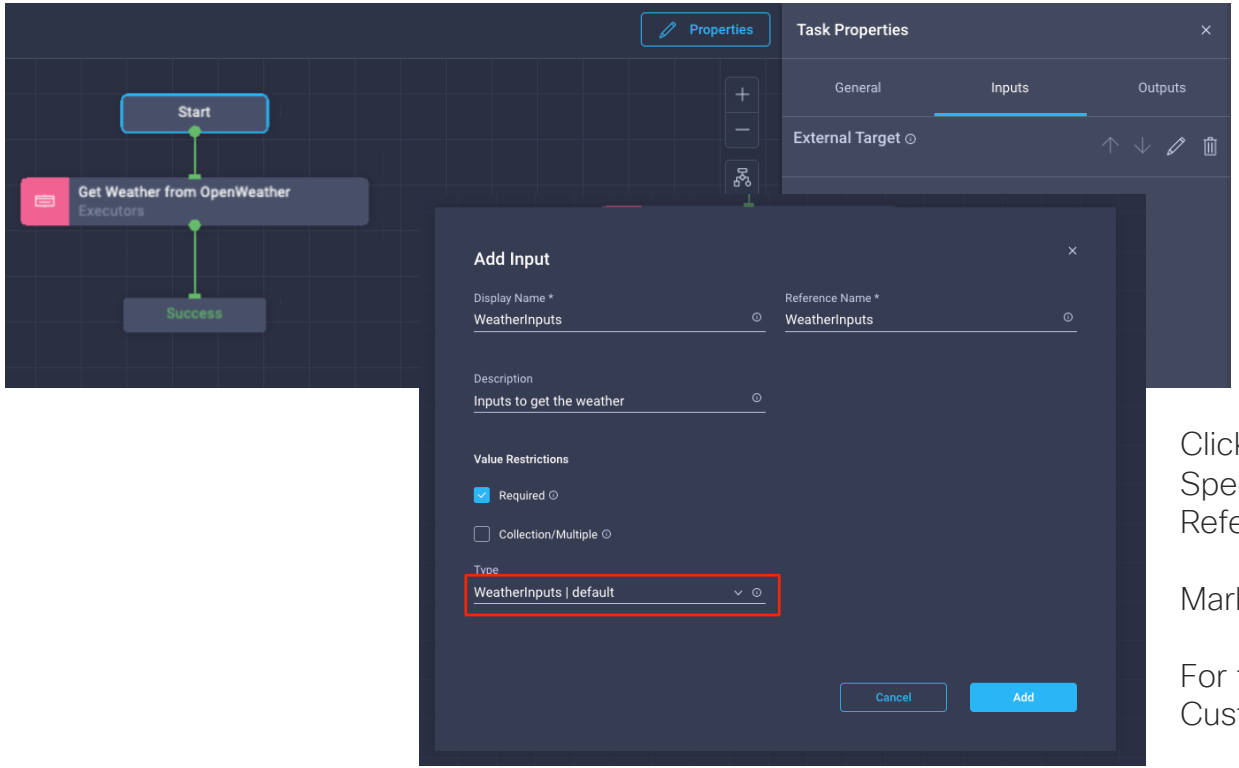
Create a Custom Task using the Web API Executor 1/9



Drag the web API Request task in the central pane and connect the 'Start' and 'Success' blocks to it.

Select an organization and give the task a name (**Get Weather**).

Create a Custom Task using the Web API Executor 2/9



Drag the web API Request task in the central pane and connect the ‘Start’ and ‘Success’ blocks to it.

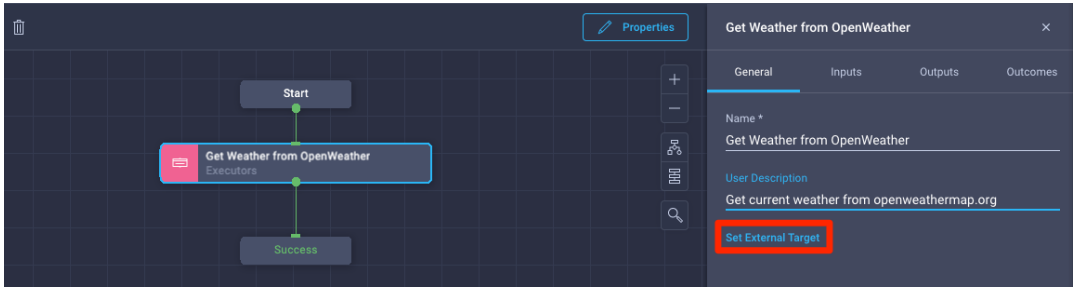
Select an organization and give the task a name (**Get Weather**).

Click ‘Create Task Input’.
Specify a Display Name as well as a Reference name (**WeatherInputs**).

Mark it as ‘Required’.

For the data type, we are going to use our Custom Data type **WeatherInputs**

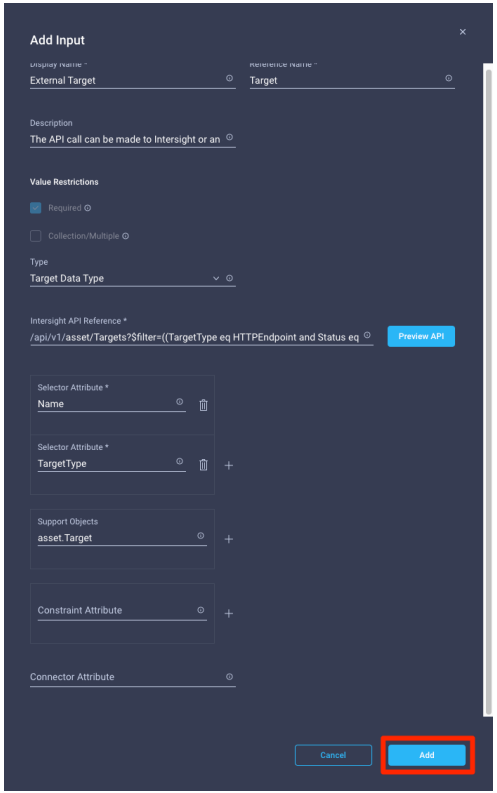
Create a Custom Task using the Web API Executor 3/9



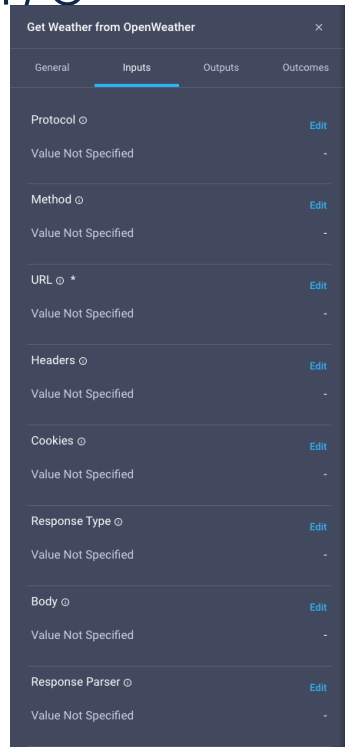
Click on the executor task, then click on ‘Properties’. Give the executor a name (**Get Weather from OpenWeather**) and set the external target. This option will allow to execute API calls against external endpoints as by default those calls will hit the Intersight API.

Click **Add** to confirm the default Target Types, this will include all valid targets you can run API calls against.

These are currently: HTTP endpoint, Pure Storage, NetApp, Hitachi, Terraform Cloud, vCenter, MDS, Nexus and Cisco APIC



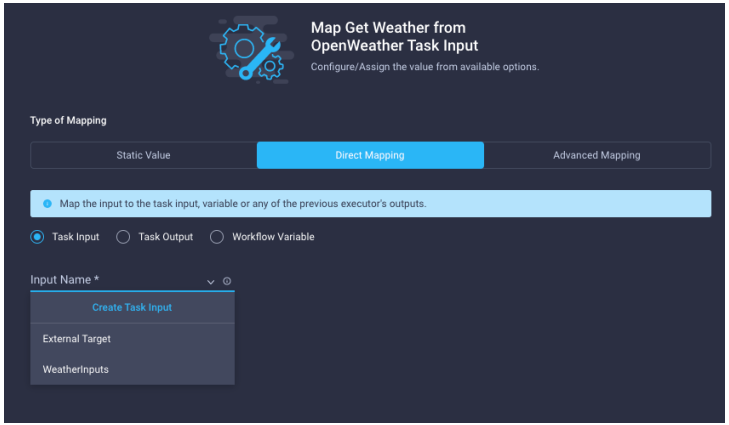
Create a Custom Task using the Web API Executor 4/9

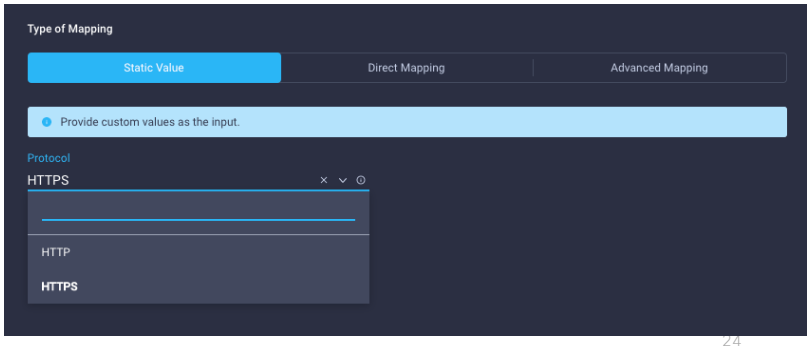


Inputs Tab

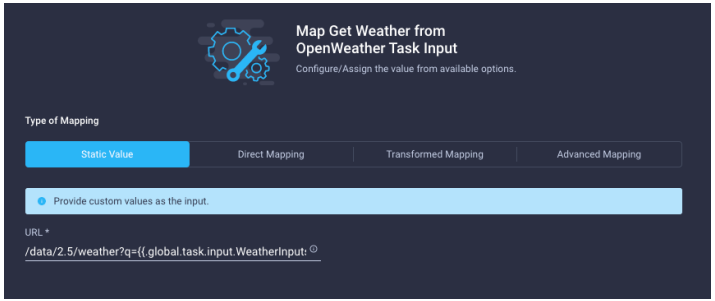
Click on ‘edit’ to set a value for each input of this task. The ‘i’ icon will provide detailed information of each input as well as the default value.

When you click on ‘edit’, you can either map an existing task input, create a new one or set a custom value





Create a Custom Task using the Web API Executor 5/9



Edit the 'URL' input. This is a relative URL that gets appended to the target (in this case openwheather), not an FQDN.

We set a **Static Value**.
We want to replicate exactly the same call we did with curl:

```
$ curl 'https://api.openweathermap.org/data/2.5/weather?q=rome,it&units=metric&appid=a0dddc1224b1a18ef1346d50cc9711f1'
```

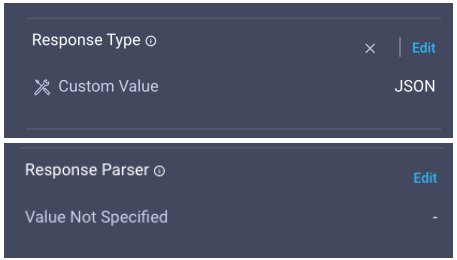
Fill in with the following (replace with your own OpenWeather API key):

```
/data/2.5/weather?q={{.global.task.input.WeatherInputs.CityName}},{{.global.task.input.WeatherInputs.Country}}&units={{.global.task.input.WeatherInputs.Units}}&appid=a0dddc1224b1a18ef1346d50cc9711f1
```

Everything between {{ }} will be replaced dynamically:
.global.task.input.WeatherInputs.CityName

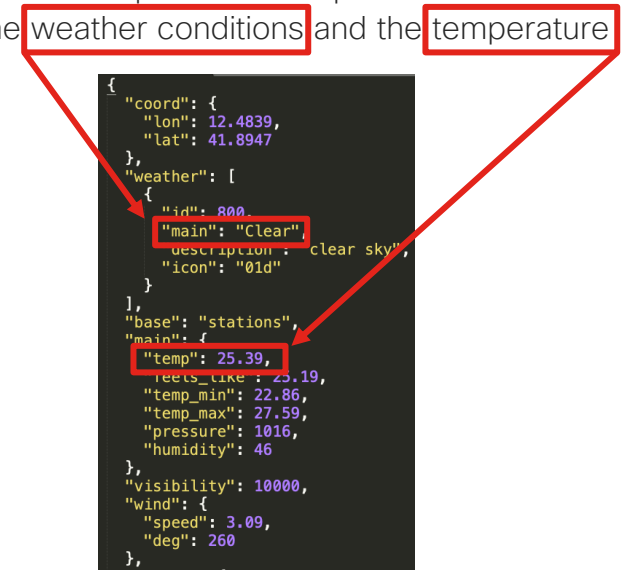
One of the custom task input
Name of the Input
Key in the input

Create a Custom Task using the Web API Executor 6/9



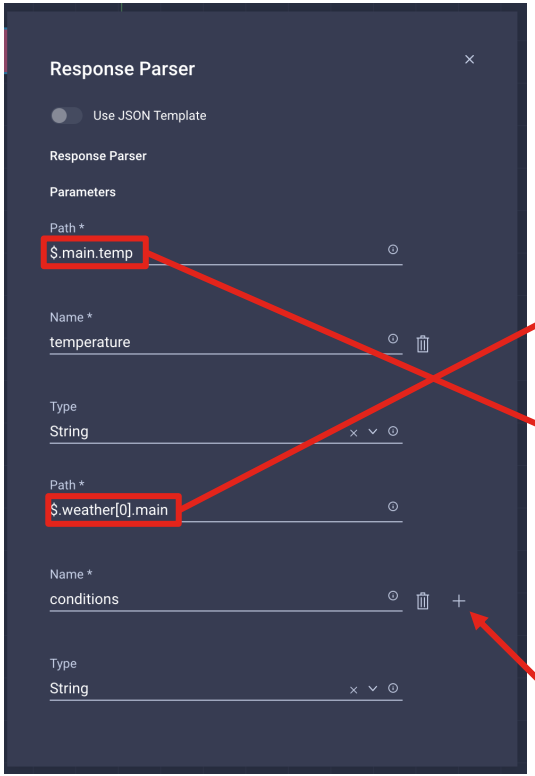
Set the ‘Response Type’ to JSON

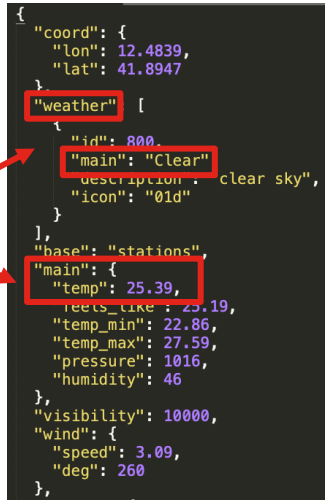
The response parser is going to extract and manipulate the response from the target. In our case we want to extract the weather conditions and the temperature



Openweathermap response example

Create a Custom Task using the Web API Executor 7/9





Set the jsonpath of the value you want to extract from the response.

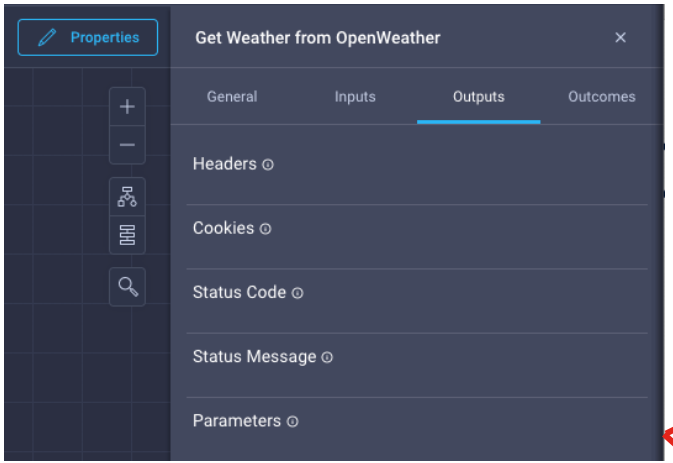
In the example:

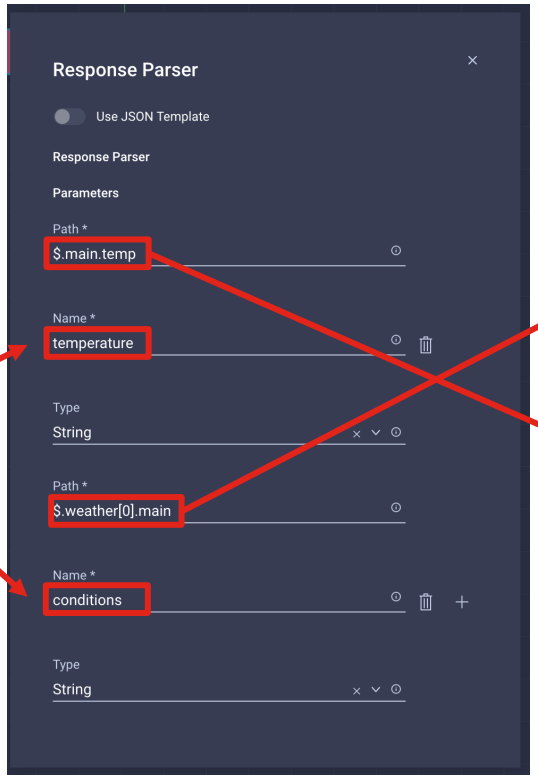
1. $.main.temp will extract 25.39 as a string and set this value to the parameter task output with 'Name' = **temperature**

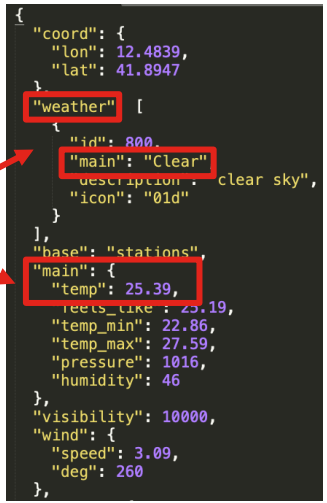
2. $.weather[0].main will extract 'Clear' from the 'main' key on the first array element [0] of the 'weather' key. The value will appear in the parameter task output with 'Name' = **conditions**

Click the plus (+) sign to add more parameters

Create a Custom Task using the Web API Executor 8/9

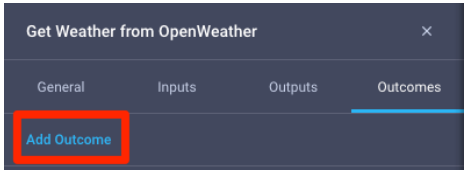


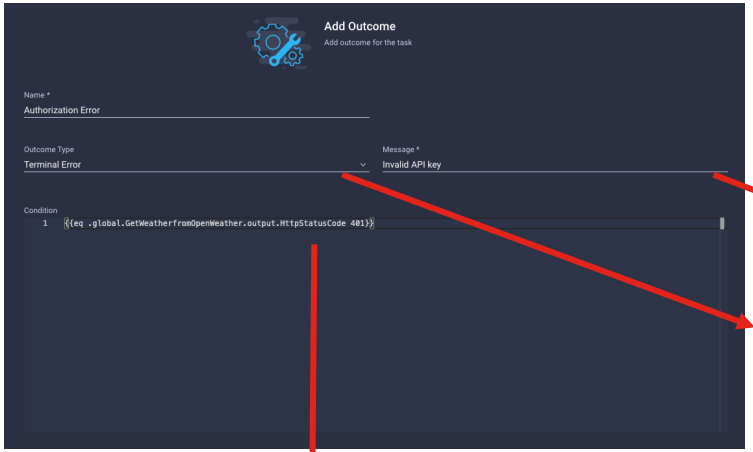




In the task output tab, there is a list of predefined Outputs. The response parser will put the extracted values using the names specified. In our example, temperature and conditions respectively.
These are normal task outputs and can be leveraged elsewhere in ICO

Task Outcomes





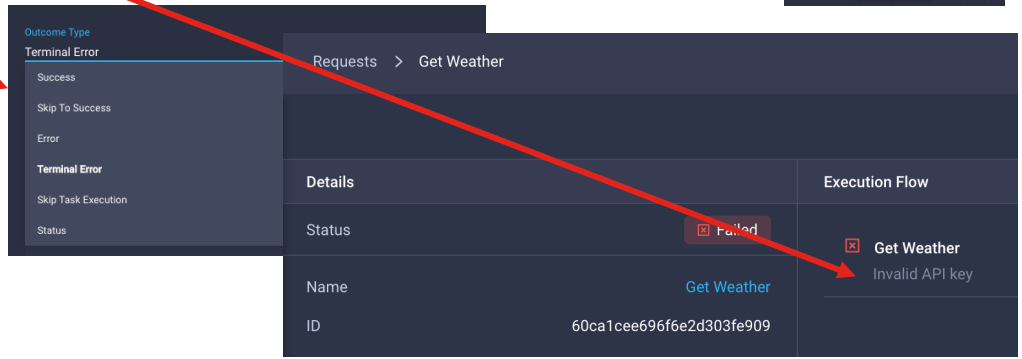
{{eq .global.GetWeatherfromOpenWeather.output.HttpStatusCode 401}}

A task outcome defines how the task behaves based on configurable conditions.

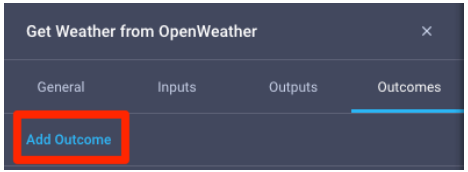
In the example:
{{eq .global.GetWeatherfromOpenWeather.output.HttpStatusCode 401}}

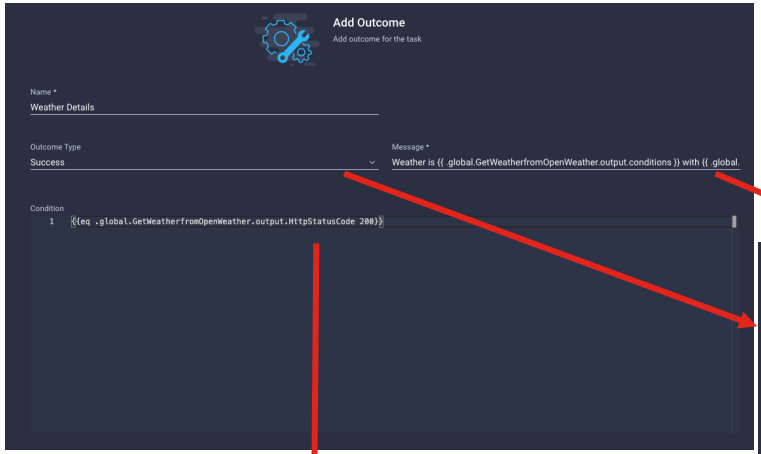
Checks whether the HttpStatusCode task out is 401. If that's the case, we return a **Terminal Error** with the custom **Message** "Invalid API key"

Where **GetWeatherfromOpenWeather** is the name of the task with no spaces.



Task Outcomes





{{eq .global.GetWeatherfromOpenWeather.output.HttpStatusCode 401}}

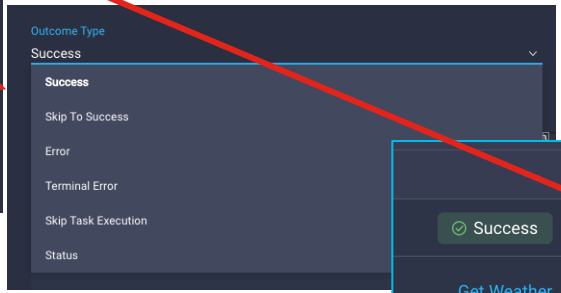
We are going to add one more task outcome.

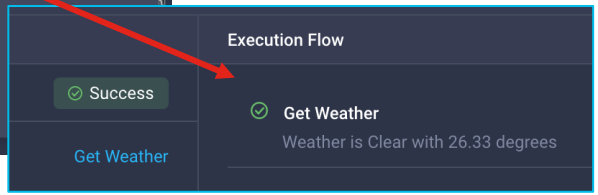
In the example:
{{eq .global.GetWeatherfromOpenWeather.output.HttpStatusCode 200}}

Checks whether the HttpStatusCode task out is 200. If that's the case, we succeed with the custom **Message** "**Weather is {{ .global.GetWeatherfromOpenWeather.output.conditions }} with {{ .global.GetWeatherfromOpenWeather.output.temperature }} degrees**"

We can use the task outputs to customise the Message.
Assuming the output **conditions** to be "Clear" and the **temperature** to be "26.33" the message would look like this:

"Weather is Clear with 26.33 degrees"





Additional Outcome Expression Examples

Let's assume the sample scenario where you want to make a task **fail** if the server returns an empty response (example: `"[]"`)

As of February 2022, ICO doesn't store the full response by default.

You will need to create a parameter from the **Response parser** that stores the full response using '$' as JSONPATH.

In this example, it will be called **fullResponse**

Case 1 – Full Response as String

Here we want to trigger the outcome if the returned payload string is equal to `"[]"`

Outcome Condition

Type: Terminal Error
Message: 'Response is empty'
Condition:

```
{{eq .global.InvokeGenericWebApi1.output.fullResponse "[]"}}
```

Case 2 – Full Response as JSON

Here we want to trigger the outcome if the returned payload has zero elements in the array

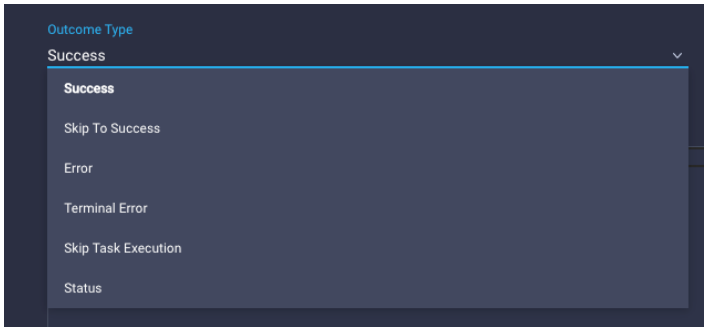
Outcome Condition

Type: Terminal Error
Message: 'Response is empty'
Condition:

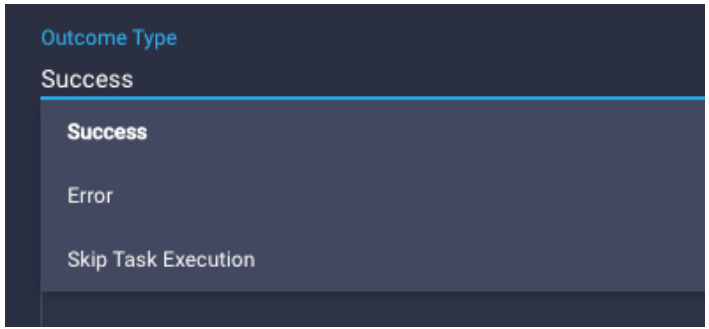
```
{{eq (len .global.InvokeGenericWebApi1.output.fullResponse) 0}}
```

Outcomes Deep Dive

Depending whether you are using an executor as an embedded task or you are building a reusable custom task, you will have different outcome types available.



Custom Task

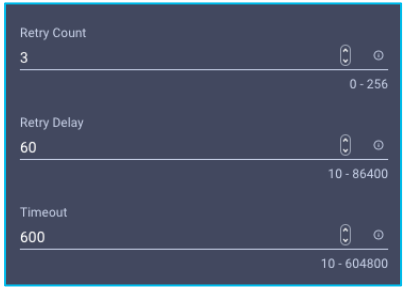


Embedded Task

Retry Parameters

Outcomes use retry parameters to define the behavior of a single executor or the whole custom task.
Retry parameters are available **only** for custom tasks created in the **Task Designer (General Tab)** and are predefined (not configurable) in the **Workflow Designer.**

Note: *Retries are **global** to the custom task, this means that the amount of retries are shared among all tasks in a custom task. If a single task consumes all retries, there will be no more retries left for subsequent tasks*



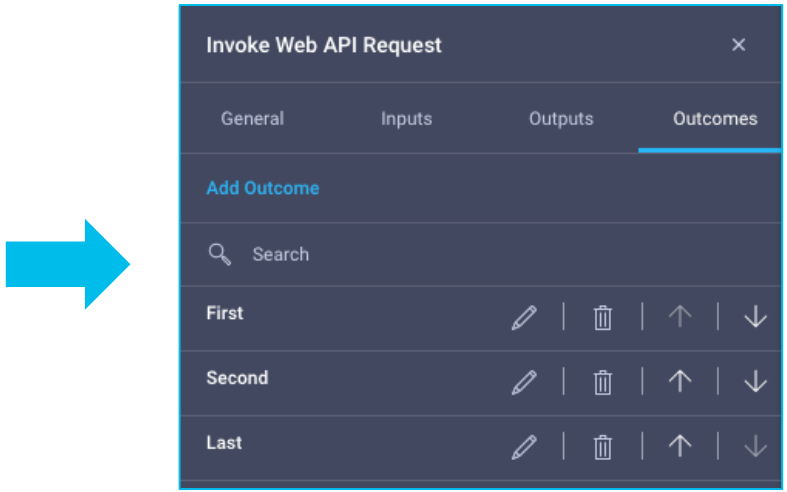
Name	Description	Default	Notes
Retry Count	The number of times a task should be tried before marking as failed	3	Not Configurable in Workflow Designer
Retry Delay	The delay in seconds after which the the task is re-tried	60	Not Configurable in Workflow Designer
Timeout	The timeout value in seconds after which task will be marked as timed out. Max allowed value is 7 days	600 (120 for embedded tasks)	Timeout is set to 120 in the Workflow Designer, not configurable by the user
Retry Policy	Can be Fixed (Retry happens after a specific amount of time) or BackOff	Fixed	Backoff* currently not supported (as of April 2022)

* Backoff will allow delay to increase between retries

Order of Operations

Outcomes are evaluated sequencially, order matters!

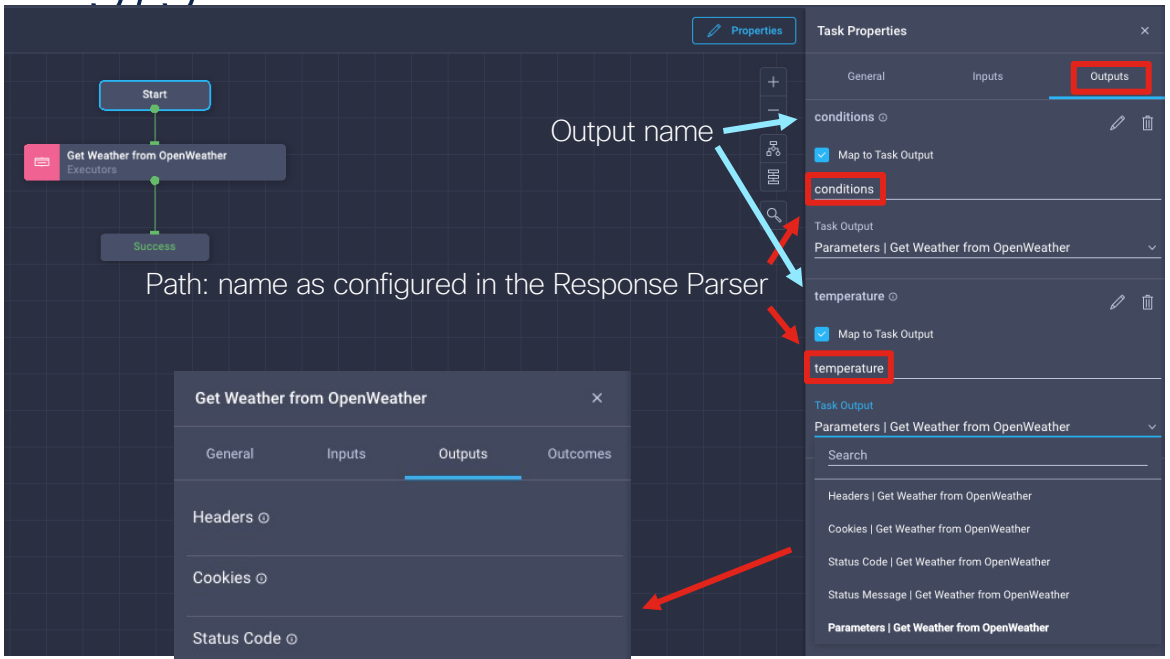
If no outcome will be matched, the task will follow its default behaviour, that is timing out or success.



Outcome Types

Outcome Type	Description	Availability
Success	Task will succeed if condition evaluates to true	TD and WD
Skip To Success	If the condition matches, all subsequent API in a custom task will be skipped and go directly to **Success** block	Task Designer Only
Error	If the condition matches the whole custom task will be retried assuming there are retries left	TD and WD
Terminal Error	If the condition matches, retry count will be ignore and the whole task will be marked as failed	Task Designer Only
Skip Task Execution	If the condition matches the whole task will be skipped	TD and WD
Status	If the condition matches the whole custom task will be retried (assuming there are retries left), however the user will not be notified about the fail. A user would use the **status** outcome type when the target is returning a response representing an ongoing activity and they want to retry until the response matches a condition. For instance, retry an API that returns `{"Status": "Creating"}`until it returns `{"Status": "Completed"}`	Task Designer Only

Create a Custom Task using the Web API Executor 9/9



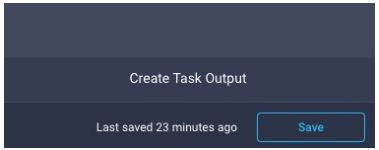
Task outputs

Click on 'Start', then 'Properties' to get the overall task (workflow) properties.

To set the overall task Outputs, click on the **Outputs** tab.

You can map outputs to task outputs.
Click on the **Task Output** dropdown menu, these are the task outputs that can be selected. Selecting **Parameters**, we can access the outputs created by the response parser.

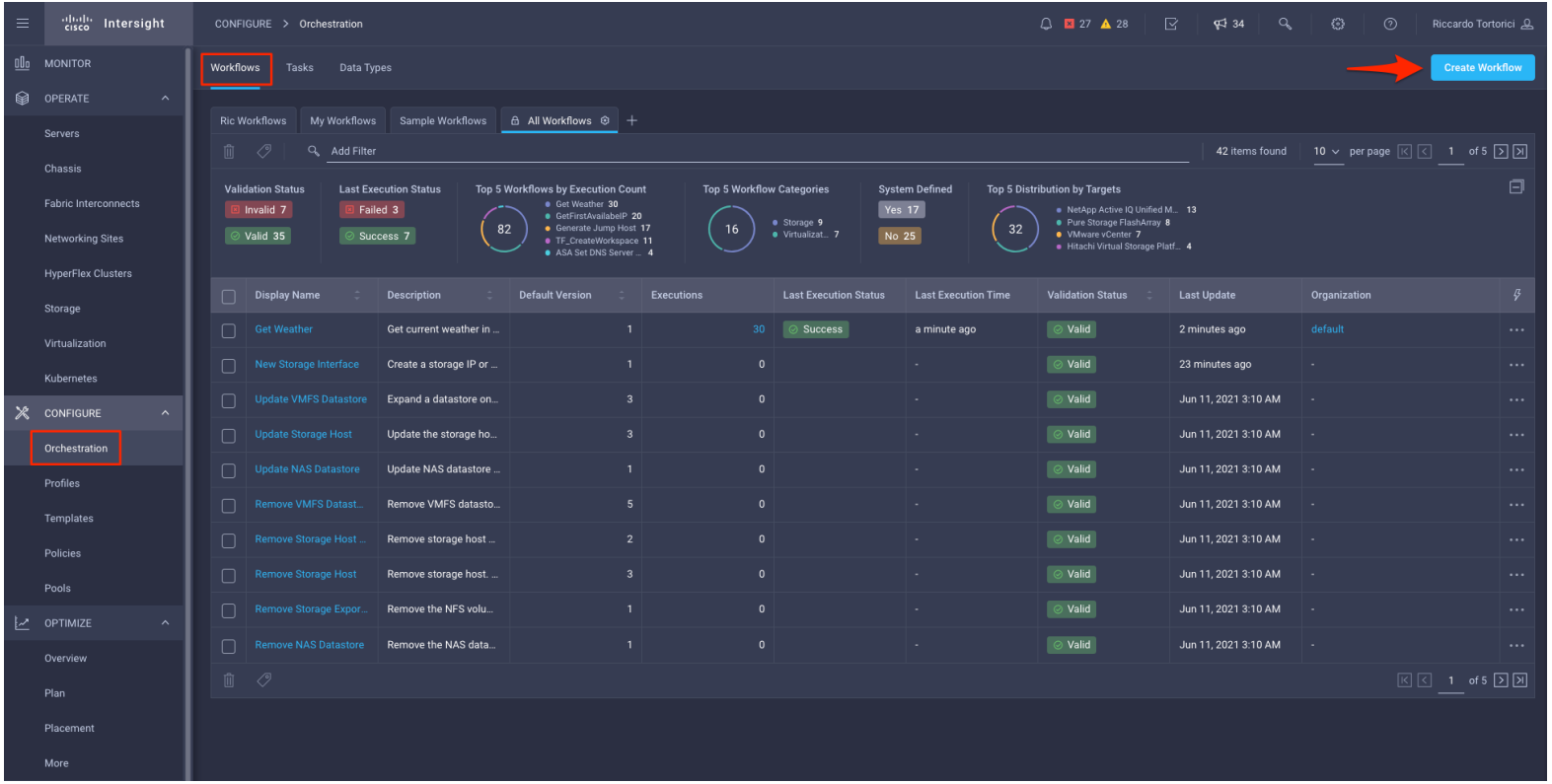
Create outputs for both **conditions** and **temperature**.
In this example, the name of the outputs matches the keys we used to represent the values extracted by the response parser



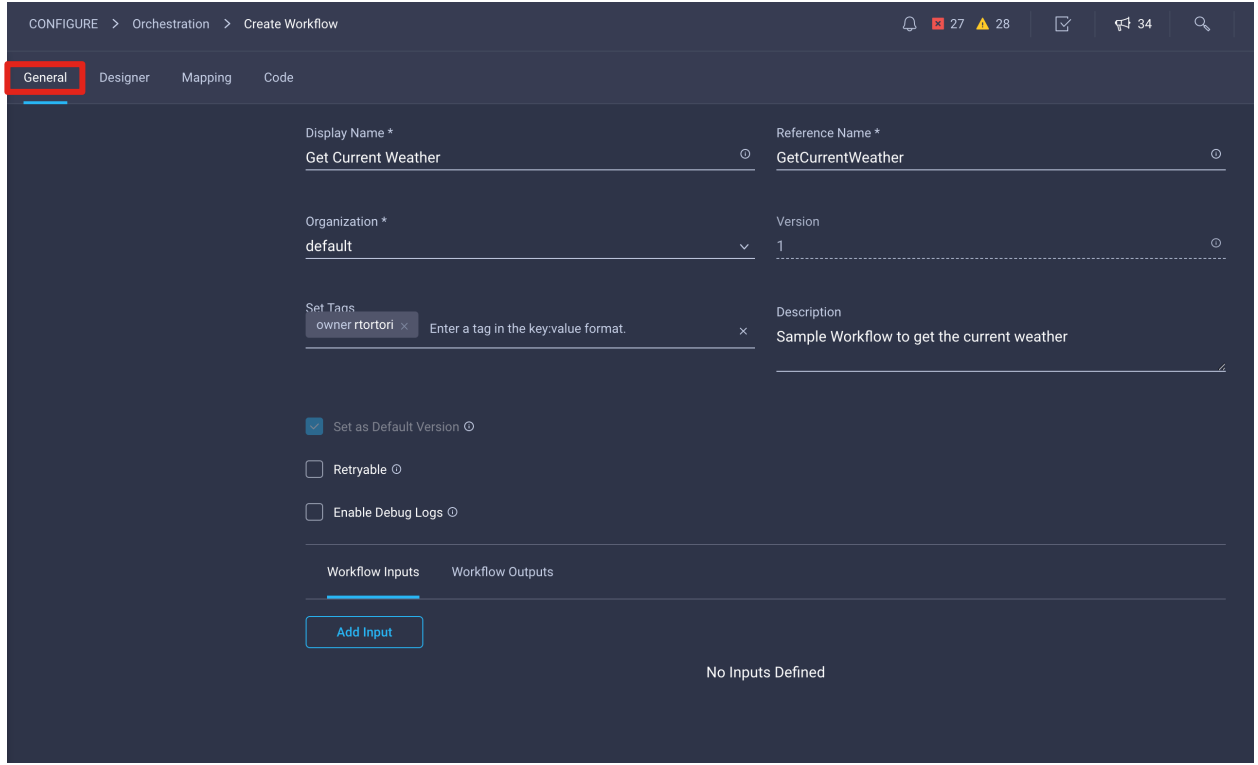
When done, click the **Save** button

Workflow Designer

Create a Workflow



Create a Workflow



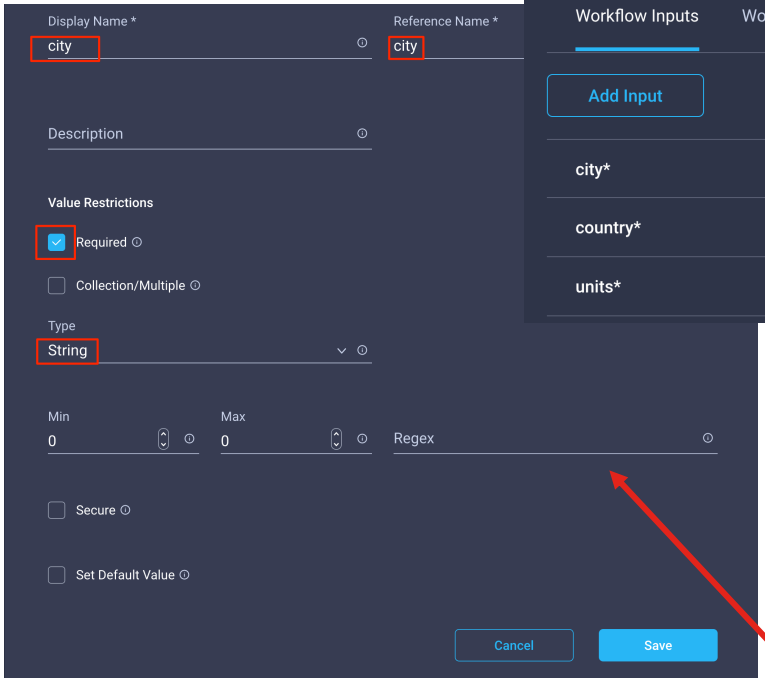
We are going to create a workflow that gets the current weather in a given city and country.

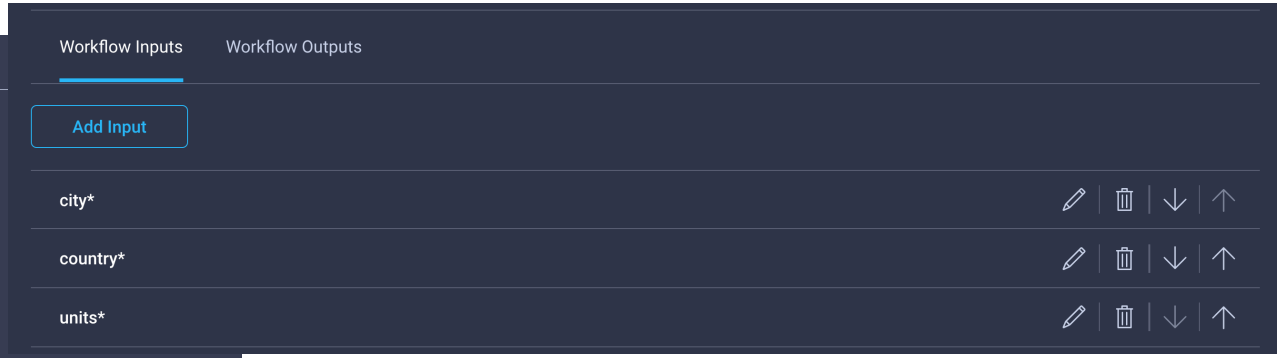
In the **General** tab, we give the workflow a **Display Name** as well as a **Reference Name** which will be used to reference this workflow within ICO.

Clicking on **Add Input,** we want to add 3 inputs:

- City
- Country
- Units

Create a Workflow



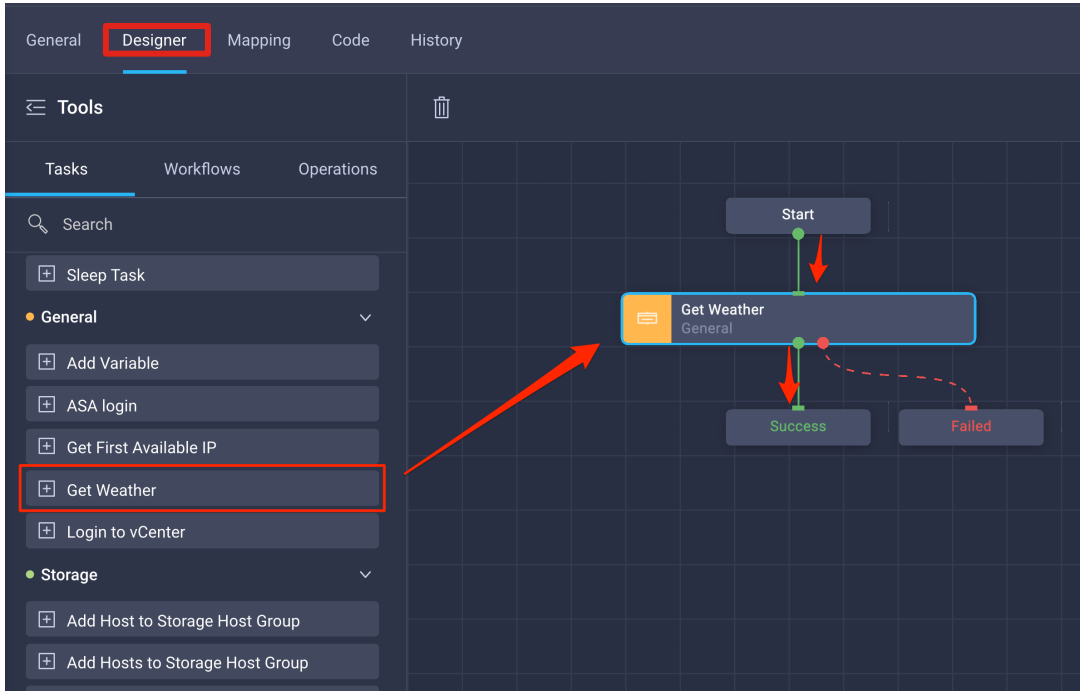


All three inputs are **Required** with **Type** = String
You can also set a **Default Value** if needed.

Note that currently (June 2021) the input validation for CDTs is enforced only if using Static Mapping. Direct Mapping of individual inputs to CDT parameters is not currently supported in the UI while advanced Mapping ignores CDT input validation

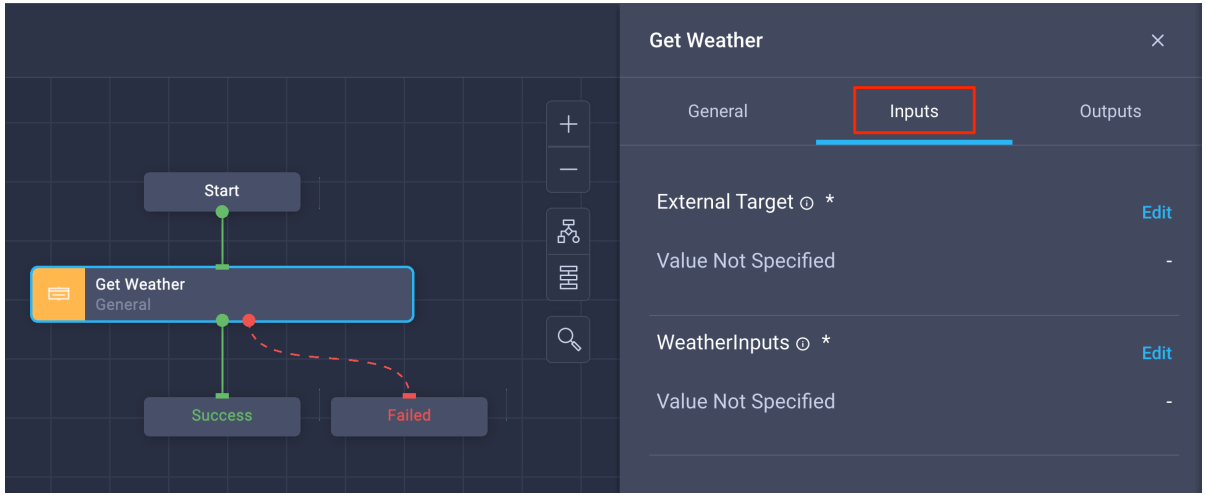
This means that for workflows requiring advanced mapping, the user is responsible to enforce regexp validation also at the workflow input level

Create a Workflow



In the **Designer** tab, drag the 'Get Weather' custom task we created in the palette.
Connect the **Start** block to the green connector of the custom task, as well as the **Success** block as shown.

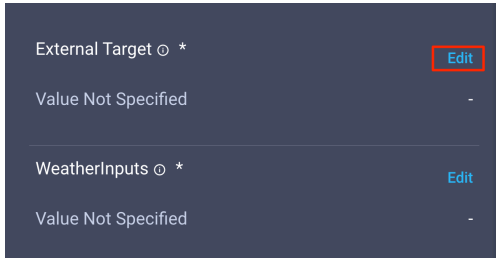
Create a Workflow

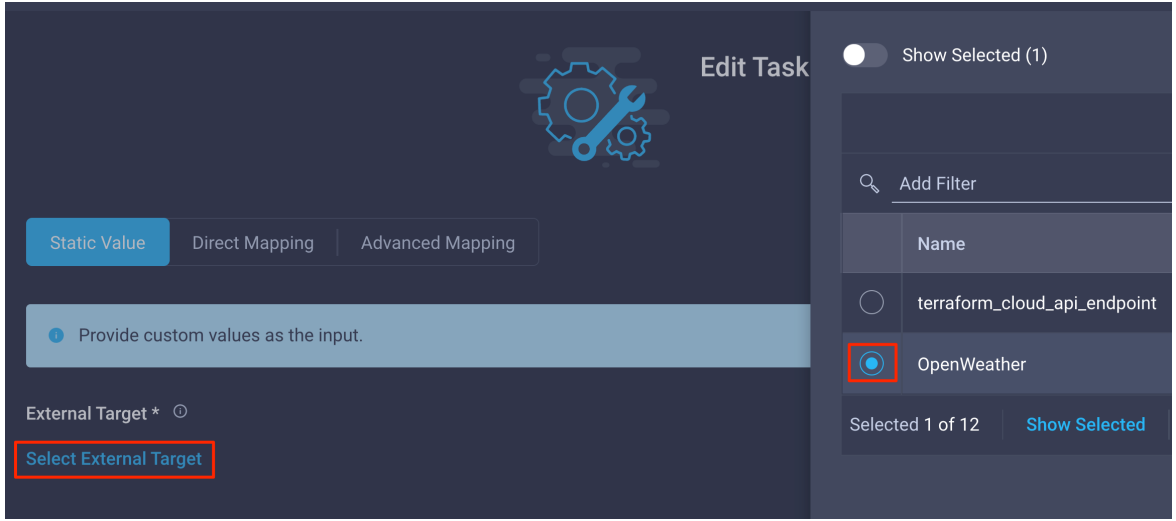


Clicking on the **Inputs** tab on the 'Get Weather' task properties, we can find two inputs are required:

1. The external API target endpoint
2. WeatherInputs, our Custom Data Type

Create a Workflow

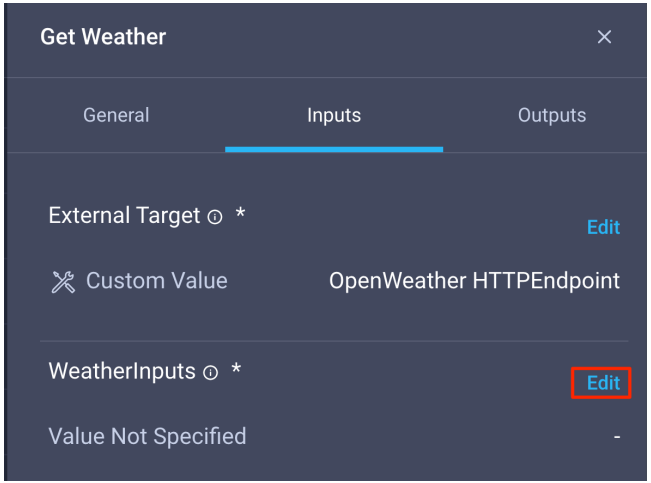


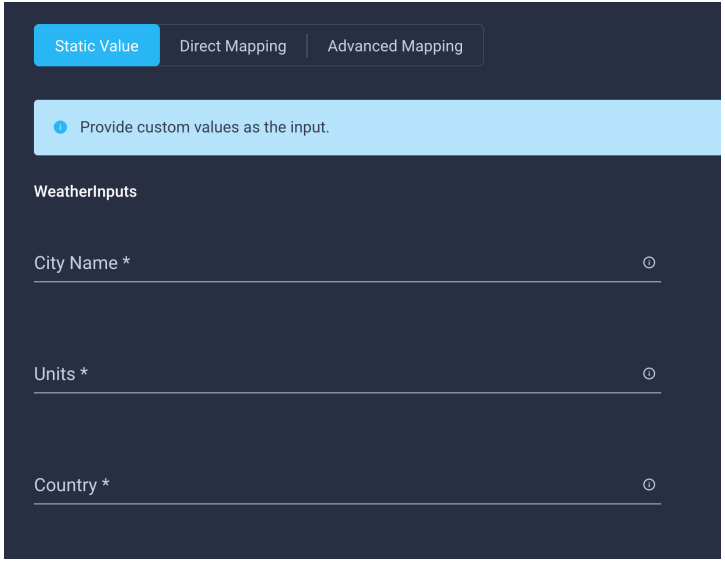


On **External** Target, click on **Edit.**
You can map the external target using a **Static Value** (the workflow will always execute against this target), **Direct Mapping** (map to an existing workflow input or task output) or **Advanced Mapping** (leverage Go templates for advanced mappings/transformations)

Click on **Select External Target** to statically map the input to the 'OpenWeather' target

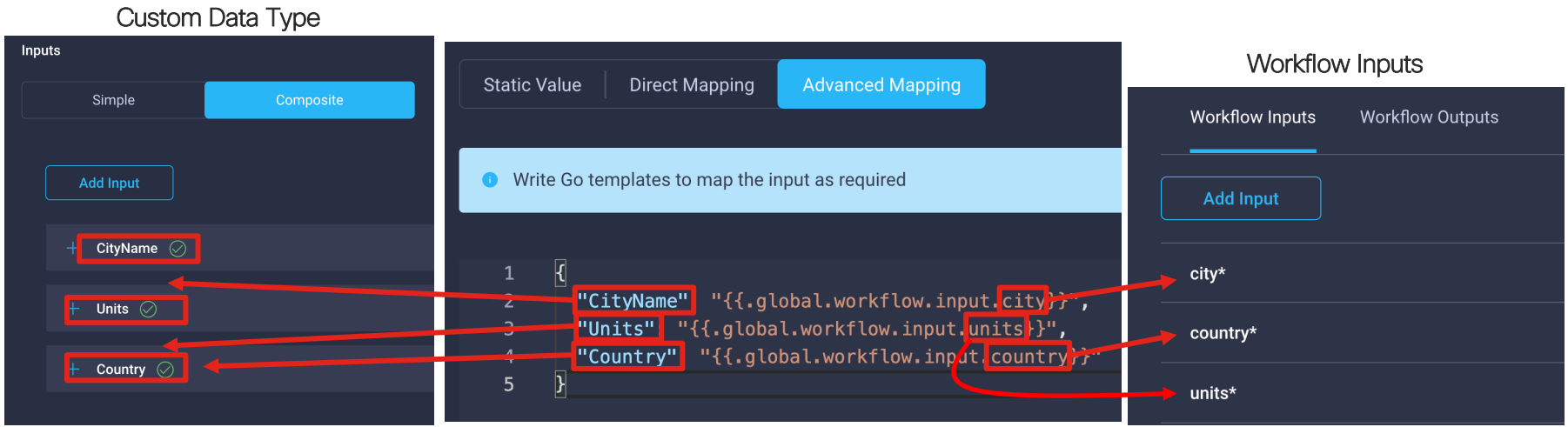
Create a Workflow





On **WeatherInputs**, click on **Edit.**
The default static value mapping will give you the opportunity to hardcode the three required values of the CDT. However, since we want them to be mapped to the workflow input (in order for the user to execute the workflow for multiple cities) and there is currently no way in the UI to map individual inputs to CDT properties, we are going to select **Advanced Mapping**

Create a Workflow



The WeatherInputs CDT we created has three properties:

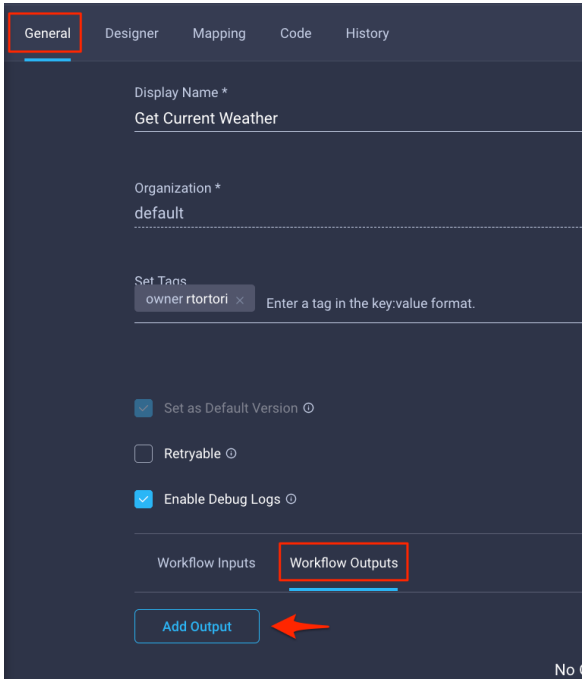
1. CityName
2. Units
3. Country

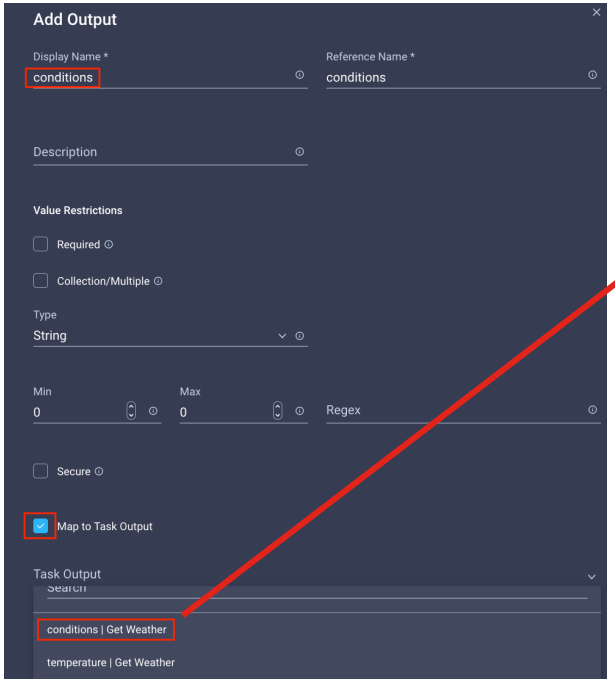
This Go Template maps each CDT parameter to specific workflow inputs.
Refer to the documentation for additional details on Template Parameters:

https://www.intersight.com/help/resources/web_api_request#template_parameters

```
{
"CityName": "{{.global.workflow.input.city}}",
"Units": "{{.global.workflow.input.units}}",
"Country": "{{.global.workflow.input.country}}"
}
```

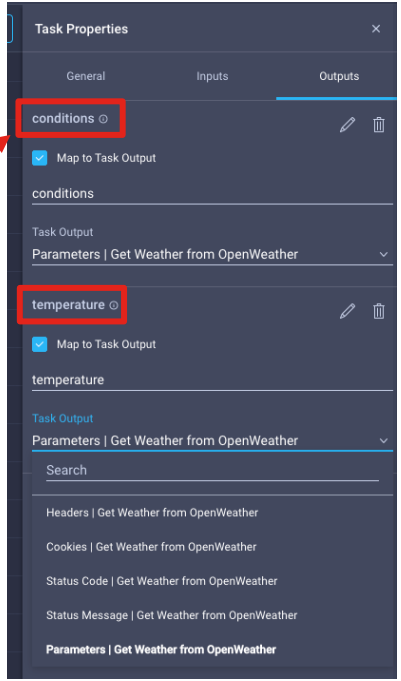
Create a Workflow





Recall: we have set two outputs on the custom tasks:

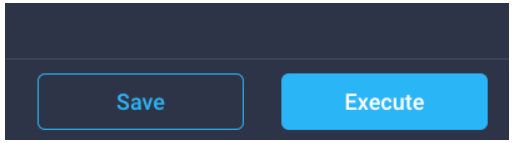
- conditions
- temperature



Add two **Workflow Outputs** named ‘conditions’ and ‘temperature’ then map them to appropriate ‘Parameters’ task outputs

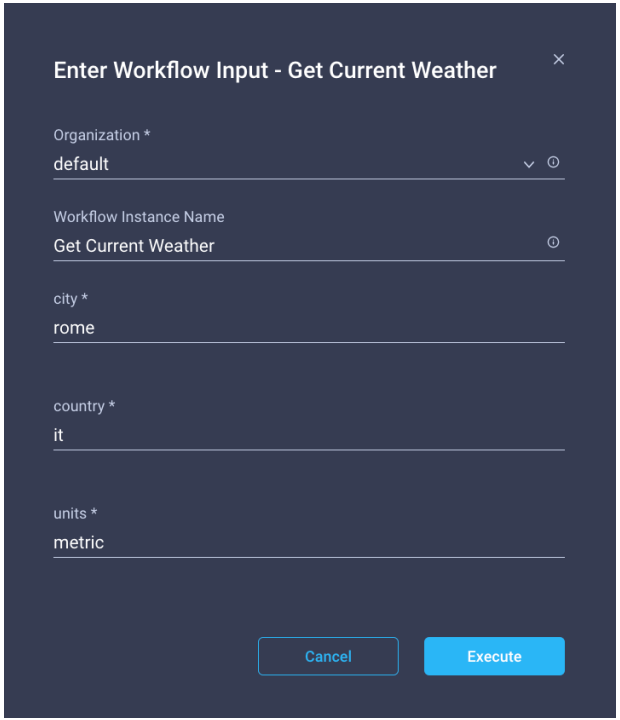
Validate and Execute a Workflow

Validate and Execute the Workflow



Save the workflow. The system will perform validation checks.

Once done, click on **Execute** to run the workflow

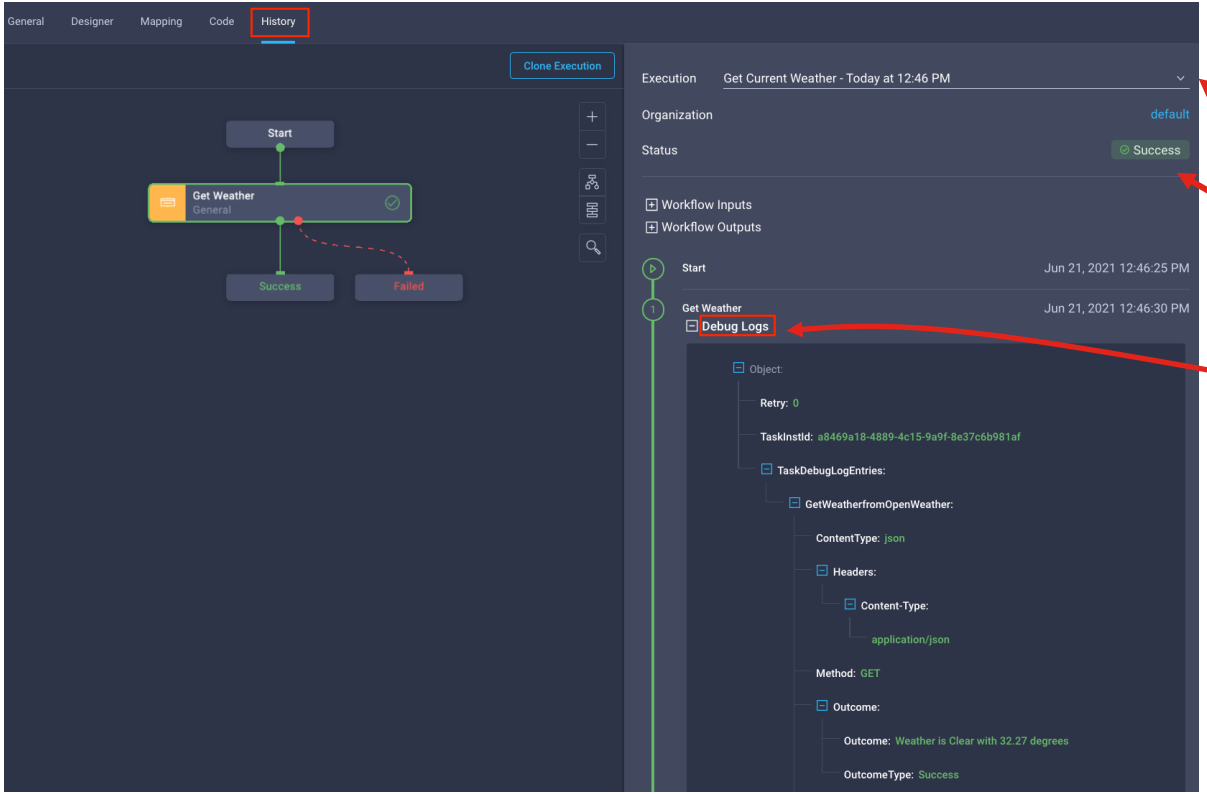


Select an organization, the required inputs of the workflow should now appear.

Fill in the details with your favourite city, country (ISO3166) as well as the units (metric or imperial).

Click on **Execute**

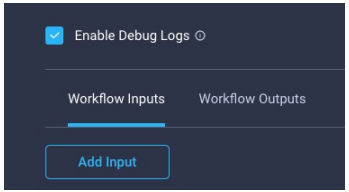
Validate and Execute the Workflow



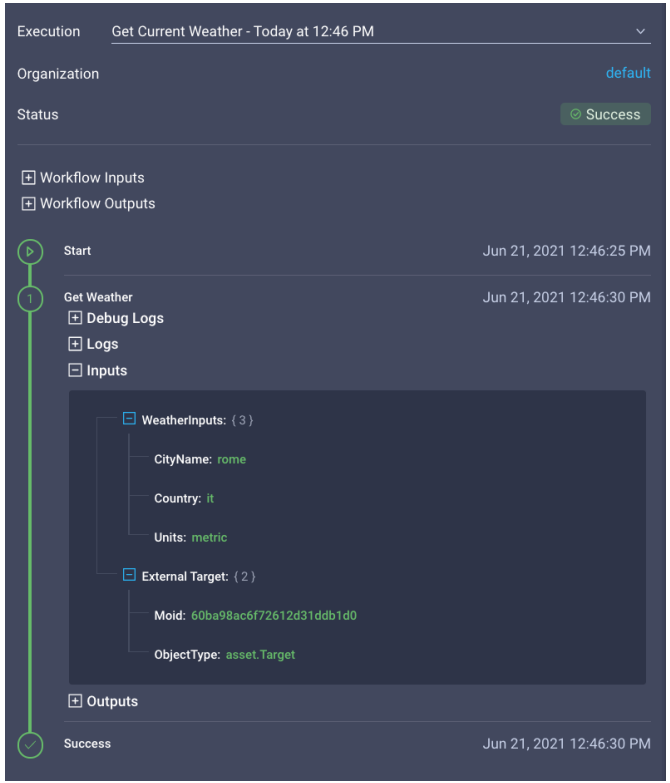
Right after the execution you'll move to the **History** tab of the workflow. Here you can check details of all workflow executions

Here is the status of the selected execution

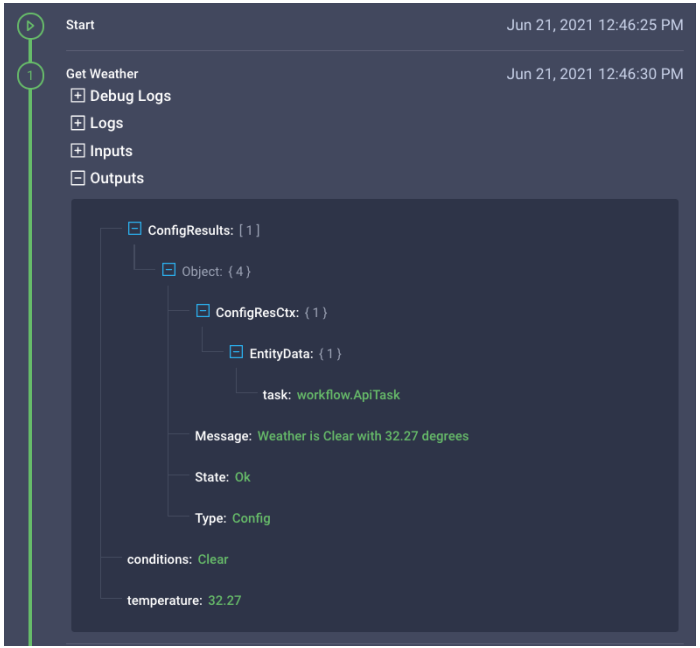
Shows **Debug Logs** (if enabled in the workflow under the **General** tab) for deeper inspection or troubleshooting. Example: TargetURL, Headers, ContentType, Outcomes, etc.



Validate and Execute the Workflow



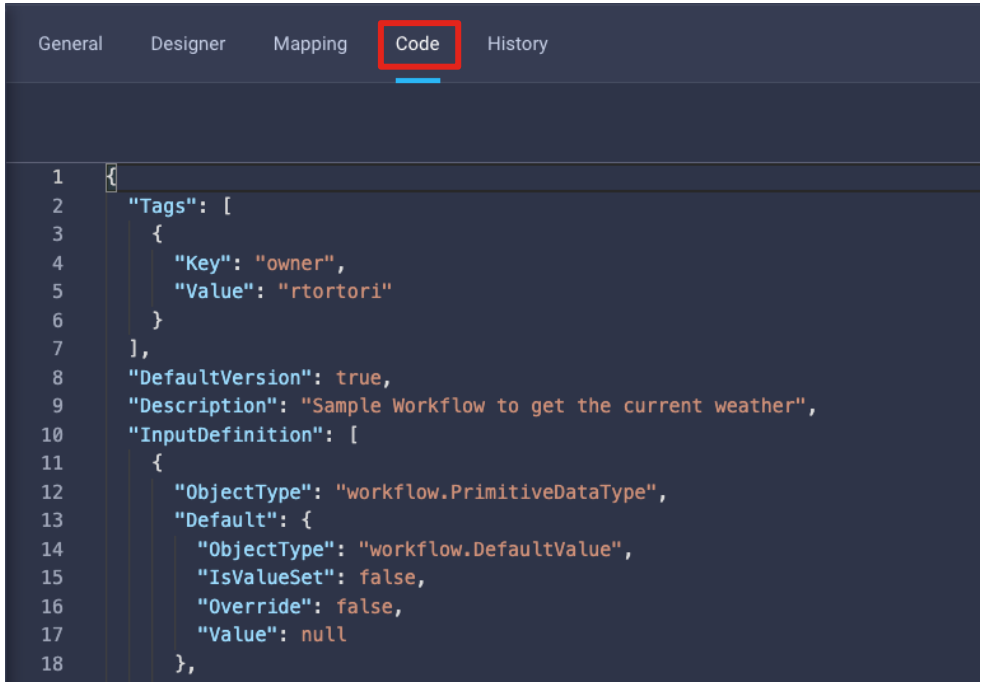
The **Inputs** fields will display the workflow inputs value used in this specific execution



The **Outputs** fields will display the workflow outputs.

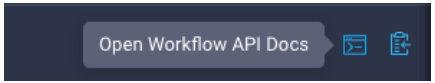
You can concatenate outputs to other task inputs

Validate and Execute the Workflow



The **Code** tab displays the generated code for this workflow.

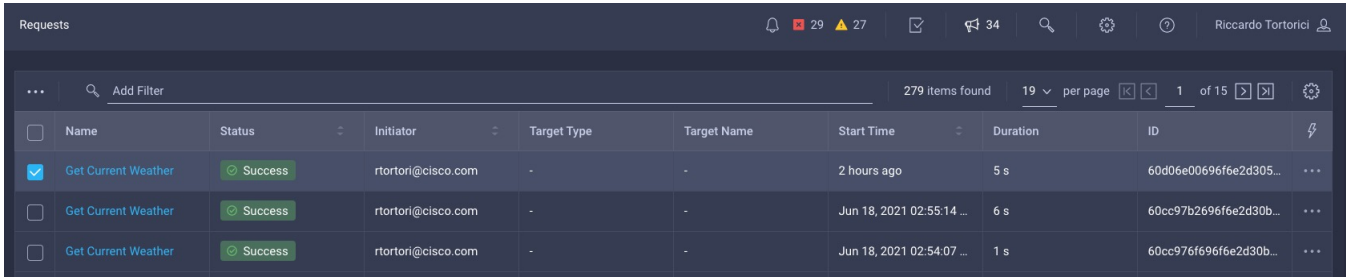
There will eventually be the ability to live edit the code.
For now it's there to use as reference for things you can't see in the UI, such as unique names of the task blocks as they're put on the canvas

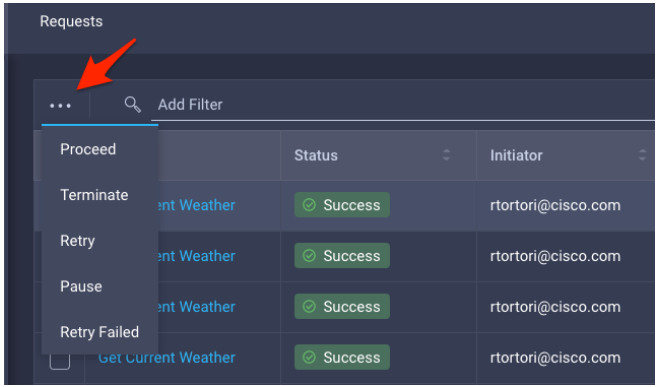


Clicking on the 'terminal' icon in the upper right corner, will open the REST API docs for the workflows directly, where you can optionally copy the current code and make modifications

Validate and Execute the Workflow

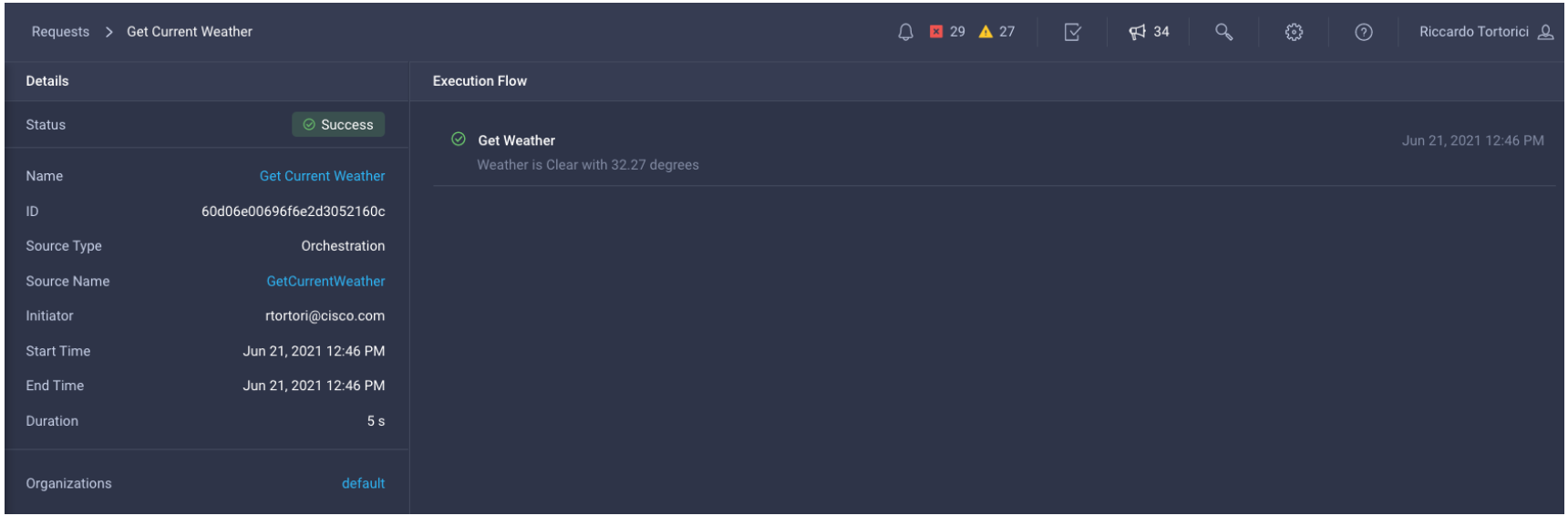
View all will display all Requests





You can select one or multiple **Requests** and execute actions

Validate and Execute the Workflow



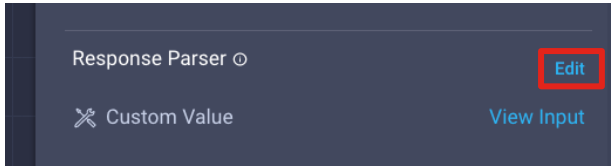
Clicking on one **Request**, you will get its **Execution Flow**, which will display outcomes for each tasks and external workflows invoked in this execution. In this case, we had a single task in our workflow but you can nest multiple workflows and use them as atomic tasks.

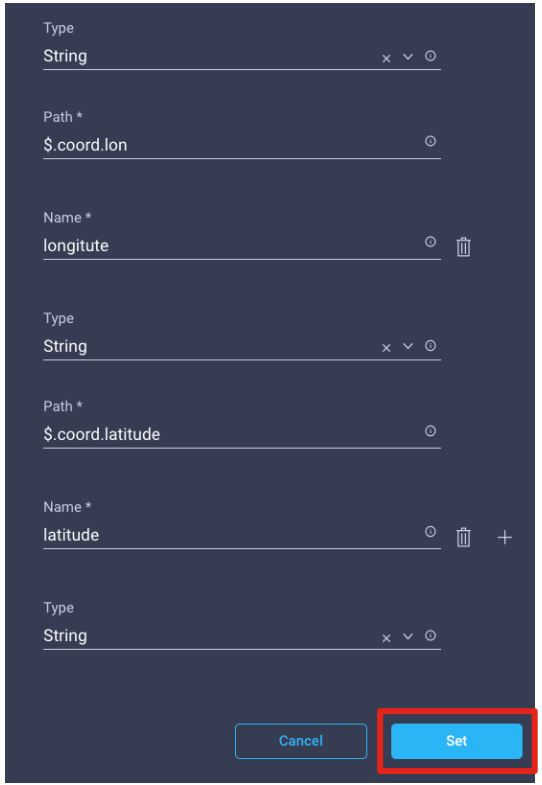
Use Task Outputs as Inputs for other Tasks

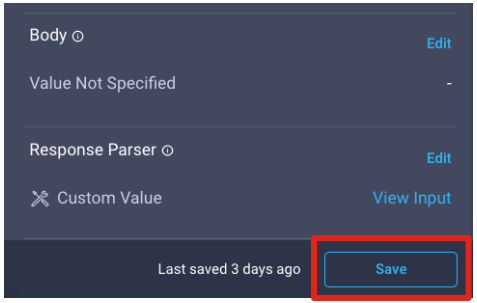
Modify Custom Tasks – Response Parser

Scope: extract from API response the ‘longitude’ and ‘latitude’

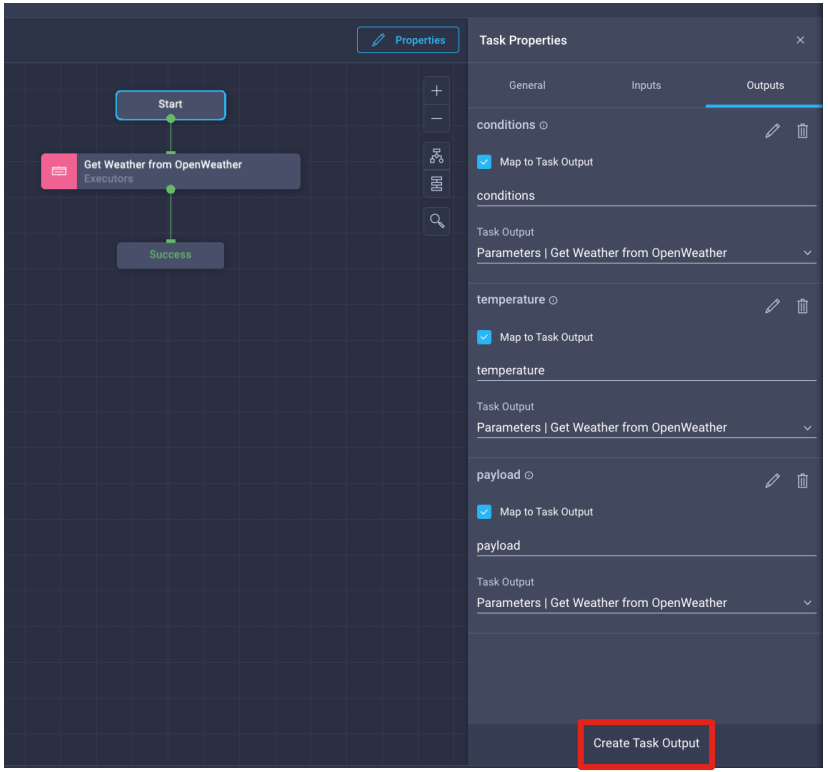
In the custom task, we edit the **Response Parser** to extract two additional parameters

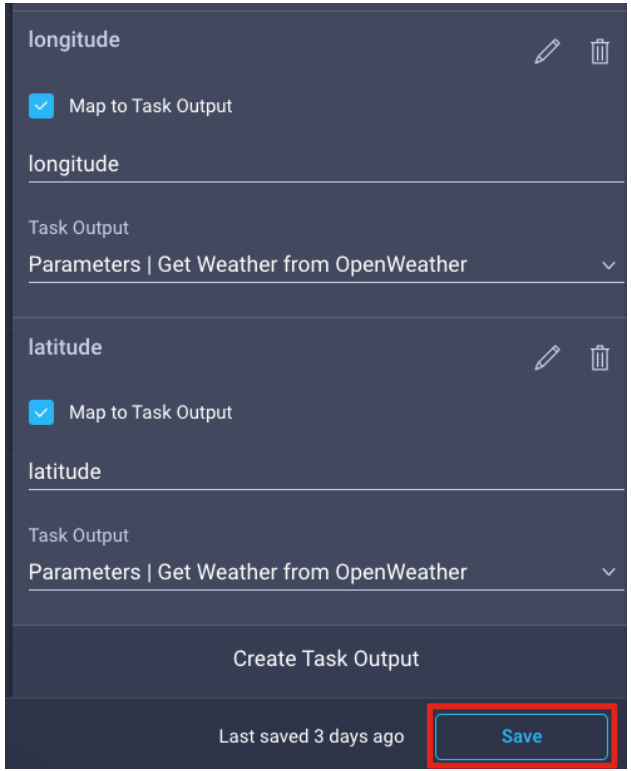




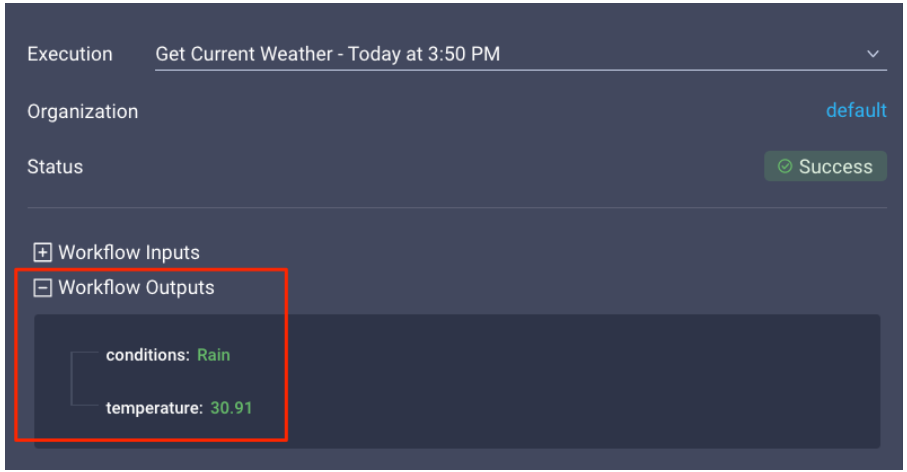


Modify Custom Tasks – Custom Task Outputs



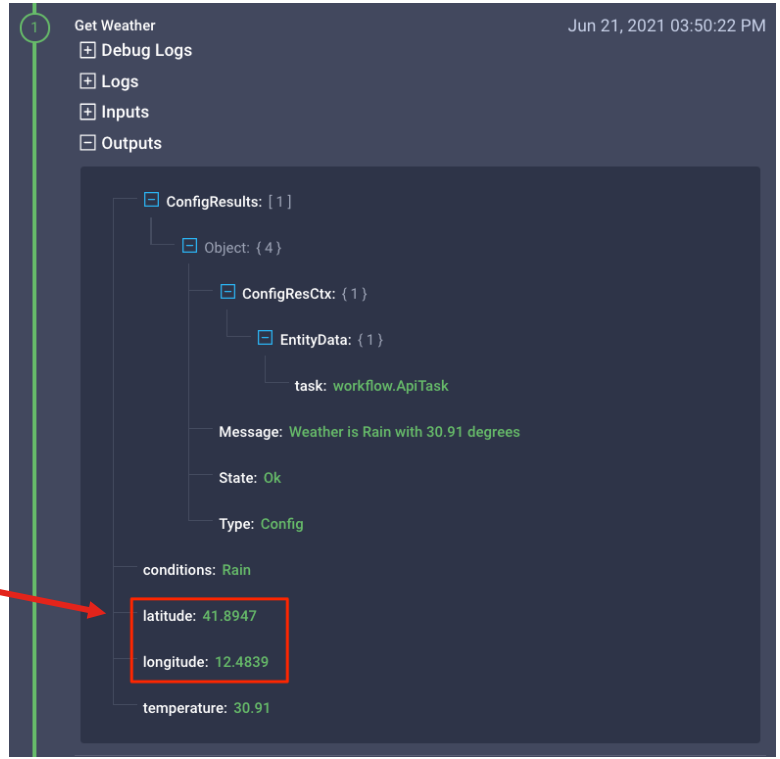


Check New Task Outputs

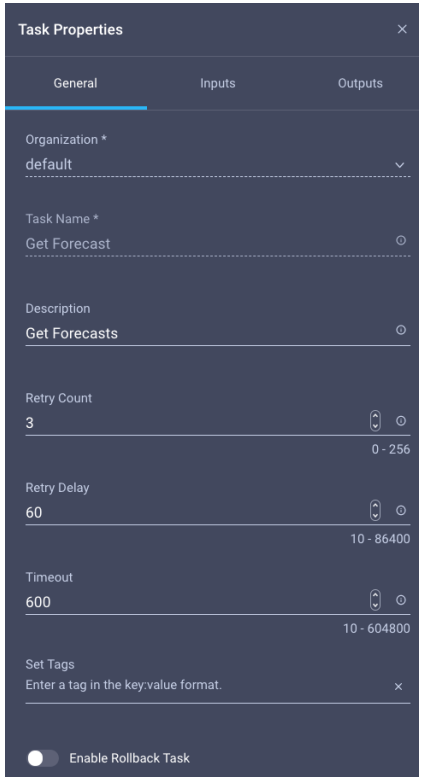


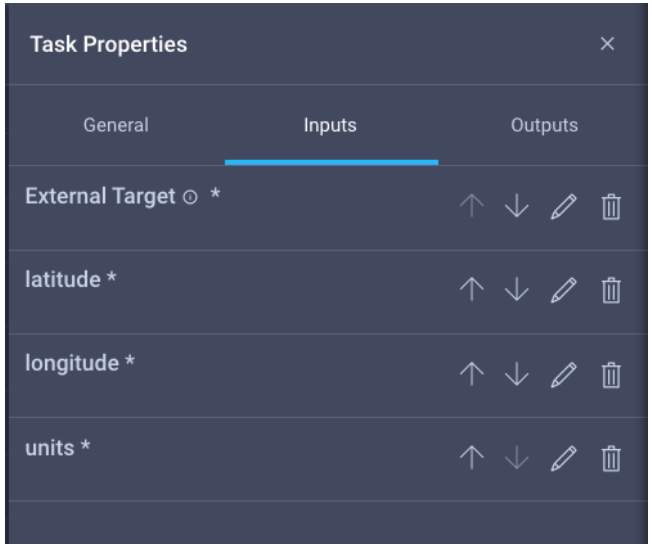
A new execution highlights the new task outputs.

Note: **Workflow Outputs** still shows the original outputs

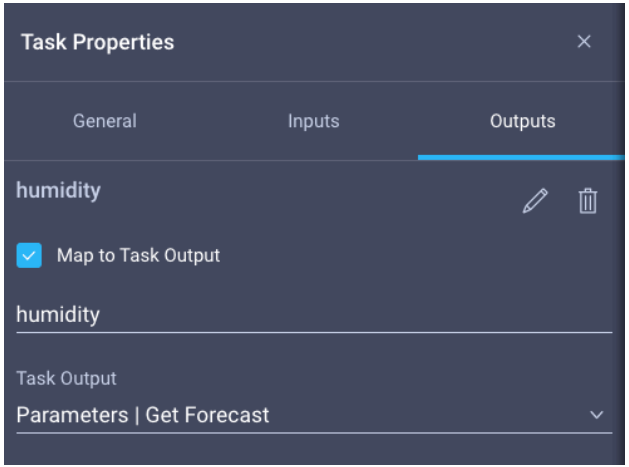


Create a New Task to Get Detailed Info – 1/3



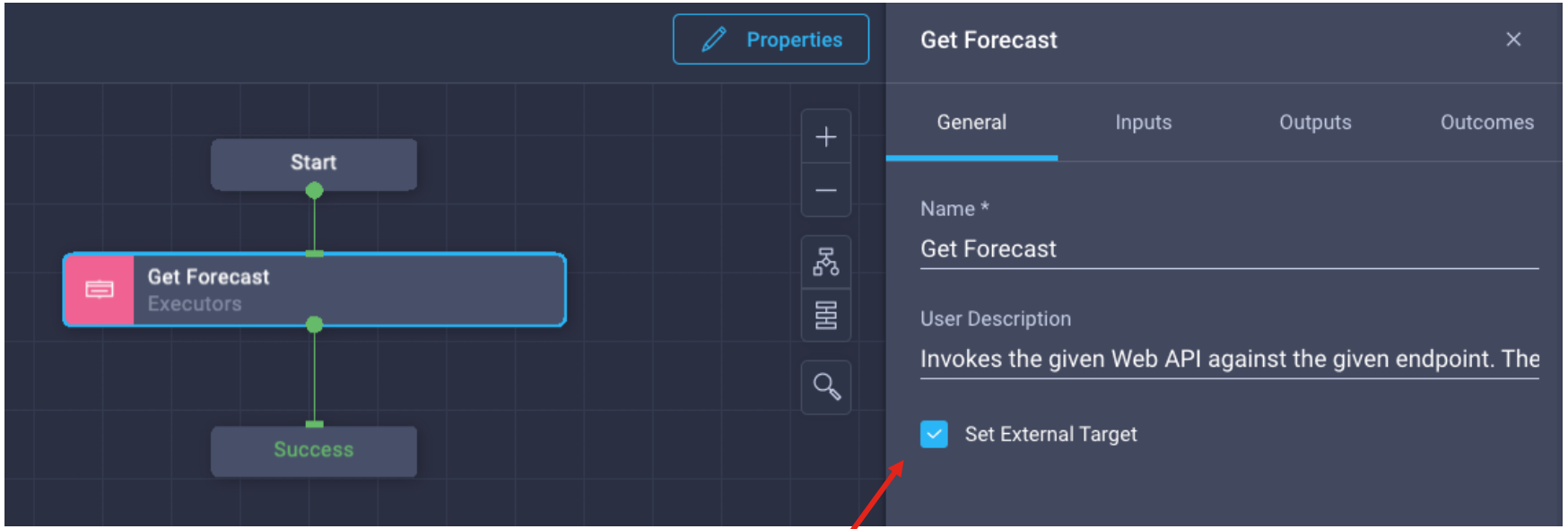


Name: ‘Get Forecasts’
All inputs as Strings.



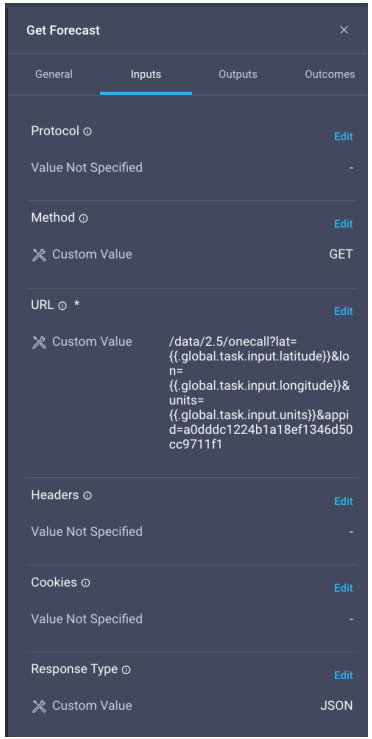
Create one output named ‘humidity’.
This will be mapped to the humidity response parser

Create a New Task to Get Detailed Info – 2/3



Executor properties.
Name: 'Get Forecast'
Set External Target checked (calls will run against the same target)

Create a New Task to Get Detailed Info – 2/3

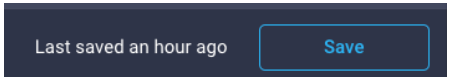


This call requires latitude and longitude. They will be mapped to the latitude and longitude outputs of the 'Get Weather' task within the main workflow.

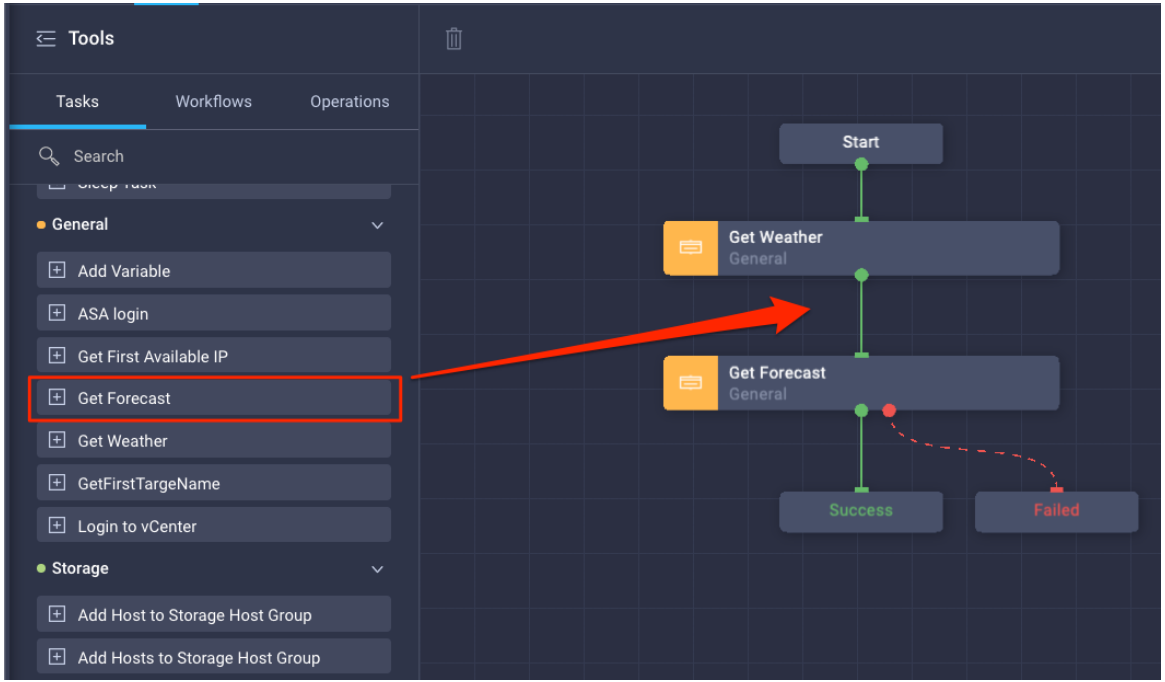
/data/2.5/onecall?lat={{.global.task.input.latitude}}&lon={{.global.task.input.longitude}}
&units={{.global.task.input.units}}&appid=a0dddc1224b1a18ef1346d50cc9711f1

Response parser is extracting the current humidity and store in the 'humidity' parameter as a string

Save once done

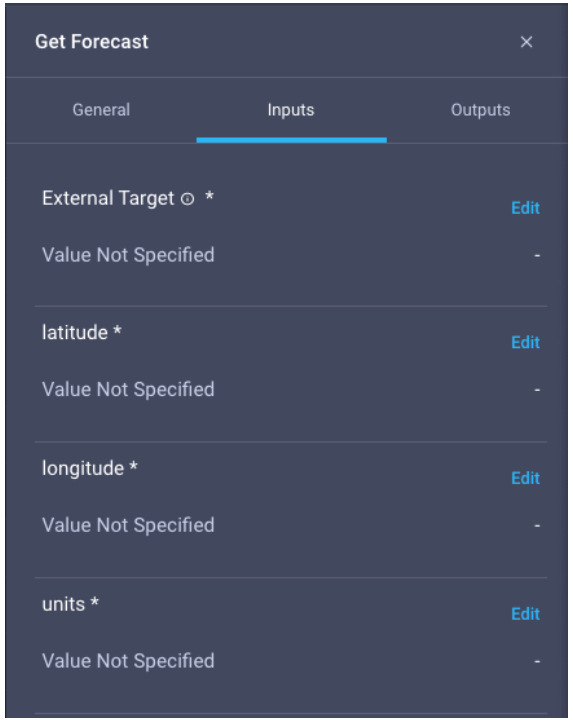


Add New Tasks to an Existing Workflow



In the main workflow, drag the new ‘Get Forecast’ task in the designer right below the ‘Get Weather’ task

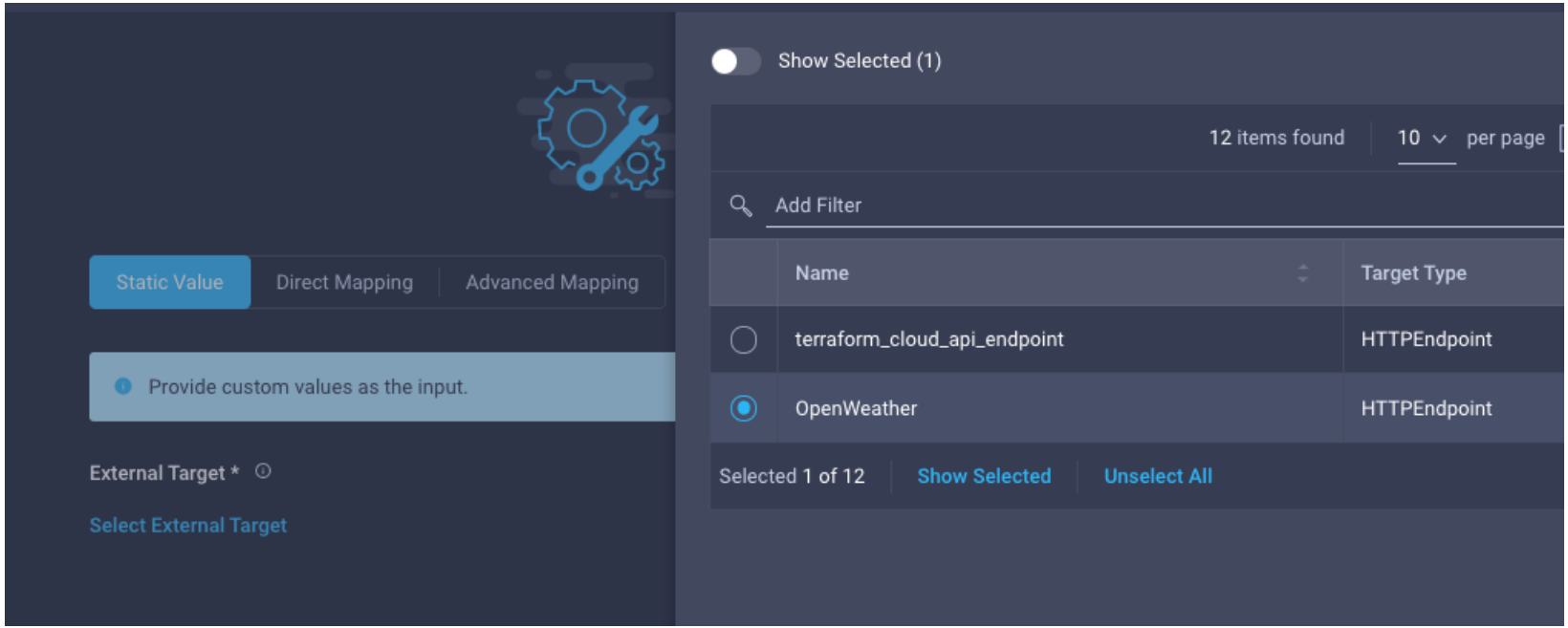
Add New Tasks to an Existing Workflow



The ‘Get Forecast’ tasks wants four inputs:

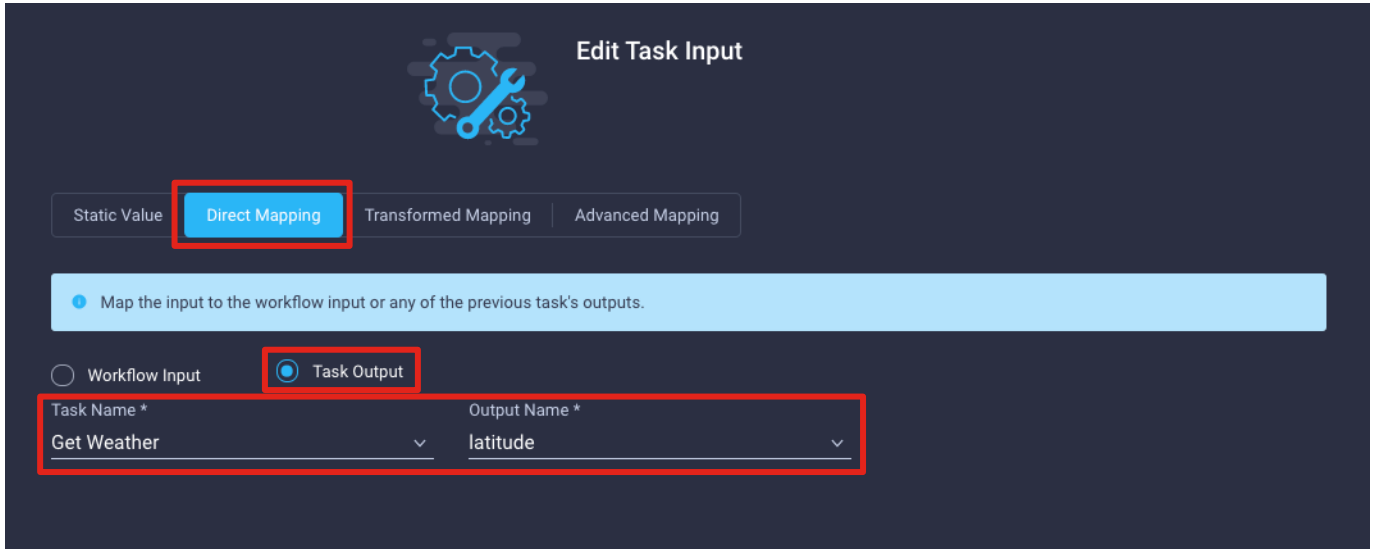
- External Target – it will be the same target we claimed for the weather
- Latitude – we get this value from the previous task output
- Longitude – we get this value from the previous task output
- Units – we get this value from the workflow input

Add New Tasks to an Existing Workflow



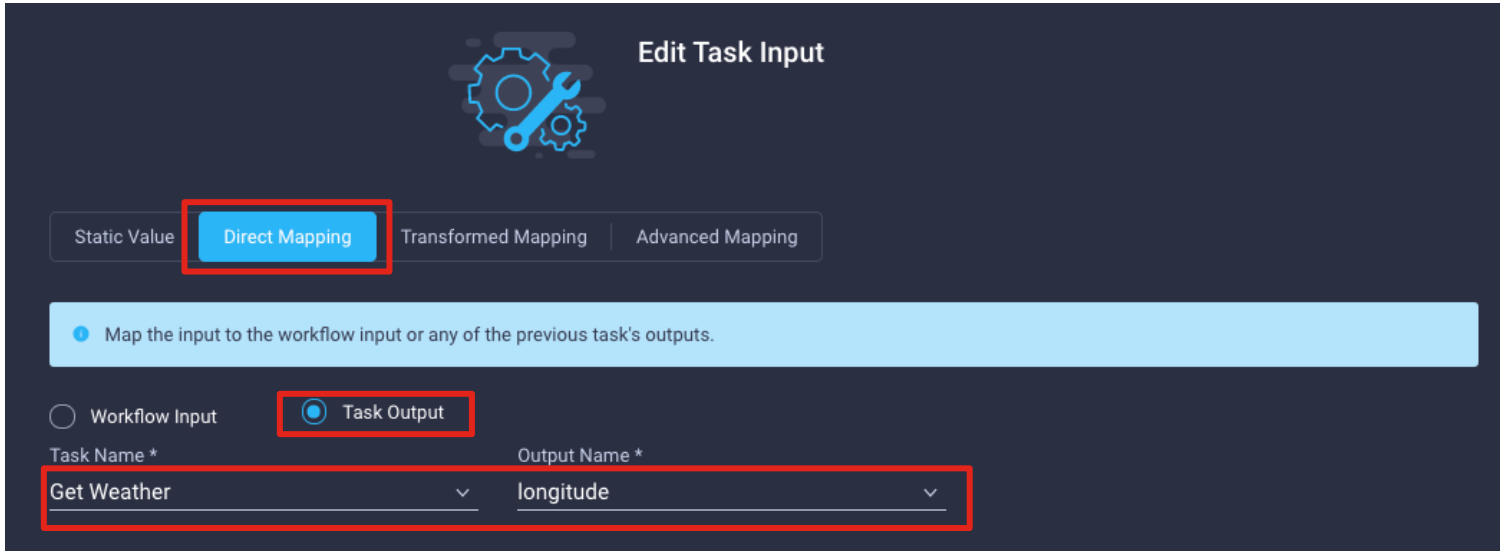
External Target will be a static mapping

Add New Tasks to an Existing Workflow



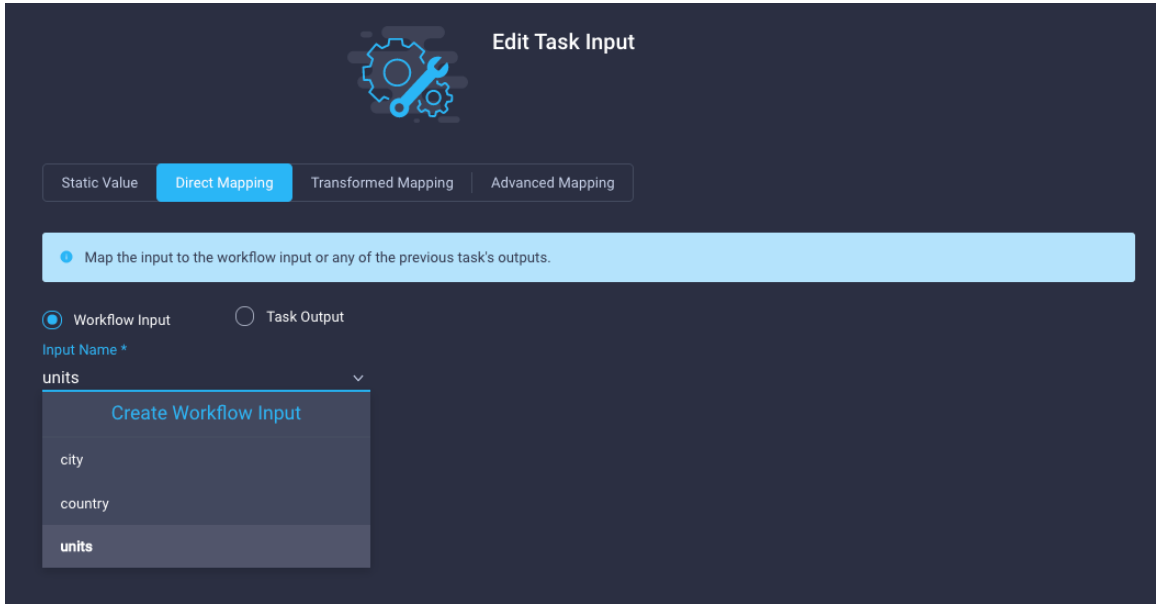
For the latitude, we want the value to be the output of the previous task (‘Get Weather’)

Add New Tasks to an Existing Workflow



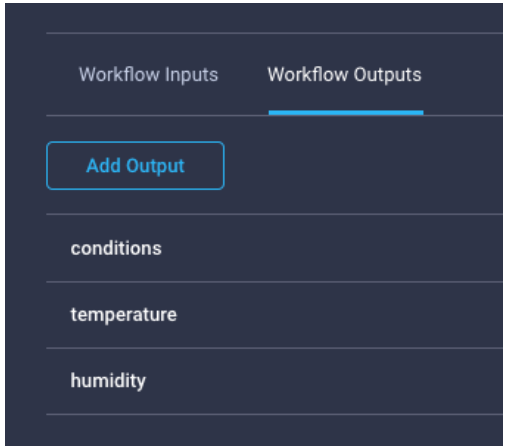
For the longitude, we want the value to be the output of the previous task (‘Get Weather’)

Add New Tasks to an Existing Workflow



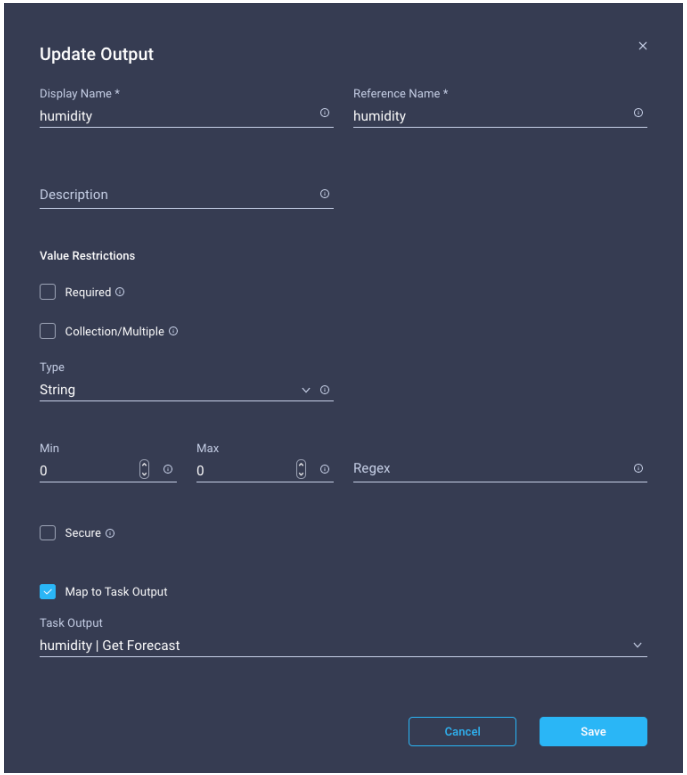
For the units, we take this value from the workflow input

Add Workflow Outputs



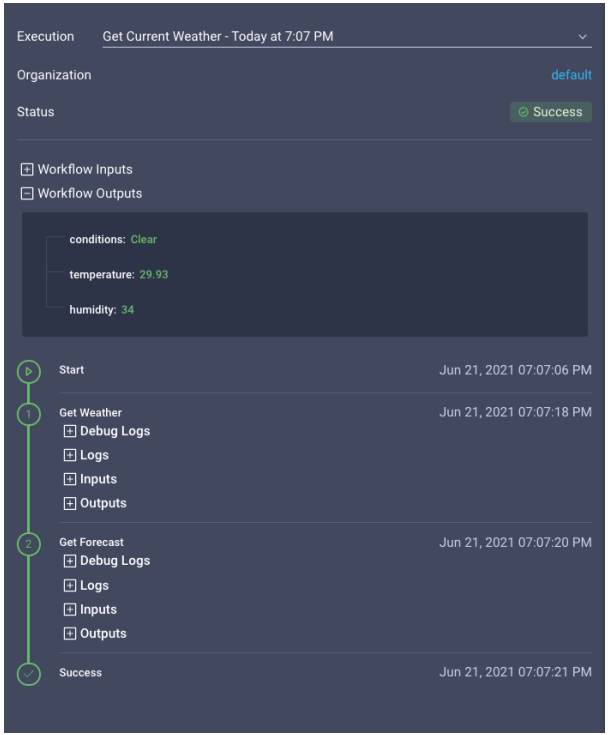
We are going to add an additional workflow output called ‘humidity’.

We will **Map to Task Output** ‘Get Forecast’ output called ‘humidity’



Save the workflow, then **Execute**

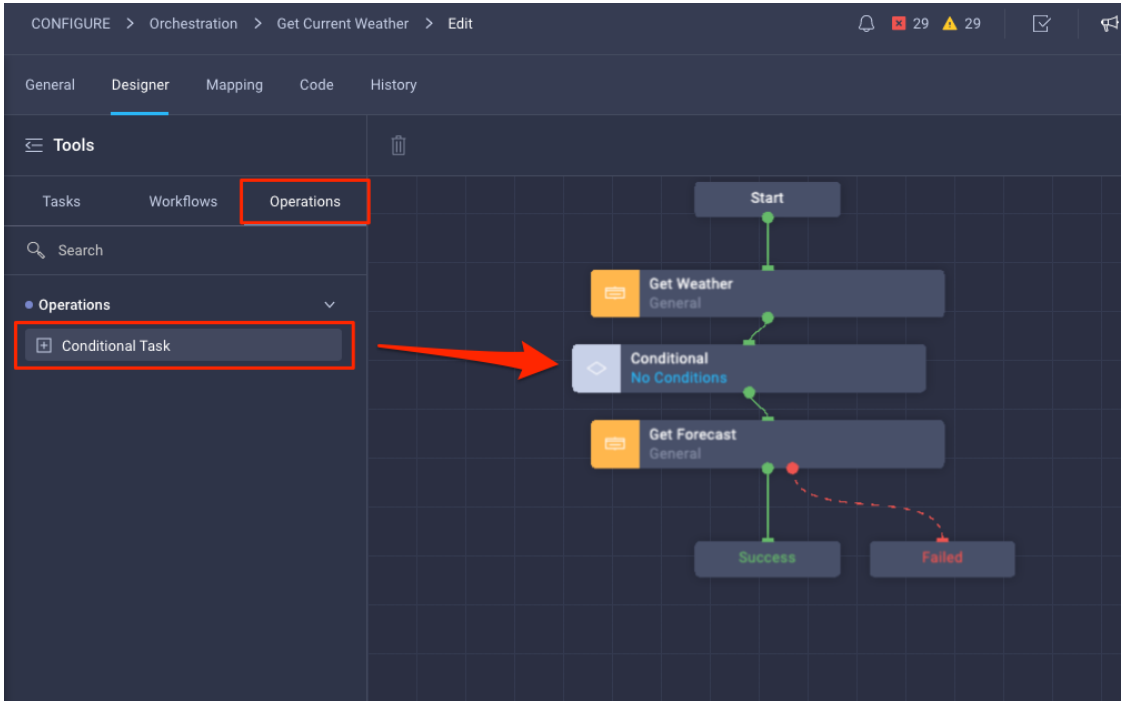
Check Workflow Execution Details



Workflow Outputs now shows conditions, temperature as well as the humidity we got with the second task

Conditional Task

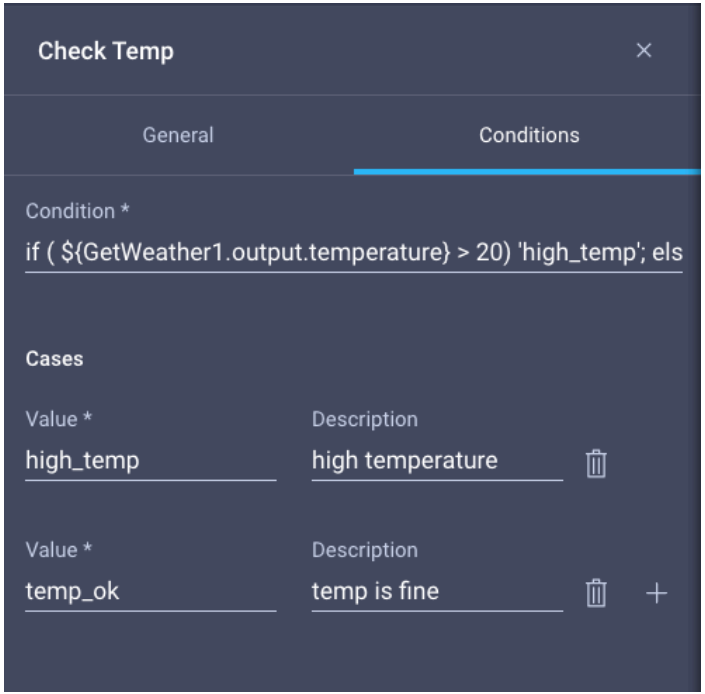
Using the Conditional Task



Scenario:
Get humidity value only if the temperature crosses a threshold

Drag the **Conditional Task** in the designer, between the 'Get Weather' and the 'Get Forecast' tasks

Using the Conditional Task



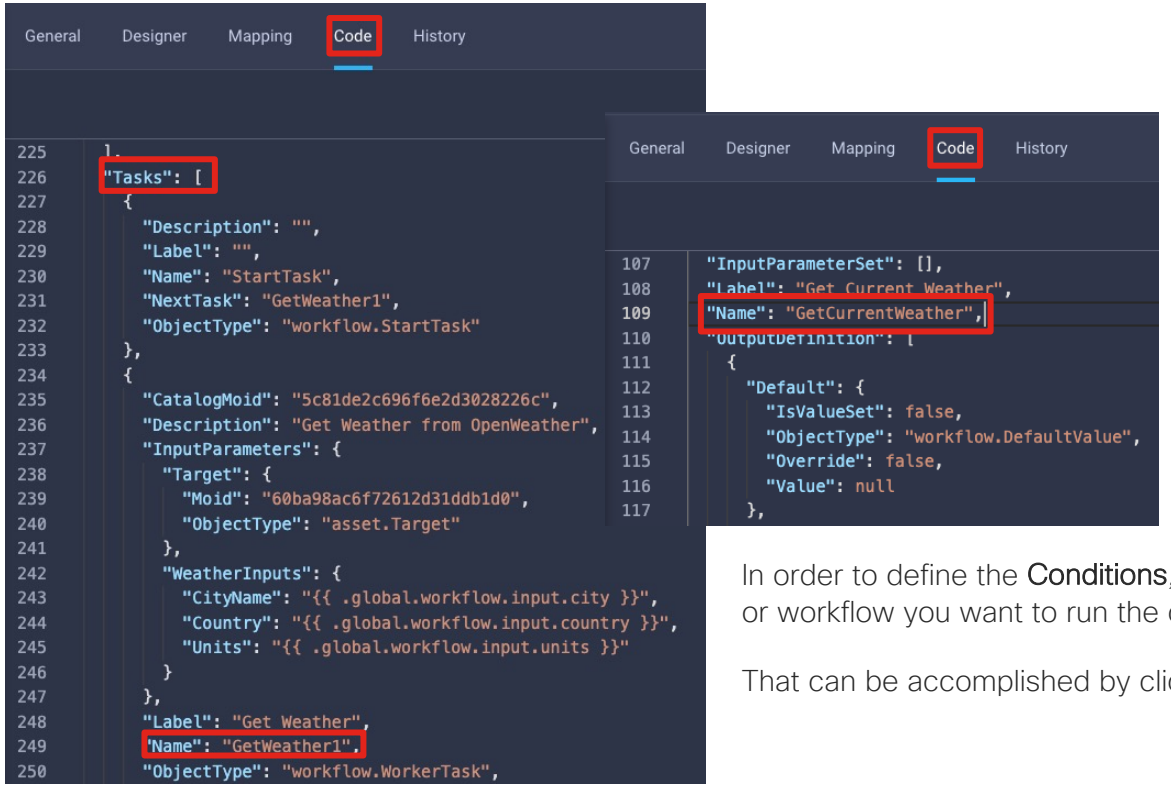
The conditional task requires two main inputs:

- The **Condition** we want to check
- The **Cases**, that is the possible outcomes

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

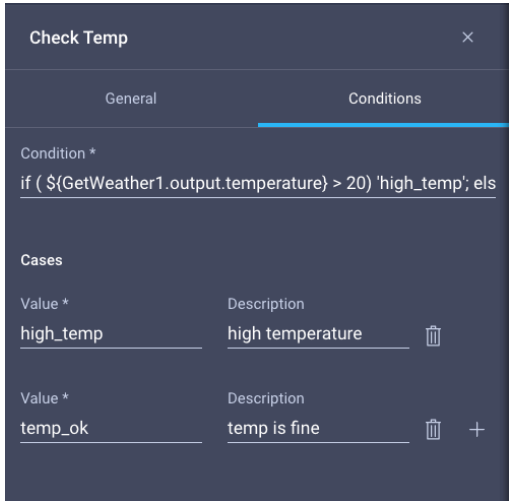
Using the Conditional Task



In order to define the **Conditions**, you will need the unique **Name** of the task or workflow you want to run the condition on.

That can be accomplished by clicking on the **Code** tab in the designer

Using the Conditional Task



If you want to check a **Condition** against a workflow input, the object syntax is:

```
${workflow.input.<workflow input ReferenceName>}
```

If you want to check a **Condition** against a task output, the object syntax is:

```
${<Task Name>.output.<Task Output Name>}
```

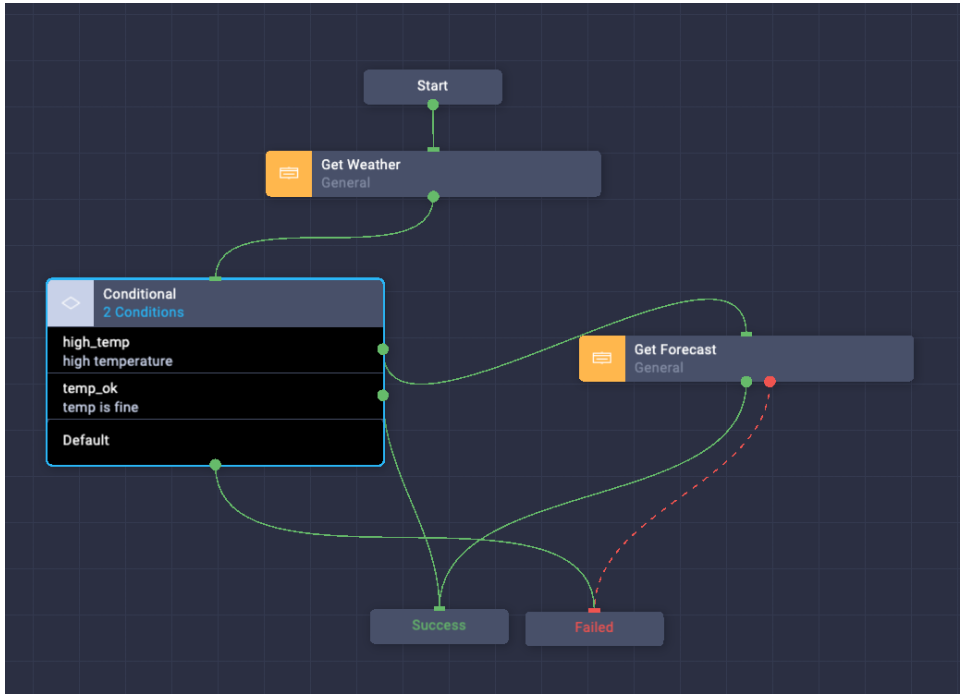
In this example, we want to check that the 'temperature' output of the task 'Get Weather' is *greater* (>) than 20 (degrees). If that is the case, we set a value of 'high_temp', otherwise we set a value of 'temp_ok'.

To set this condition, we use the following syntax:

```
if ( ${GetWeather1.output.temperature} > 20) 'high_temp'; else 'temp_ok'
```

In the **Cases**, we create two entries, one for each value we've set.

Using the Conditional Task

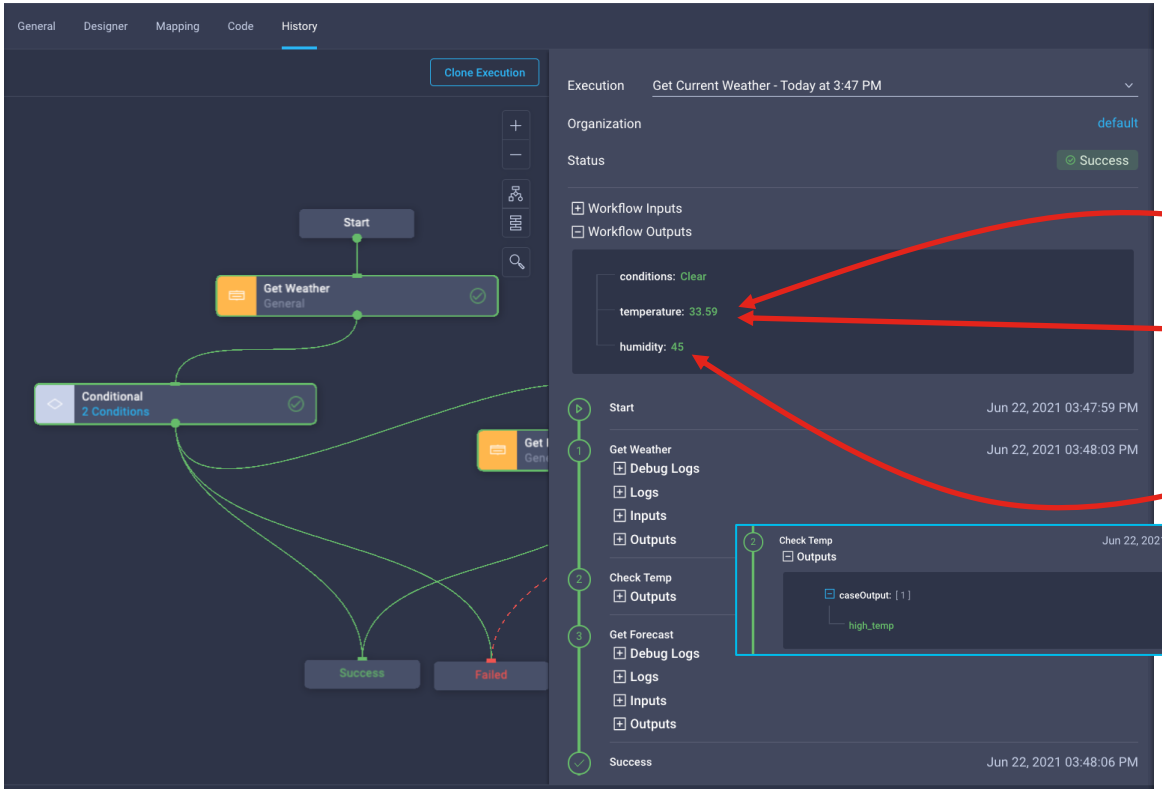


In the designer, the **Cases** we just configured will appear as blocks you can connect to other tasks based on the outcome you want or directly to the **Success** or **Failed** blocks.

In this example, as soon as we start the workflow, we will get the current weather conditions and temperature of the selected location. The conditional task will evaluate whether the temperature is high or ok and if the temperature is high, it will go to the 'Get Forecast' task that will return the relative humidity, otherwise it will exit.

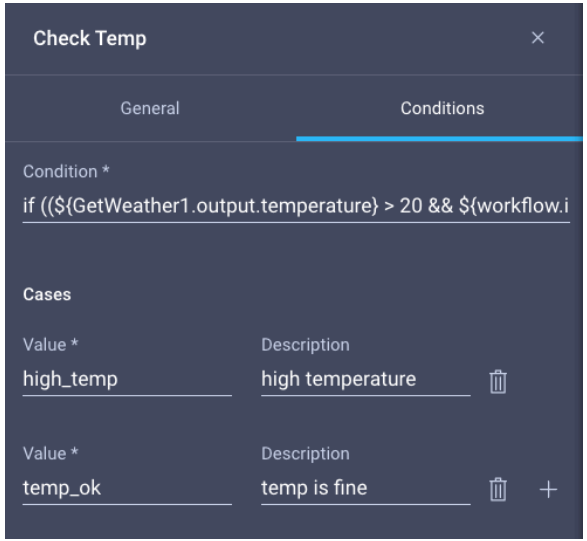
Save the workflow and **Execute**

Using the Conditional Task



Since the 'temperature' returned from the 'Get Weather' task is greater than 20 (33.59 degrees), the workflow executes the 'Get Forecast' task and gets the relative 'humidity'

Compound Conditional Operations



The current implementation only works for the ‘metric’ unit as the Condition will always look at values greater than 20 degrees even if the ‘temperature’ is returned as ‘imperial’. We want to implement a logic so we can also set a threshold for ‘imperial’ units.

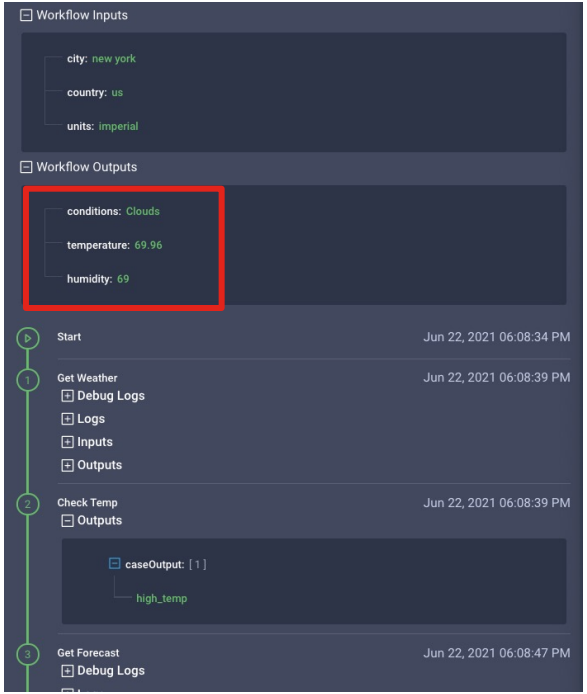
The following expression will cover both cases, using the OR (||) operator:

```
if ((${GetWeather1.output.temperature} > 20 &&
${workflow.input.units} === 'metric') ||
(${GetWeather1.output.temperature} > 68 &&
${workflow.input.units} === 'imperial')) 'high_temp'; else
'temp_ok'
```

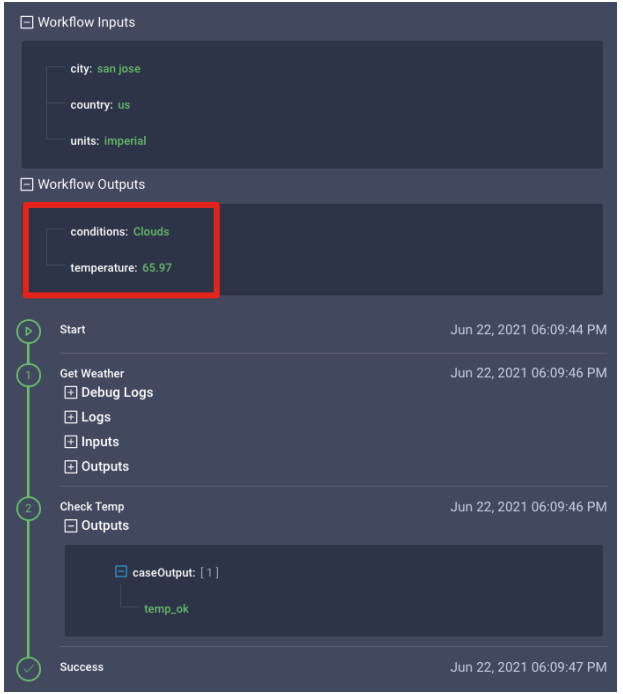
In this example we set the threshold for ‘imperial’ as 68 degrees and 20 degrees for ‘metric’.

Additional information and examples on conditional expression can be found here:
https://www.intersight.com/help/resources/Workflow_Designer#operations

Compound Conditional Operations



New York, US, Imperial
Case output: high_temp – 'Get Forecast' triggered



San Jose, US, Imperial
Case output: temp_ok – 'Get Forecast' skipped

Parallel Loops

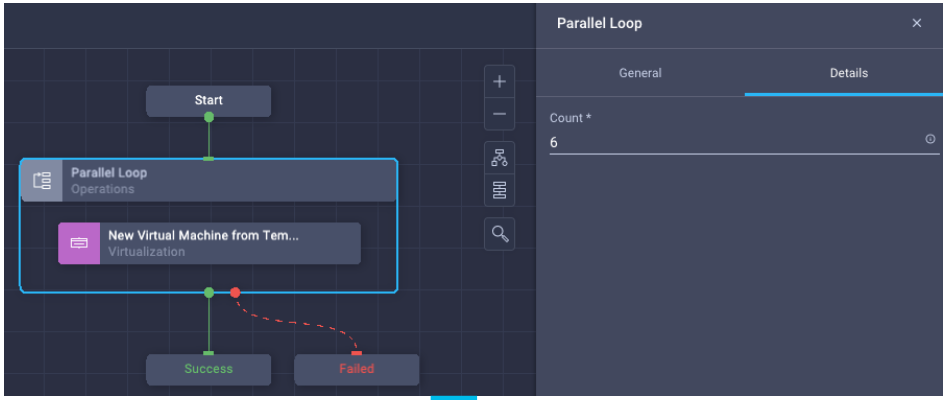
Parallel Loops

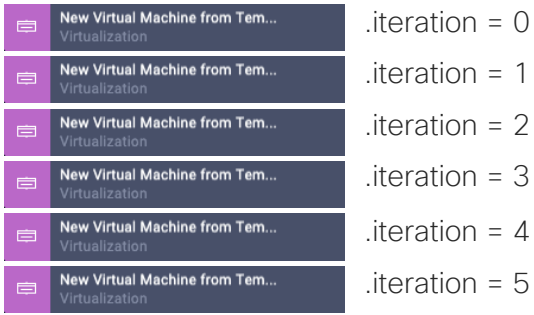
Parallel Loops allow a user to execute a specific task multiple times in parallel.
A sample use case would be to provision multiple objects at once (i.e. a given number of Virtual Machines)

There are two parameters to consider:

1. **Count**. This parameter specifies the number of executions you want to run in parallel for the task you drag into the loop
2. **Iteration.** This parameter has significance only within the individual task execution and represent the current number of the task execution

Example:
You specify a **Count** value of 10 and drag the **'New Virtual Machine from Template or Clone from Virtual Machine'** task into the loop.
The loop will execute the task 10 times in parallel, meaning that 10 Virtual Machine will be created. Inside each task execution, the **.iteration** parameter will represent the task execution number



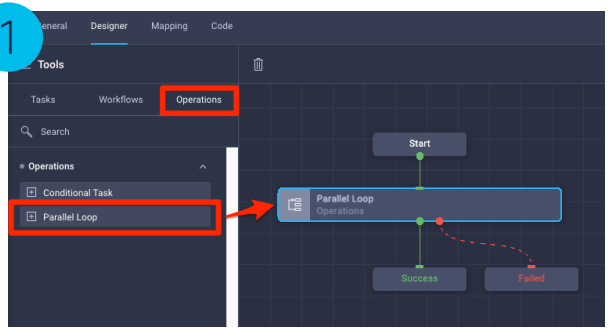


Example:
If you set the **Virtual Machine** name as
`myvm_{{.iteration}}`

ICO will create 6 VM named: myvm_0, myvm_1, myvm2, ...

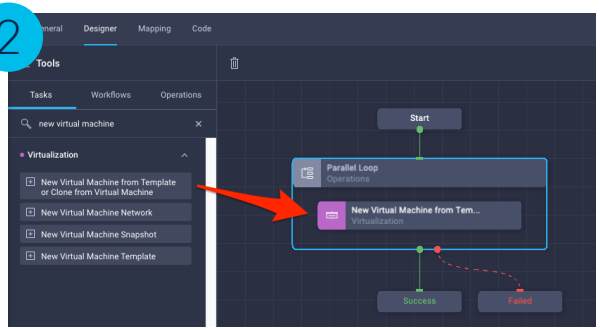
Parallel Loops Example – Create Multiple VMs 1/2

1



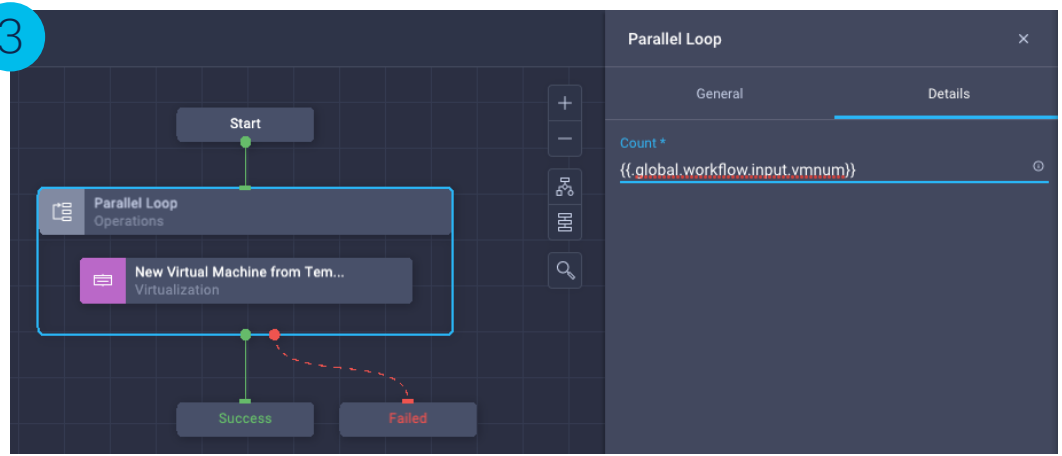
Drag the **Parallel Loop** task into the designer

2



Drag the task you want to execute multiple times in parallel **inside** the **Parallel Loop** task

3



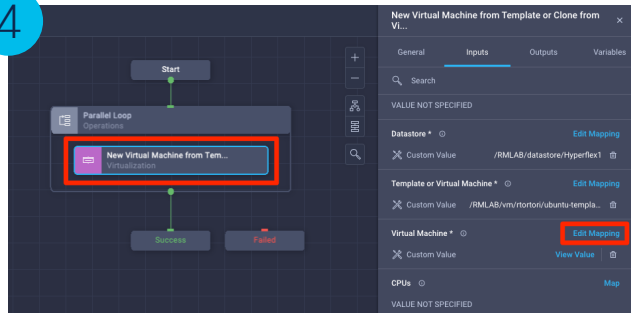
Click on the **Parallel Loop, Details** tab and specify a count value.
This needs to be an integer, you can set it statically (i.e: 6) or use the templating syntax to set it dynamically or from a workflow input.

In this case, we want the user to specify the number of execution from a workflow input so we use the `{{.global.workflow.input.vmnum}}` syntax.

Note: this requires a WF input with reference name **vmname**, type **Integer**

Parallel Loops Example – Create Multiple VMs 2/2

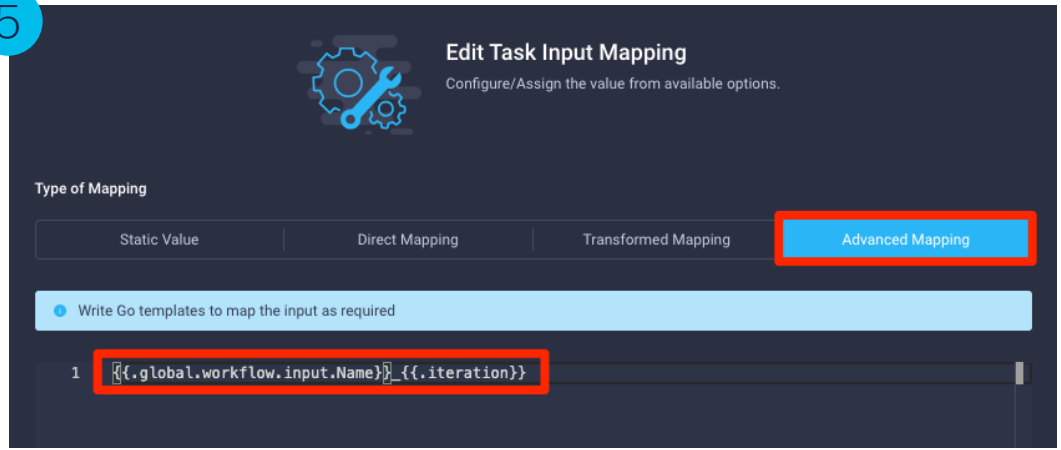
4



In this example, we want to give a distinguished name for each VM, otherwise they will all have the same name.
Click on the **New Virtual Machine from Template or Clone from Virtual Machine** task inside the **Parallel Loop** and map the **Virtual Machine** input

*Note: This example just covers the **Parallel Loop** use case, it won't cover how to set all required inputs to create a Virtual Machine.*

5



Use **Advanced Mapping** to set the Virtual Machine name:
`{{.global.workflow.input.Name}}_{{.iteration}}`

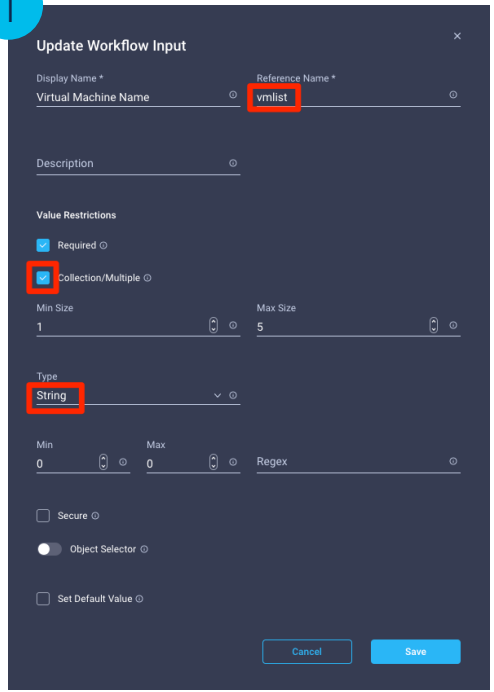
Value from a Workflow Input with reference name **Name**
Example: myvm

Iteration number
Example: 1

Sample Result: myvm_1

Parallel Loops Example – Create Multiple VMs from List

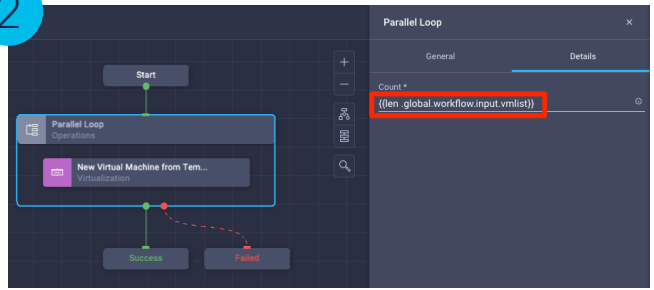
1



In this case, we have a WF Input with reference name **vmlist** defined as a collection of **String** (basically, an array).
The use case is to create *n* virtual machines, where *n* is the number of virtual machine names present in the collection.

As an example, we specified 1 and 5 as minimum and maximum number of items respectively.
This also means that we can only create max 5 VMs per each workflow execution

2



The **Count** value in the **Parallel Loop** can be set as follows:

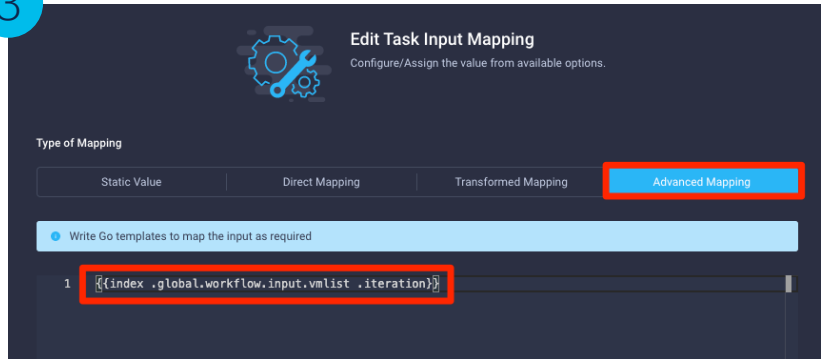
```
{{len .global.workflow.input.vmlist}}
```

This means ICO will execute *n* instances of the inner task where *n* is the length of the list provided by the user

Note: the `len` function measure the length of a string (characters) or array (number of items).
For additional info please refer to the following documentation:
https://www.intersight.com/help/saas/resources/Workflow_Designer#operations_-_parallel_loop_task

Parallel Loops Example – Create Multiple VMs from List

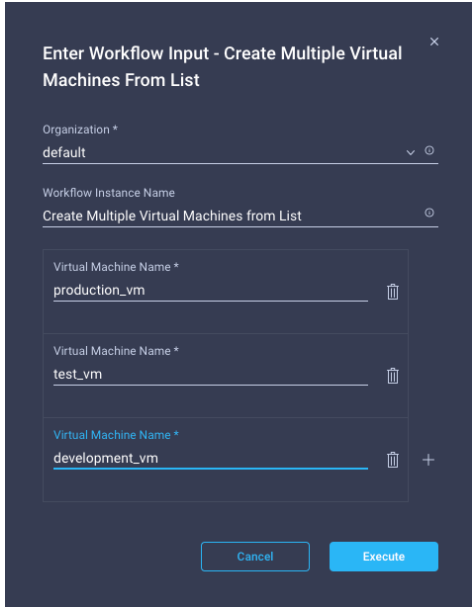
3



For the Virtual Machine name input mapping we are now following a different approach:
As the user will specify a list of **Virtual Machine** names, we want to assign those exact names to each VM.

`{{index .global.workflow.input.vmlist .iteration}}`

The `index` function will extract the value of the `.iteration` element of the array `.global.workflow.input.vmlist` (Workflow Input)



Executing the workflow, the user will set, as an example, 3 different **Virtual Machine** names which are creating our collection of elements.

In this case, **production_vm** will be element 0 (`.iteration = 0`), **test_vm** will be element 1 (`.iteration = 1`) and **development_vm** element 2 (`.iteration = 2`)

Example:
When the task with `.iteration = 2` will execute, the `index` function will extract from the array the value of element 2, in this case **development_vm**

Workflow Variables

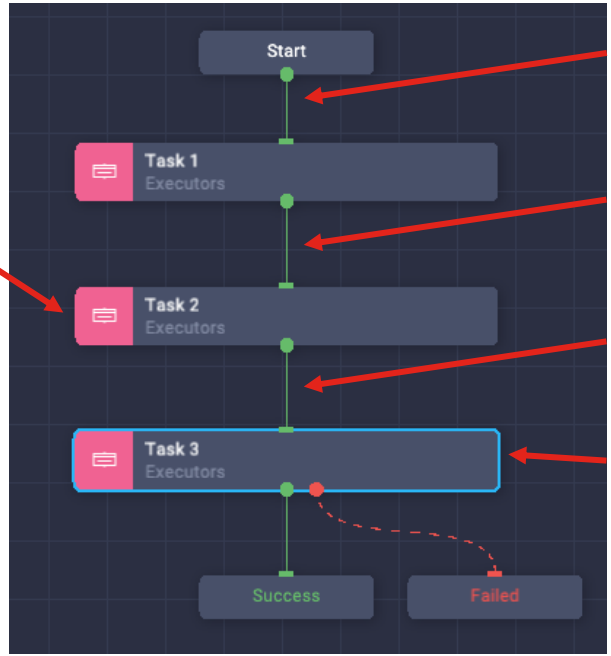
Workflow Variables

Workflow variables are similar to local variables within functions of a programming language.
Workflow variables are scoped within a workflow.
In a workflow, all tasks are bound to the scope of the workflow and can **read** or **update** the workflow variables that are defined for the workflow.

The update of a variable happens **after the task has been executed**, for instance **Task 2** executes and before it exits it updates the variable **mynumber**

Sample use cases:

- Simplify workflows using conditional branches. Multiple tasks can update (save a value) the same variable, this value can be then used by another task irrespective of the branch from which it was taken
- Transformation. Apply a transformation function and save the result in a variable without having to redo the transformation in every mapping. For instance, you turn a Virtual Machine name into all uppercase and this transformation will be available in all tasks requiring the Virtual Machine Name



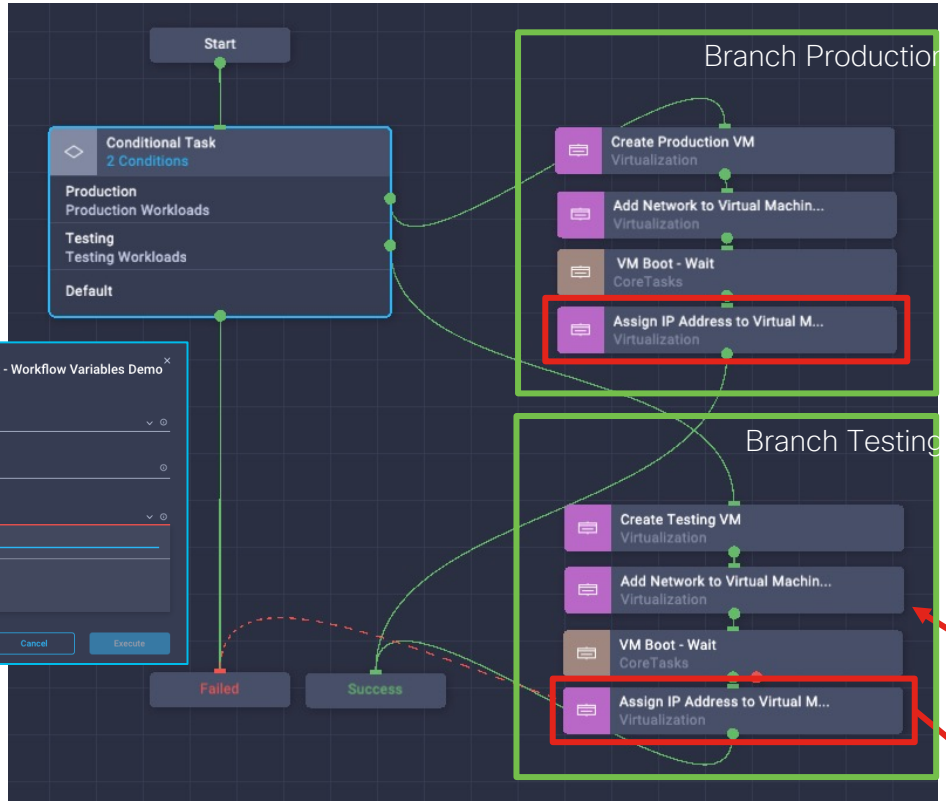
Variable mynumber:
Initial value: 10

Variable mynumber:
Task 1 updates value: 2

Variable mynumber:
Task 2 updates value: 3

Task 3 uses variable
Mynumber. Value: 30

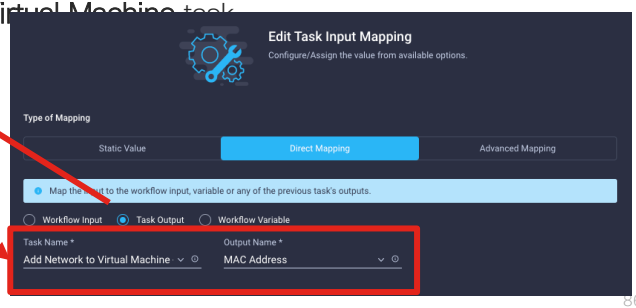
Example of Workflow without Variables



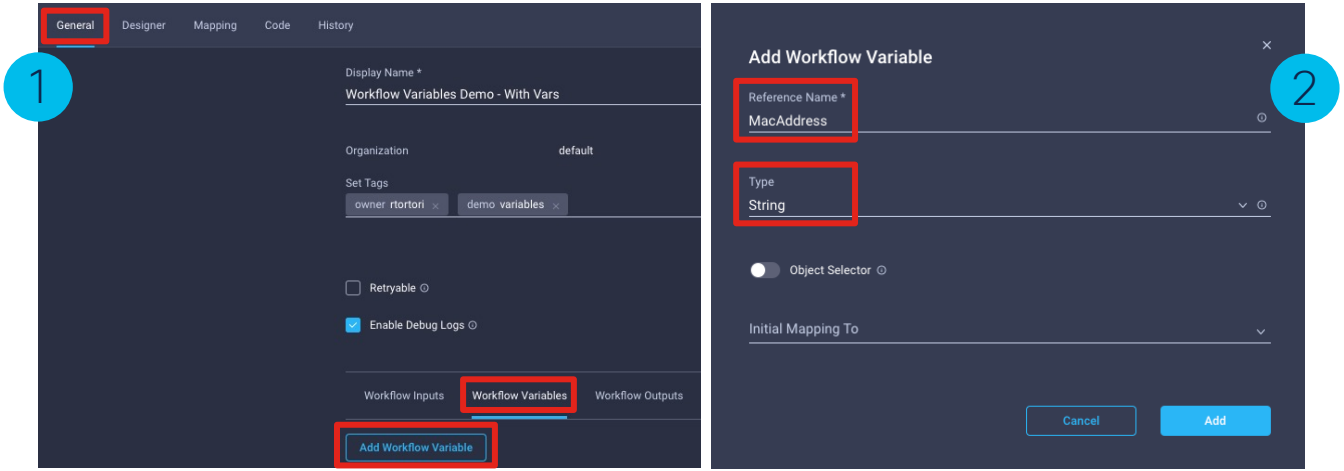
Consider this sample workflow.

There is a workflow input that allows the user to request a 'Production' VM or a 'Testing' VM.
Based on this condition we split in two branches to create the VM with the required specs, add the right network to it and eventually assign an IP address.

As the **Assign IP Address to Virtual Machine** task requires the MAC address of the VM, there is no way to know it before the VM has been created.
Since we can't map two task outputs to a single input, we create the same task in the two branches and map the **Virtual NIC – Mac Address** input to the **MAC Address** output of the corresponding **Add Network to Virtual Machine** task

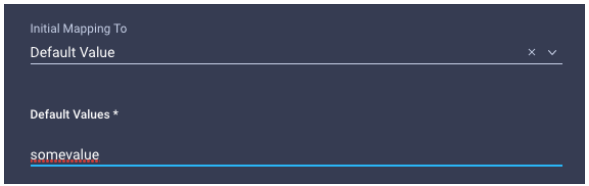


Using Variables 1/3



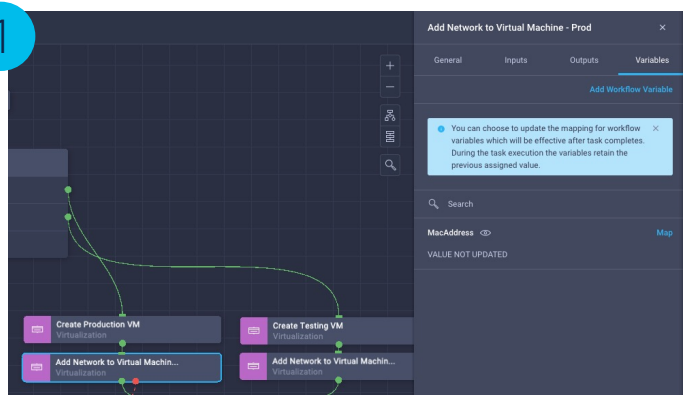
In the General tab, click on Workflow Variables and Add Workflow Variable. Assign a **Name** and **Type**.
At the time of writing (March 2022) we support the following types: *String, Integer, Float, Boolean, Json, Enum, MoReference and Target Data Type*

You can optionally assign a value using the **Initial Mapping To** option.
The variable will be created and can be mapped to any compatible input and updated by any task



Using Variables 2/3

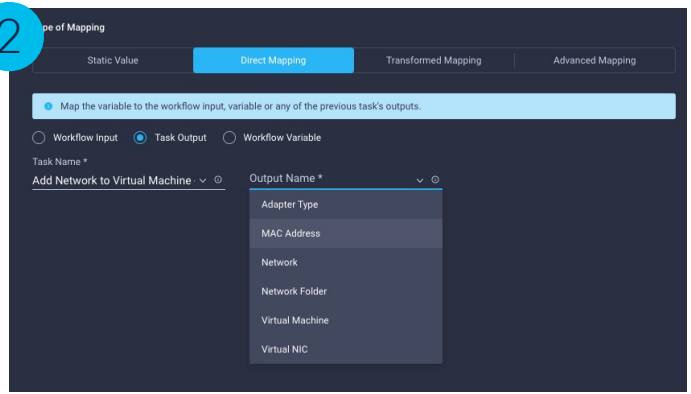
1



Select any task and click on the **Variables** tab.
You will see the configured workflow variables.
If you want, you can also create Workflow variables from there.

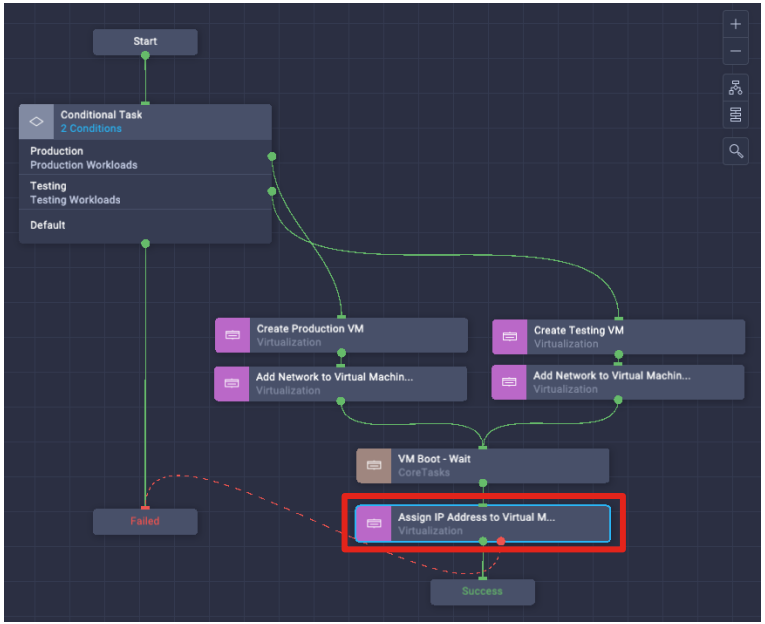
Clicking on **Map** you can assign a value using a **Static Value, Direct Mapping, Transformed Mapping** (if String) or **Advanced Mapping.**
After the execution of the task the variable will have the configured value.

2

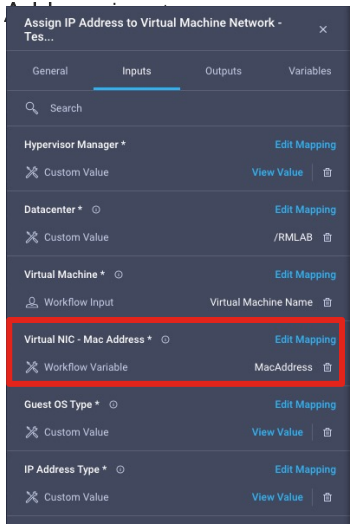


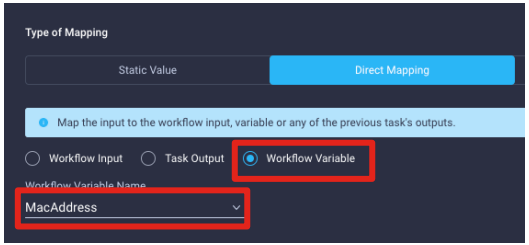
In our example, we will assign as a value the **Output MAC Address** coming from the **Add Network to Virtual Machine** task, which is actually itself. This means that once the task is completed, the variable will contain the MAC address we are looking for.
For this specific case, we do the same operation on the other branch **Add Network to Virtual Machine** task.

Using Variables 3/3



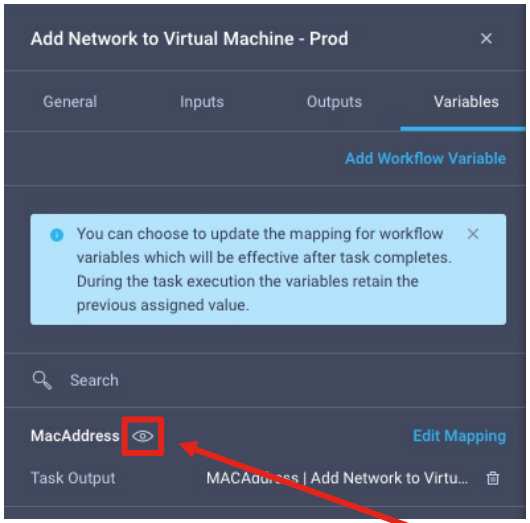
The refactored workflow will look something like this.
We now have only one instance of **Assign IP Address to Virtual Machine**, which uses the variable **MacAddress** as the value for **Virtual NIC – Mac**

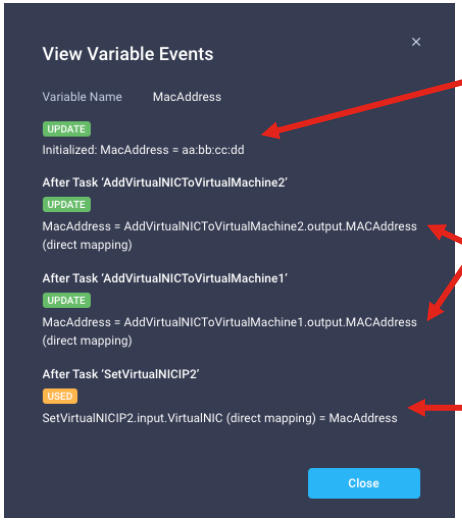




With this implementation, we simplified the workflow from 9 to 7 tasks, removing some redundancies.
The **Assign IP Address to Virtual Machine** task will get the MAC Address from the variable, which gets updated by whatever task is executed, independently on the conditional branch that was executed

Workflow Variable Events





The variable **MacAddress** has a default value of:
aa:bb:cc:dd

Tasks **AddVirtualNICtoVirtualization 1 and 2** (whatever gets executed based on the conditional task) will update the variable based on the MAC address they return as an output

Task **SetVirtualNICIP2** will use the variable (its value) using direct mapping

In the **Variables** tab of every task, clicking on the 'Eye' button, you can view the variable events, which represent the lifecycle of the variable from the beginning of the workflow, to the end.
You can also access variable events in the **General** tab of the workflow, under **Workflow Variables**

Serial Loops

Serial Loops

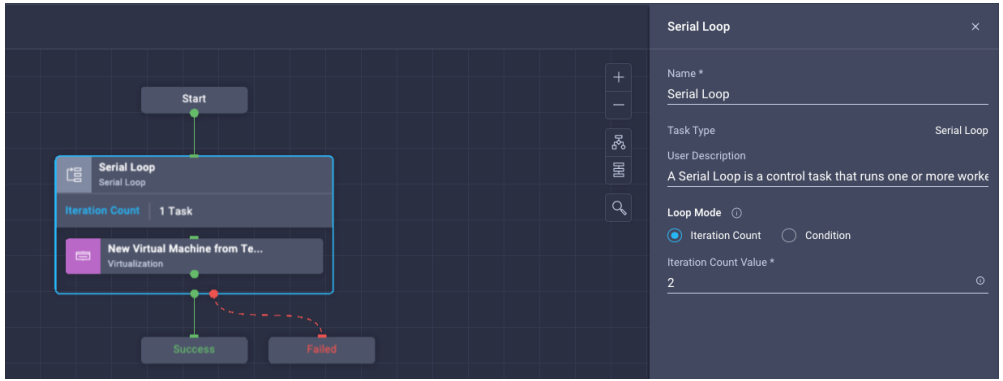
Serial Loops allow a user to execute one or more tasks in a loop.

You can't use Serial Loops, Parallel Loops or Conditional Tasks inside a Serial Loop.
To prevent infinite loops, we enforce the limit of maximum 100 iterations. This number will be gradually relaxed in the future.

Serial Loops can be based on **Iteration Count** or **Condition** (mutually exclusive)

There are three parameters to consider:

1. **Iteration Count**. This parameter specifies the number of times you want to execute the tasks in the loop
2. **Condition.** This parameter specifies a condition for which the loop runs as long as the condition is true *
3. **Iteration.** This parameter has significance only within the individual task execution and represent the current number of the task execution (iteration), similar to Parallel Loops



.iteration = 0

.iteration = 1

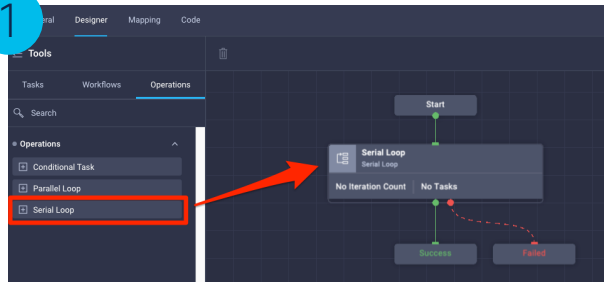
Example:
If you set the **Virtual Machine** name as
`myvm_{{.iteration}}`

ICO will create 2 VM in a serial fashion named: myvm_0, myvm_1, myvm2, ...

*The current serial loop implementation is `do-while`
This means that the first iteration will be **always** executed

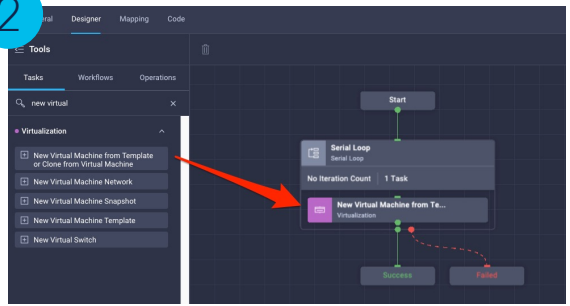
Serial Loops Example – Create Multiple VMs with Count 1/3

1



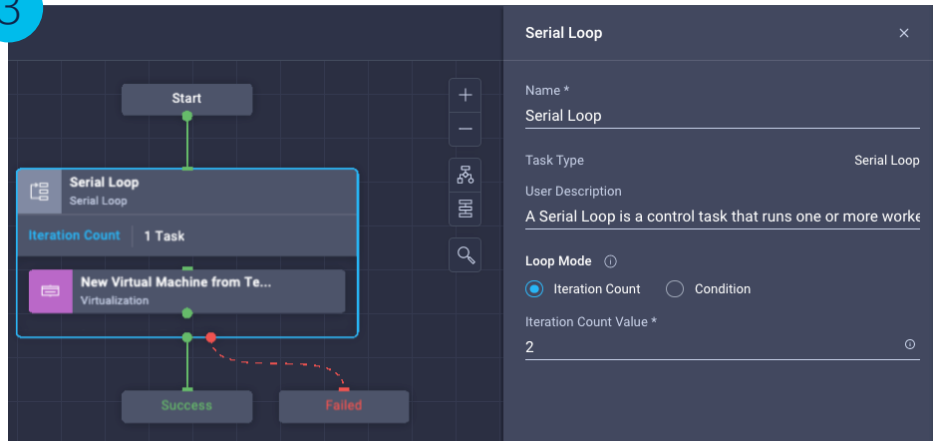
Drag the **Serial Loop** task into the designer

2



Drag one or more tasks **inside** the **Serial Loop** task

3

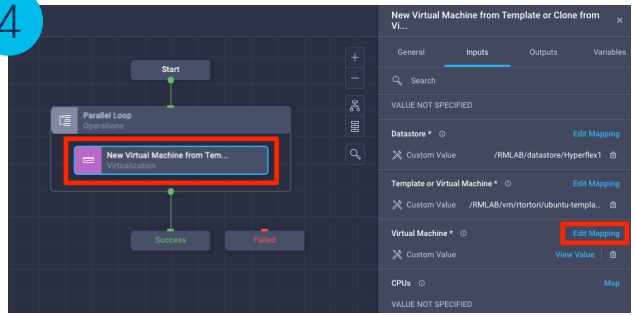


Click on the **Serial Loop** and specify a count value.
This needs to be an integer, you can set it statically (i.e: 2) or use the templating syntax to set it dynamically or from a workflow input as you would do in a **Parallel Loop**

In this case, we want the user to specify the number of execution statically using an integer.

Serial Loops Example – Create Multiple VMs with Count 2/3

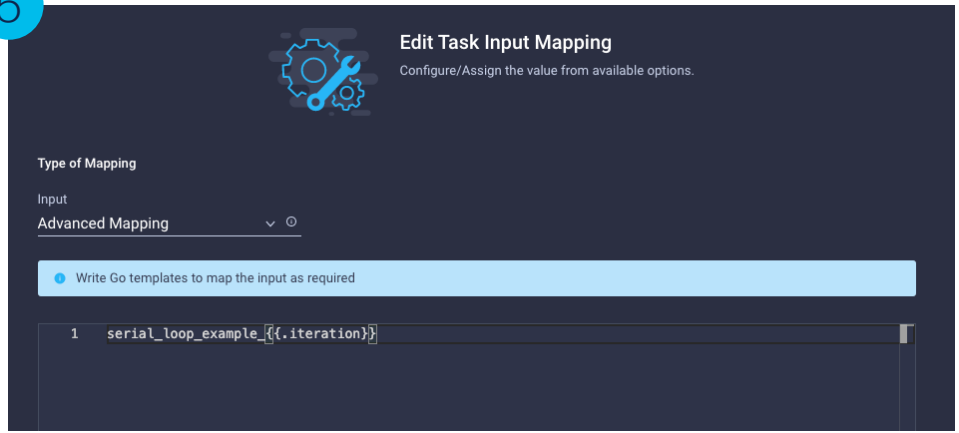
4



In this example, we want to give a distinguished name for each VM, otherwise they will all have the same name.
Click on the **New Virtual Machine from Template or Clone from Virtual Machine** task inside the **Serial Loop** and map the **Virtual Machine** input

*Note: This example just covers the **Serial Loop** use case, it won't cover how to set all required inputs to create a Virtual Machine.*

5



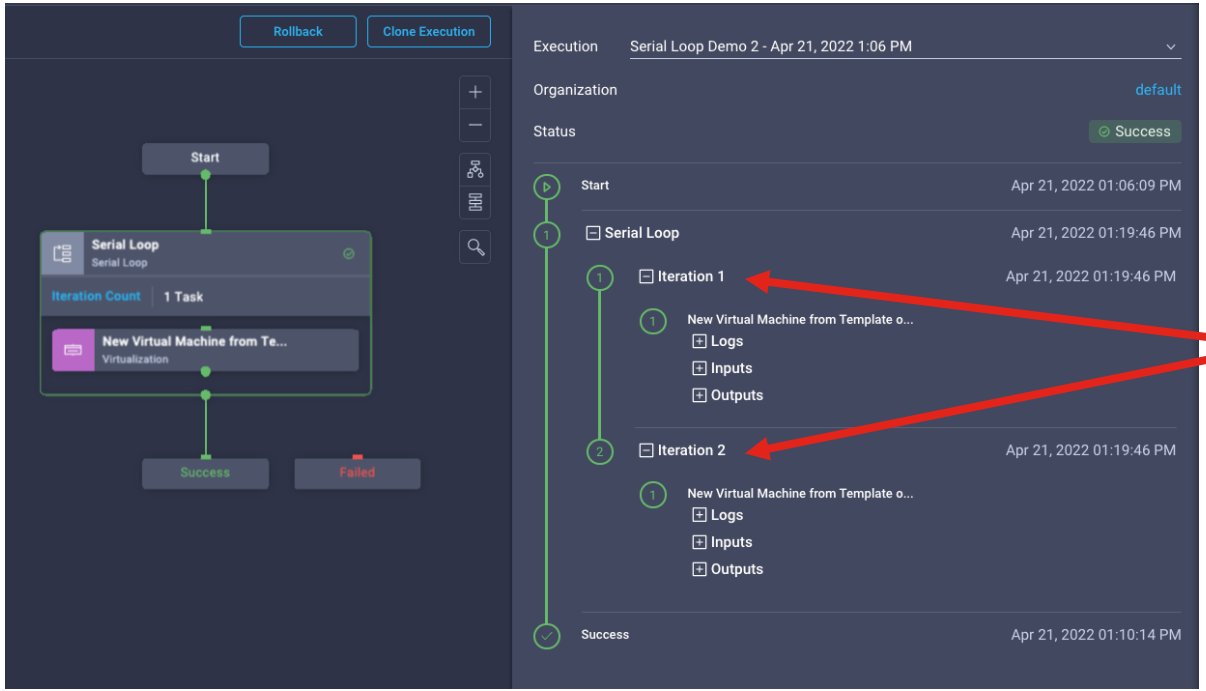
Use **Advanced Mapping** to set the Virtual Machine name:
`serial_loop_example_{{.iteration}}`

Static Naming Prefix — Iteration number

Example: 1

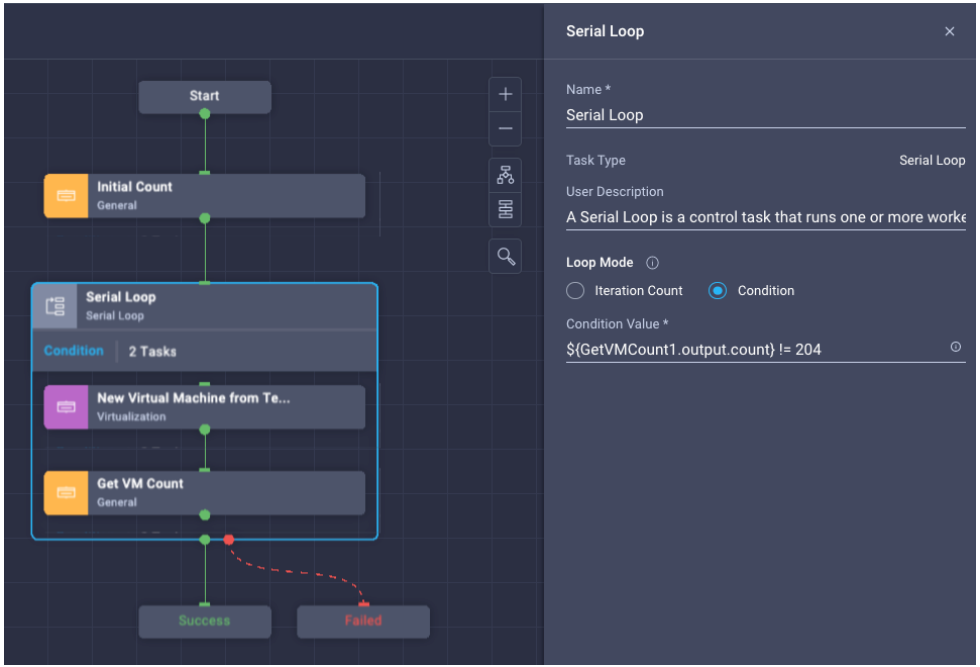
Sample Result: serial_loop_example_1

Serial Loops Example – Create Multiple VMs with Count 3/3



Iteration Count Value = 2

Serial Loops Example – Create Multiple VMs with Condition



In this example, we don't know in advance how many VMs to create.

We are using a custom task that returns the VM count from the target vCenter. For this specific case, we will start from 202, which is the current number of VMs present.

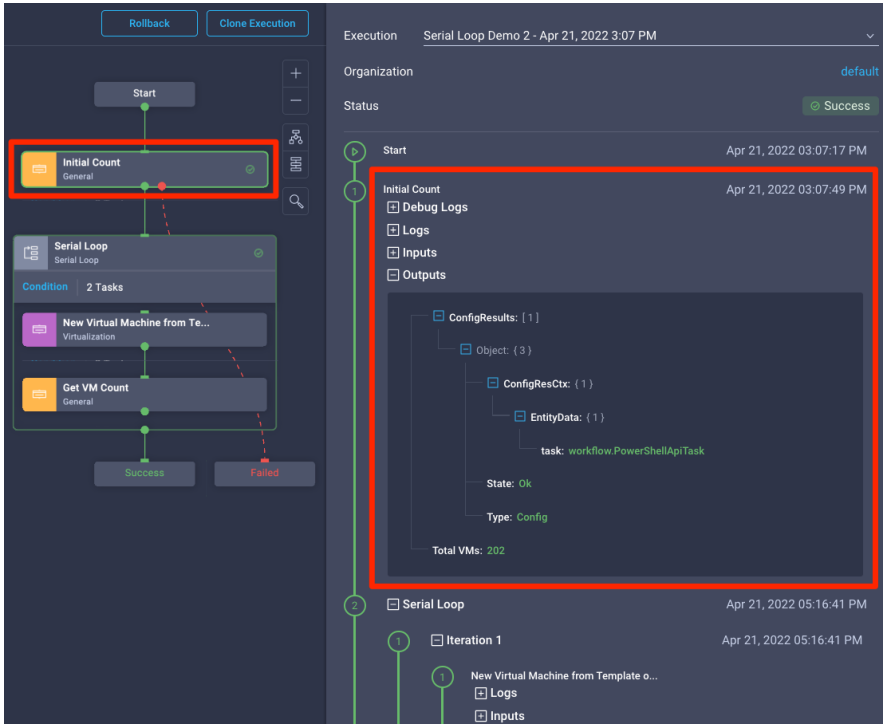
Note that to prevent infinite loops we currently enforce the limit of max 100 iterations.

We set a **Condition** that creates VMs until the total number of VMs in the target vCenter is 204

For this sample scenario, we drag the count task **within** the serial loop, so the loop will execute the VM creation and the count tasks.

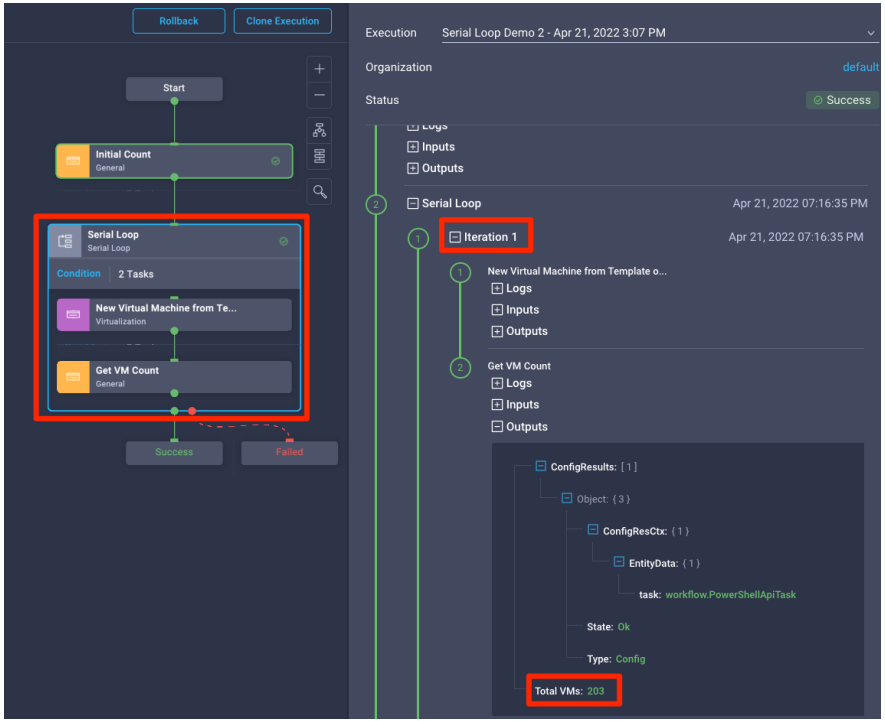
The first iteration will always run, however at the end of each iteration, we check the output of the **Get VM Count** task. If that is different than 204, we keep running, otherwise we exit the loop.

Serial Loops Example – Create Multiple VMs with Condition
Execution



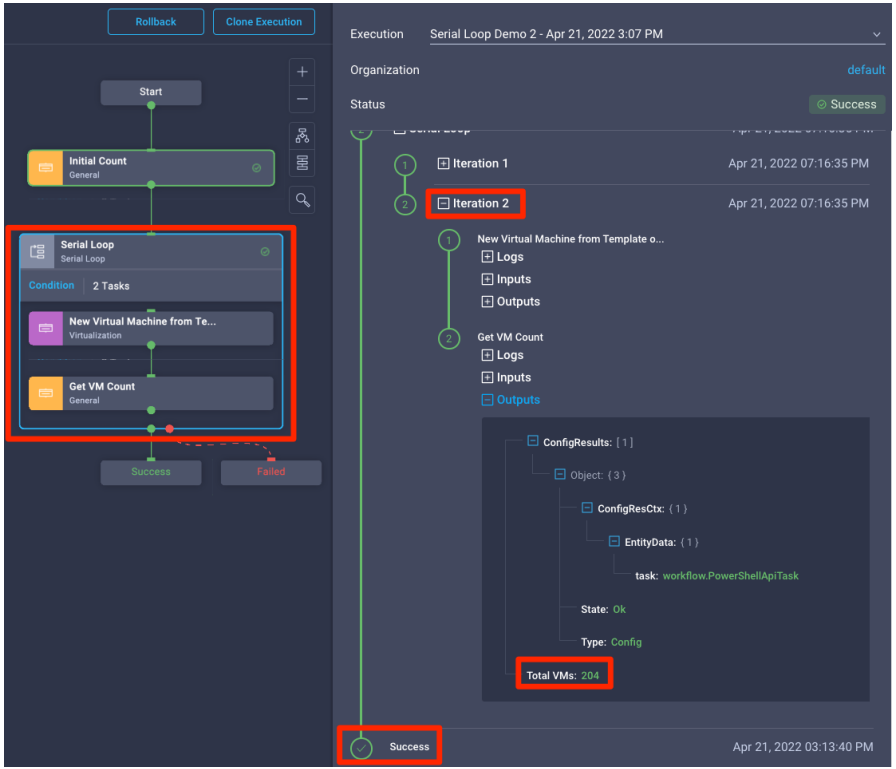
For reference, we run an **Initial Count** outside the loop, which returns 'Total VMs = 202'

Serial Loops Example – Create Multiple VMs with Condition
Execution



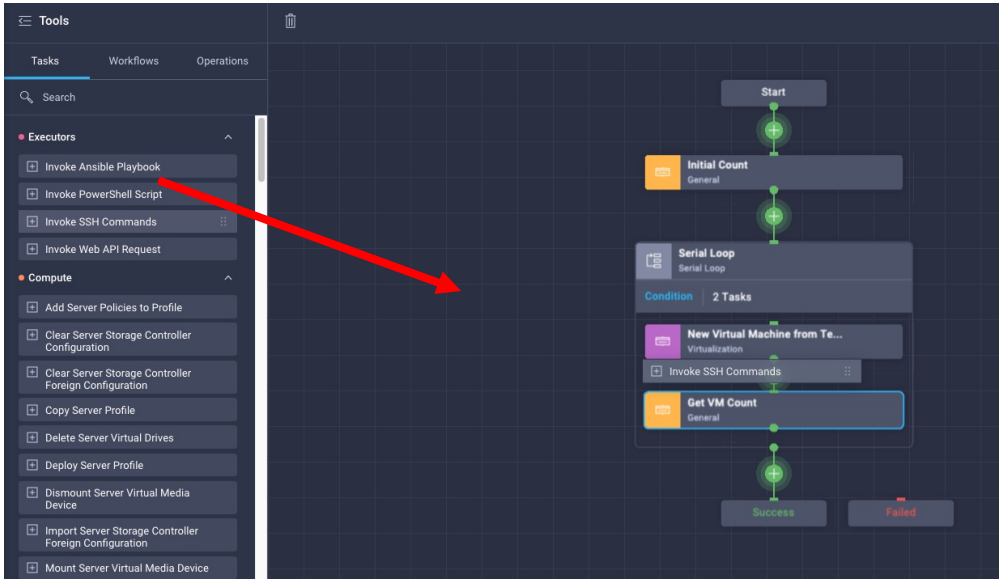
After the first iteration, we count 203 Total VMs

Serial Loops Example – Create Multiple VMs with Condition
Execution

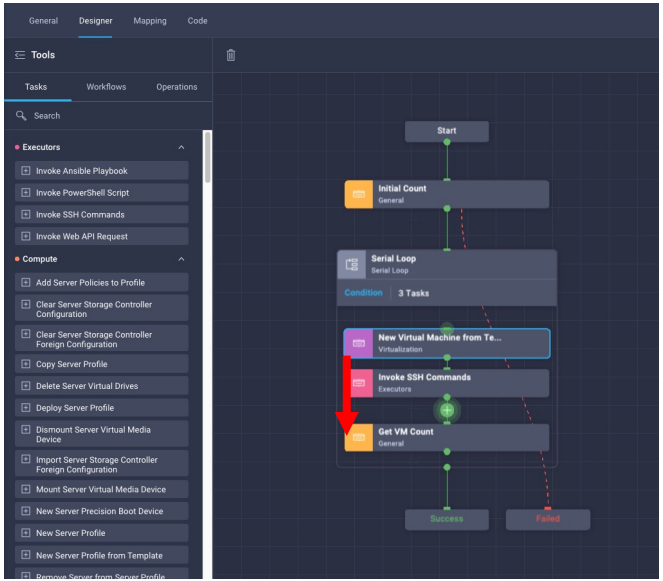


After the second iteration, we count 204 Total VMs, so we stop iterating and exit the loop

Add Multiple Tasks to the Loop and Reordering



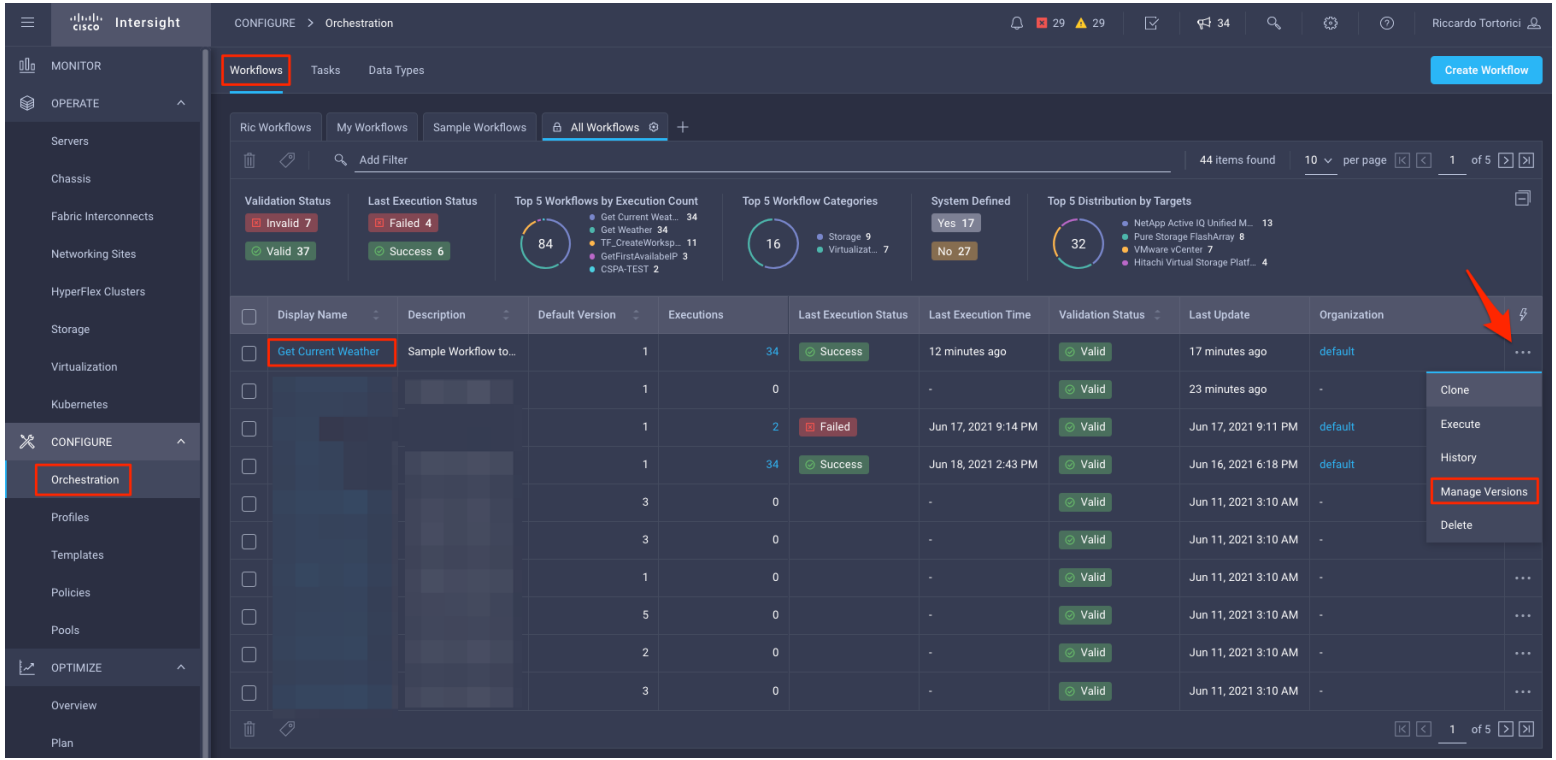
Drag a task and drop it in one of the drop zones within the loop box



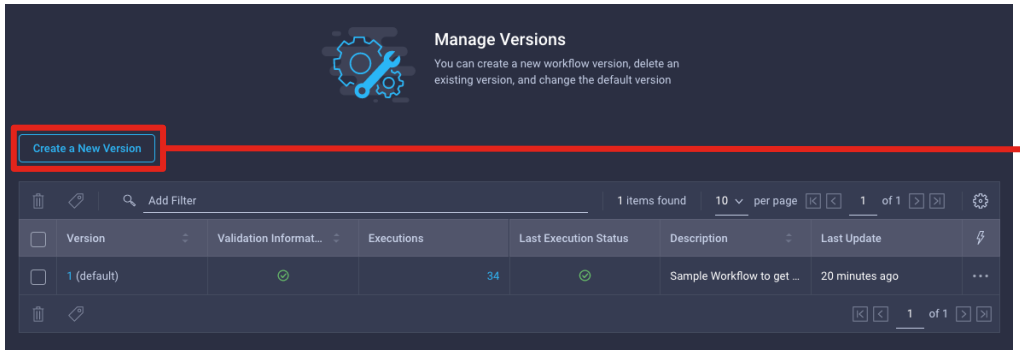
Drag a task and drop it in one of the drop zones within the loop box to rearrange the task order

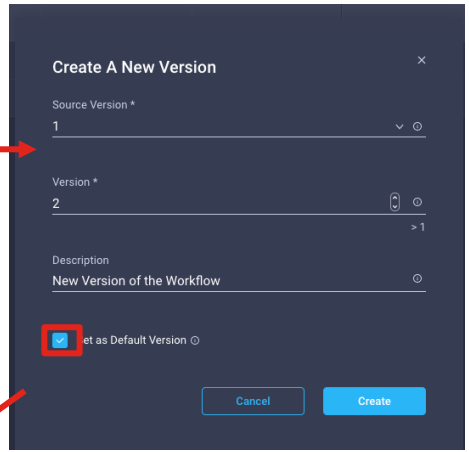
Workflow Versions

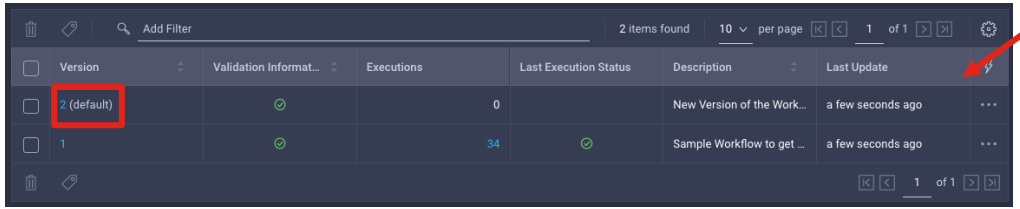
Managing Workflow Versions



Managing Workflow Versions



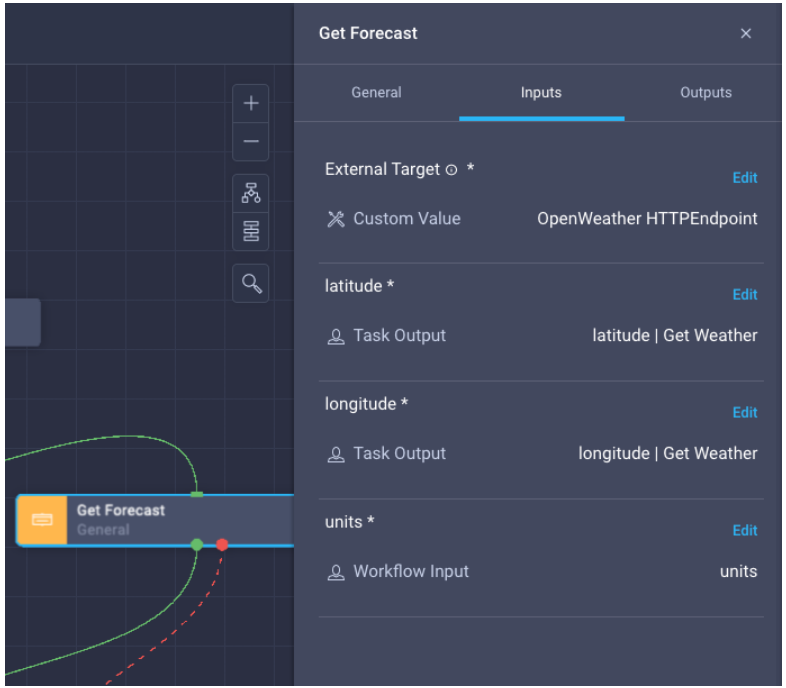


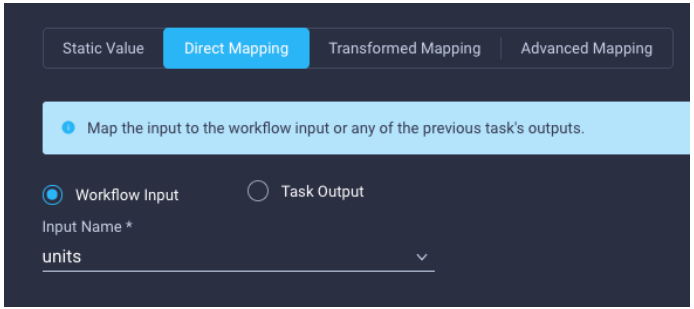


If **Set as Default Version** is checked, all execution will use version 2 if no version is specified.
You can execute specific version of the workflow or set any existing version as the default

Transformations

Input Transformations

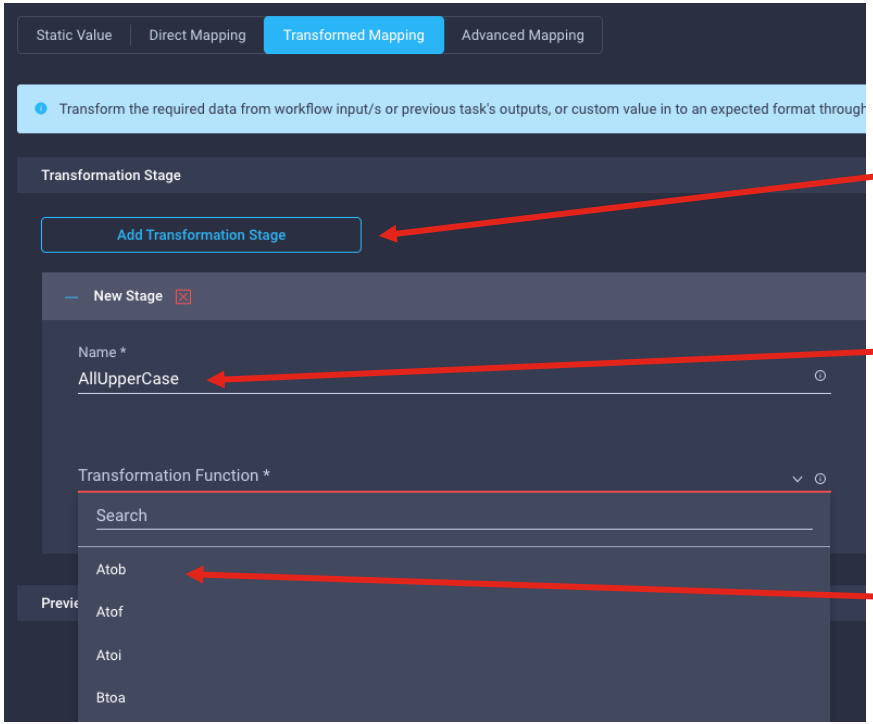




Scenario:
The input 'units' for the task 'Get Forecast' is currently set as a **Direct Mapping** to the 'units' workflow input.
We want to manipulate this input before it get processed by the 'Get Forecast' task.

To do so, we need to use **Trasformed Mapping**

Input Transformations

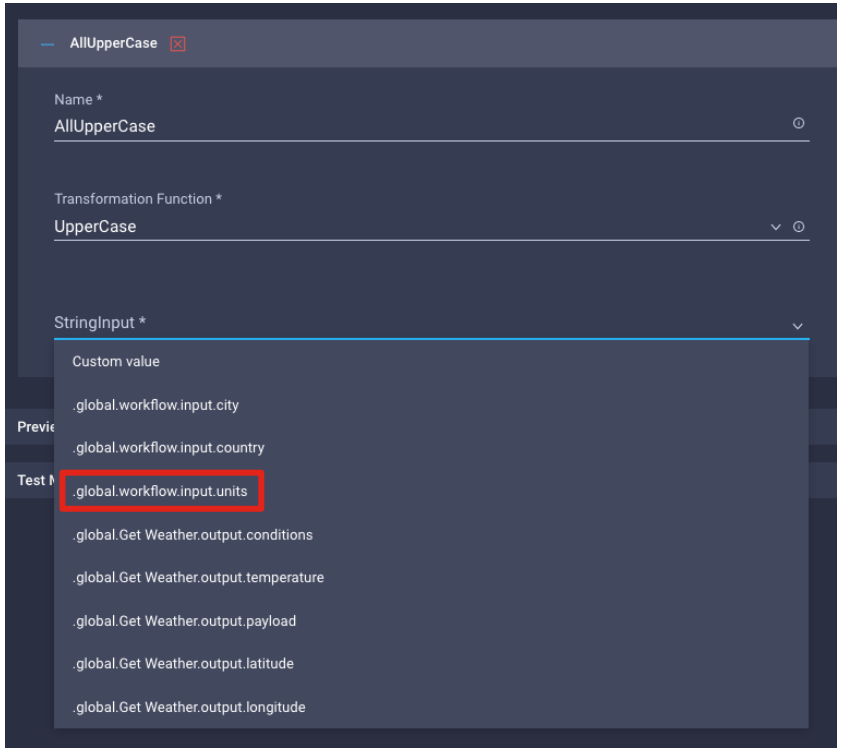


You can add one or multiple **Transformation Stages.**

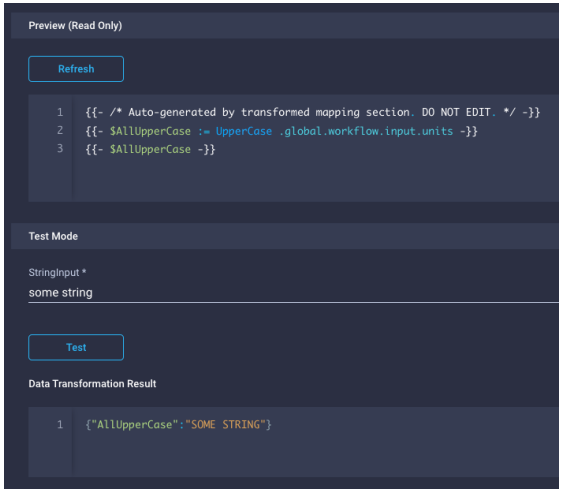
Arbitrary Name - less than 64 characters and can only contains letters (a-z,A-Z) or numbers

Select a **Transformation Function**

Input Transformations

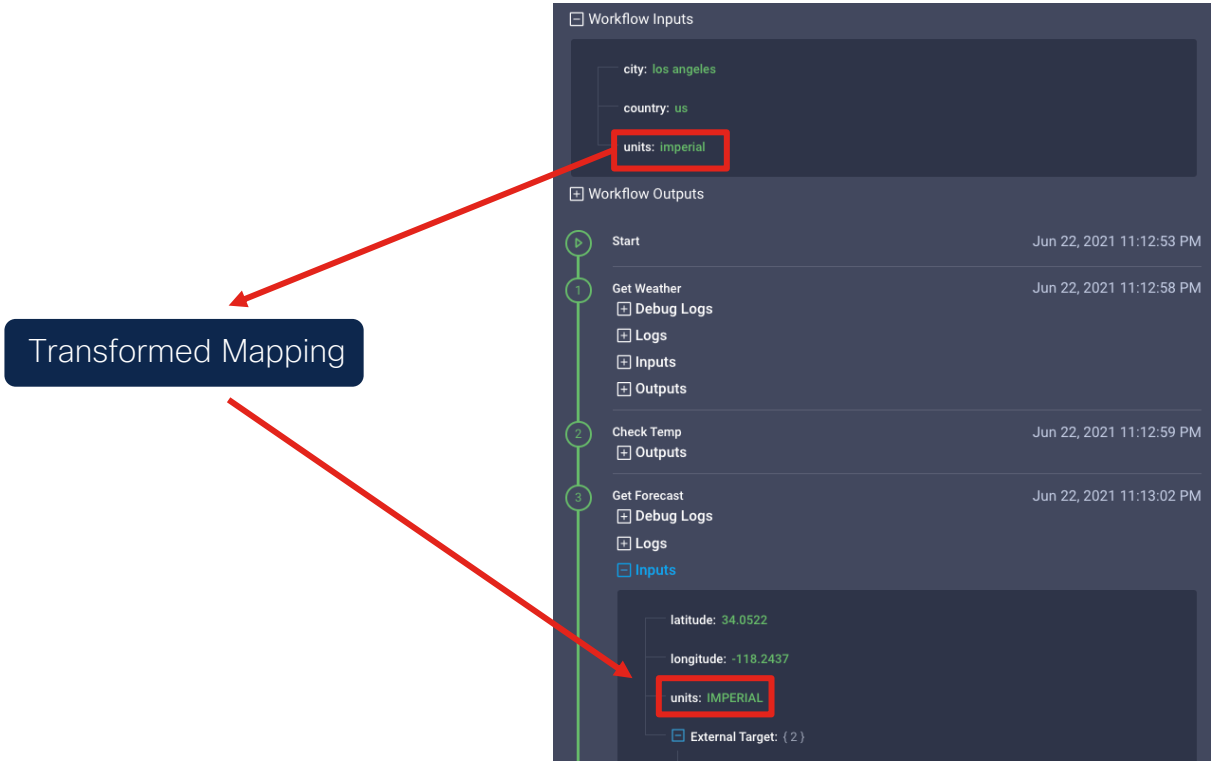


Select the **UpperCase** transformation function and the **StringInput**. This could be either one of the input/outputs or a **Custom Value.**



In this example, we picked the 'units' workflow input (.global.workflow.input.units)
You can **Test** the results in real time and **Preview** the corresponding Go Template

Verify Input Transformation



Transformation Functions 1/2

`FindAllString` returns a slice of all substrings that match given regular expression in given string.

`TrimSpace` returns a slice of the string s, with all leading and trailing white space removed, as defined by Unicode.

`ContainsString` reports whether substr is within s.

`Atoi` is equivalent to ParseInt(s, 10, 0), converts provided string to Type int.

`StringReplace` returns a copy of the string s with the first n non-overlapping instances of old replaced by new. If old is empty, it matches at the beginning of the string and after each UTF-8 sequence, yielding up to k+1 replacements for a k-rune string. If n < 0, there is no limit on the number of replacements.

`UpperCase` returns s with all Unicode letters mapped to their upper case.

`LowerCase` returns s with all Unicode letters mapped to their lower case.

`ToJson` returns the json format of given value that can be a number, slice, object and any golang Type that can be marshalled into json.

`Ftoa` returns the given value as string with given precision. By default golang uses the E-notation for large float64 values. This function allows the float64 numbers to be replaced with proper floating point notation.

`Itoa` converts the given int value in base 10 as string.

`Atof` converts the given string to a floating point number.

Transformation Functions 2/2

`Ftoi` converts the given float to an integer.

`Itof` converts the integer to a floating point number.

`IntToBool` converts 0 and 1 to false and true respectively.

`BoolToInt` converts true and false to 1 and 0 respectively.

`Atob` converts string values of true and false to boolean.

`Btoa` Converts boolean value to a string.

`Substring` returns the substring within the given index bounds of the given string.

`IpToCidr` retrieve CIDR notation for Ipv4.

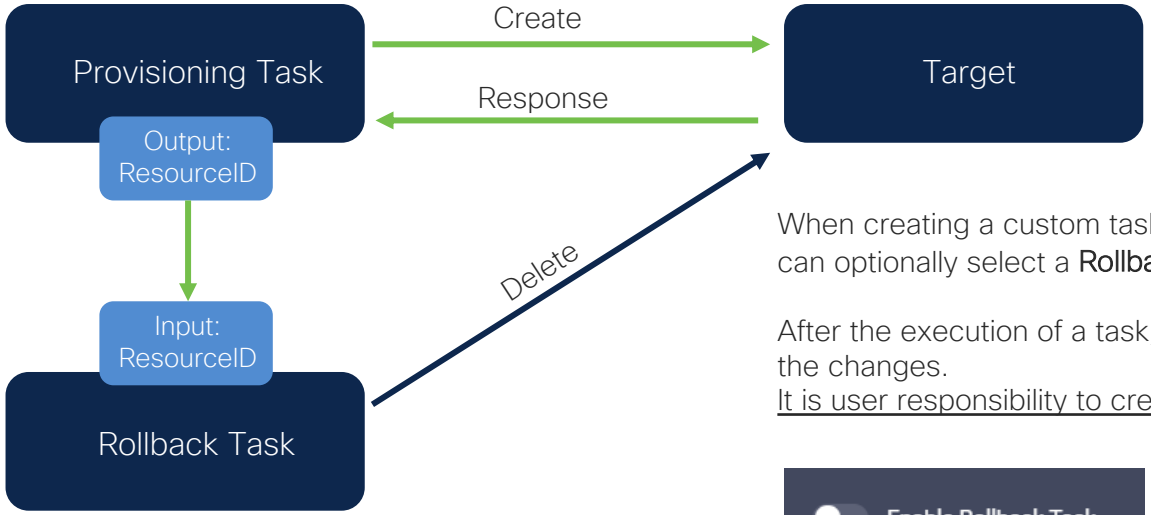
`CalculateDate` takes a datetime input in string, and returns a new datetime in standard RFC3339 format with an adjustment in seconds based on a time.Duration input. Both positive and negative increments can be handled.

`CalculateDateDifference` takes two datetime Inputs as strings, a base date and a date to be adjusted against, and returns the difference in duration between the two in seconds. It can return a positive or negative value, depending on the relation between the two datetime values.

`FormatDate` takes two datetime Inputs, one being the input date and another containing the reference layout string to which the input date has to be converted. All reference datetimes in Go follow the same format with an error being

Rollback Tasks (Task Designer)

Concept



When creating a custom task using the **Web API Request** executor, you can optionally select a **Rollback Task**.

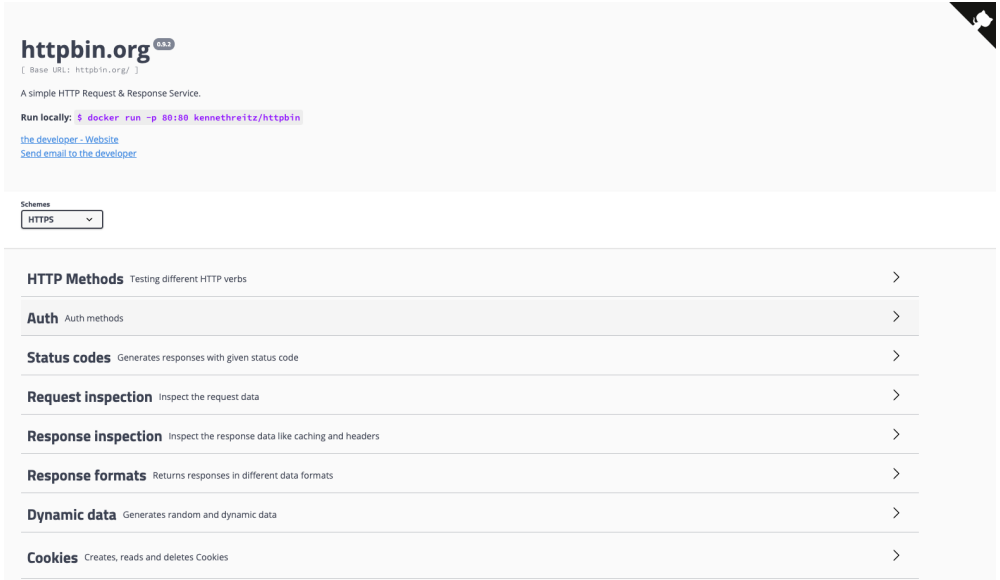
After the execution of a task, you will be given the opportunity to rollback the changes.
It is user responsibility to create a working logic to rollback a custom task

Example Scenario

Scenario:
Claim an HTTP endpoint in Intersight using the API and rollback execution

Web API Request Target:
Intersight

HTTP Target:
Endpoint: https://httpbin.org
Name: httpbin
Unauthenticated
Direct connection (no Intersight Assist)



REST API Call to Claim a Target in Intersight

Method: POST
Path: /api/v1/asset/Targets
JSON Payload:

```
{
  "Connections": [
    {
      "ManagementAddress": "httpbin.org",
      "Port": 443,
      "IsSecure": true,
      "Credential": {
        "ObjectType":
"asset.NoAuthenticationCredential"
      },
      "ObjectType": "asset.HttpConnection"
    }
  ],
  "TargetType": "HTTPEndpoint",
  "ManagementLocation": "Intersight",
  "Name": "httpbin"
}
```

Sample Response (partial information shown)

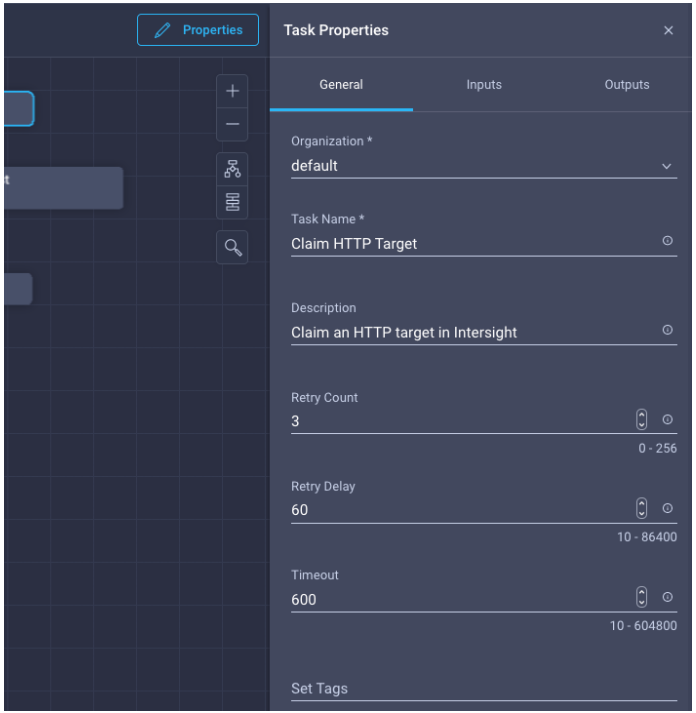
```
{
  "Moid": "60d25b3e6f72612d316c836f",
  "ObjectType": "asset.Target",
  "ClassId": "asset.Target",
  […]
  "Connections": [
    {
      "ClassId": "",
      "ObjectType": "asset.HttpConnection",
      "Credential": {
        "ClassId":
"asset.NoAuthenticationCredential",
        "ObjectType":
"asset.NoAuthenticationCredential"
      },
  […]
  "Assist": null,
  "RegisteredDevice": null
}
```

REST API Call to Unclaim a Target in Intersight

Method: DELETE
Path: /api/v1/asset/Targets/<Target MOID>

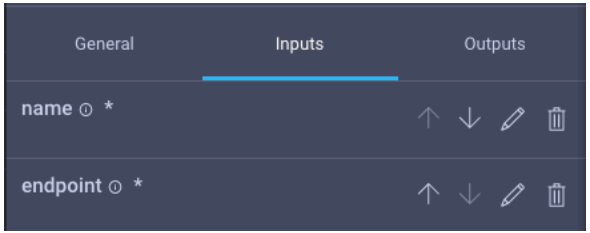
No Response provided

Creating the HTTP Endpoint Claim Task

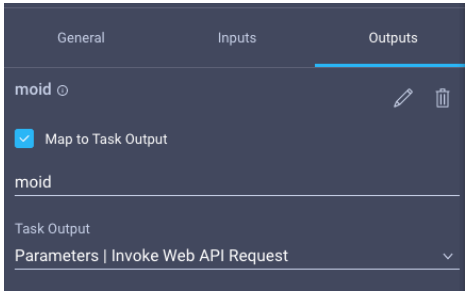


Task Properties – General
Name: Claim HTTP Target

Task Properties – Inputs
name – String, Required
endpoint – String, Required



Task Properties – Outputs
moid – String, Map to Task Output: 'Parameters | Invoke Web API Request'



Creating the HTTP Endpoint Claim Task

Executor Properties – General
Name: Claim HTTP Target
Set External Target: NO (will use Intersight as API target)

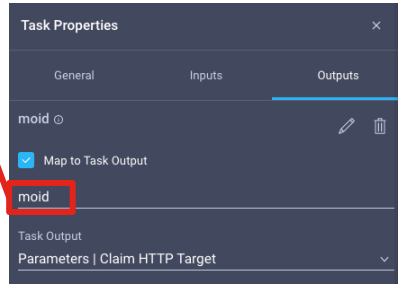
Executor Properties – Inputs
Method: POST
URL: Custom Value – `/api/v1/asset/Targets`

Body (Content Type: JSON):

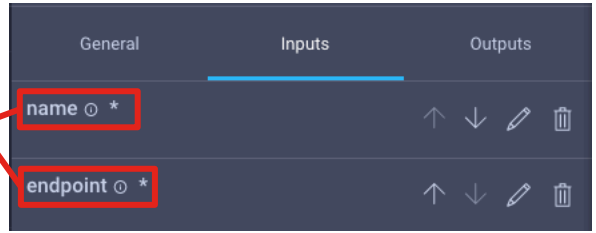
```
{
  "Connections": [
    {
      "ManagementAddress": "{{.global.task.input.endpoint}}",
      "Port": 443,
      "IsSecure": true,
      "Credential": {
        "ObjectType": "asset.NoAuthenticationCredential"
      },
      "ObjectType": "asset.HttpConnection"
    }
  ],
  "TargetType": "HTTPEndpoint",
  "ManagementLocation": "Intersight",
  "Name": "{{.global.task.input.name}}"
}
```

Response Parser Parameters:

Path: $.Moid
Name: moid
Type: String

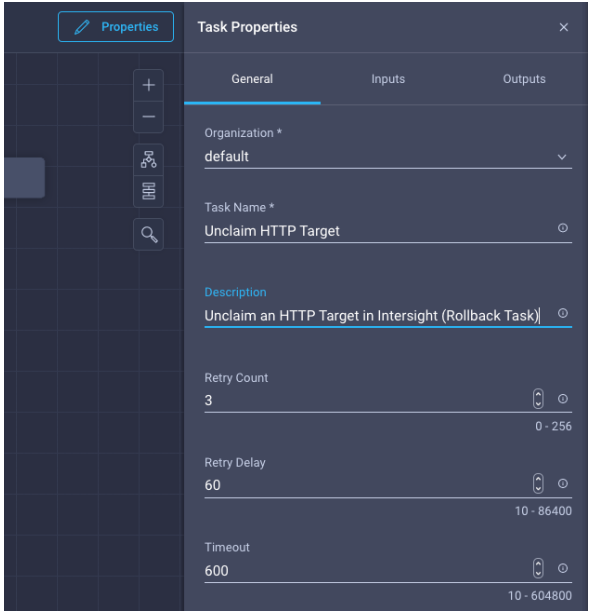


We extract the Moid from the response and output it as this will be used as an input for the rollback task



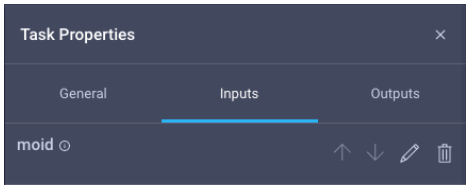
Using templates in the Body will get those values replaced by the Task Input values

Creating the HTTP Endpoint Rollback Task



Task Properties – General
Name: Unclaim HTTP Target

Task Properties – Inputs
moid – String, Required

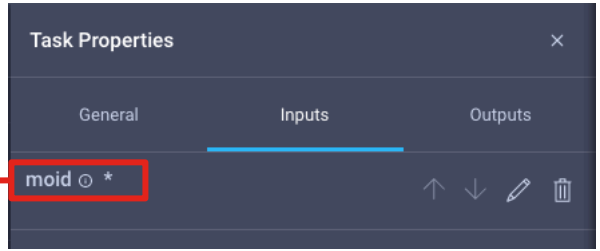


Task Properties – Outputs
None

Creating the HTTP Endpoint Rollback Task

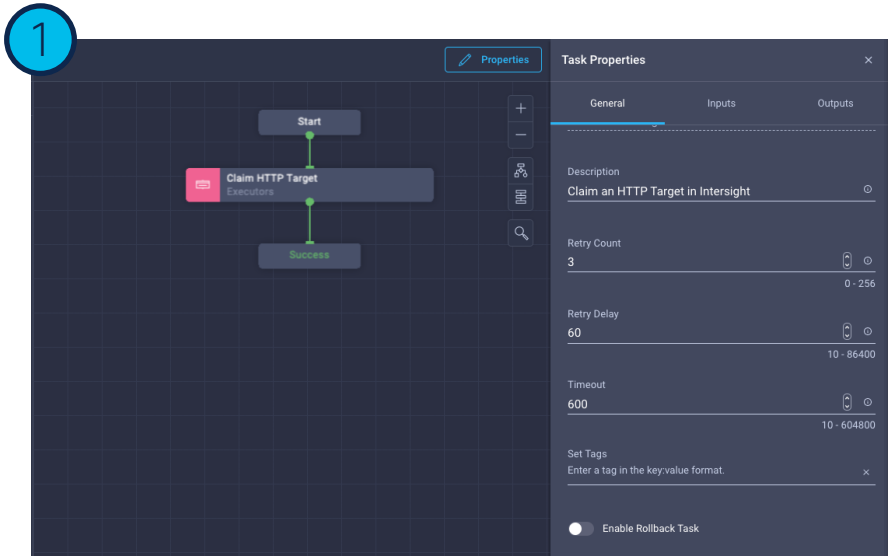
Executor Properties – General
Name: Unclaim HTTP Target
Set External Target: NO (will use Intersight as API target)

Executor Properties – Inputs
Method: DELETE
URL: Custom Value – `/api/v1/asset/Targets/{{.global.task.input.moid}}`

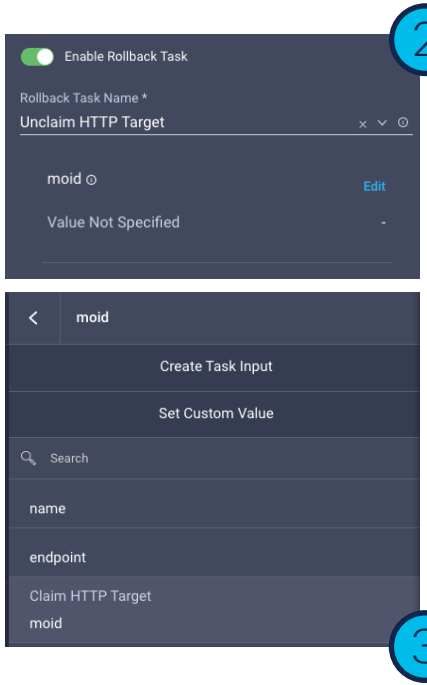


Part of the URL will be replaced with the value of the 'moid' task input

Link the Rollback Task to the Claim Task



Back in the Claim task, in the **Task Properties**, **Enable Rollback Task**

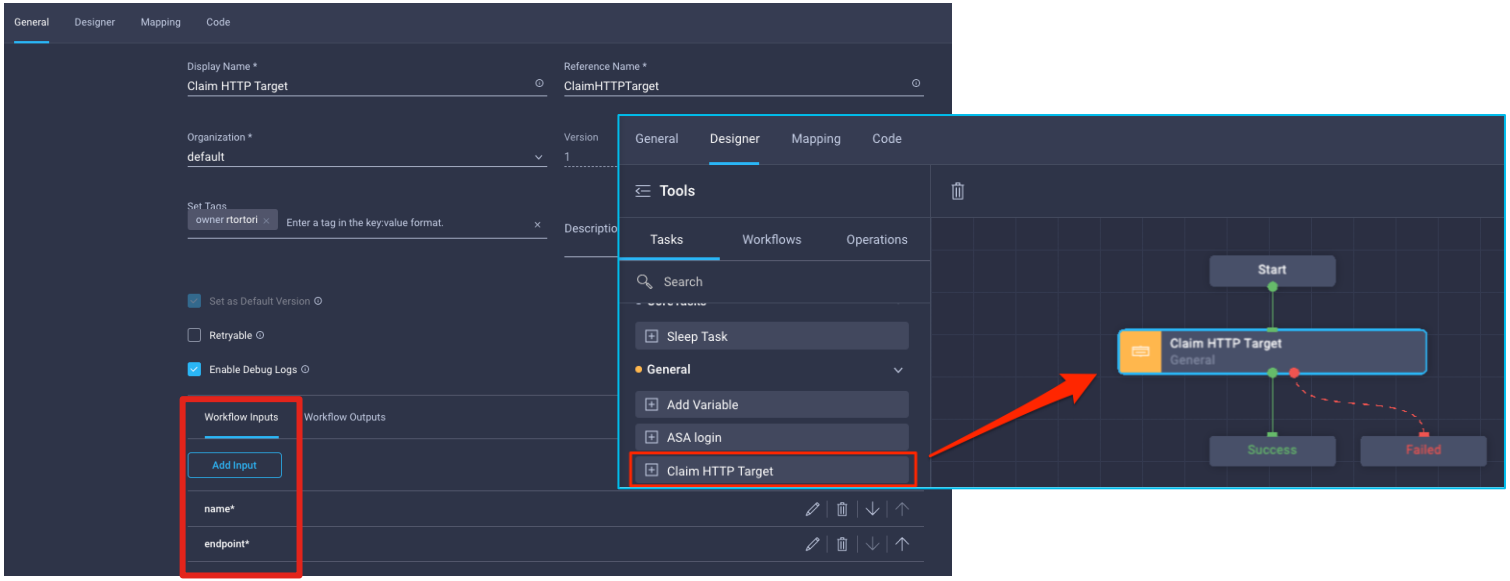


Select a **Rollback Task Name** from the list ('Unclaim HTTP Target)

The rollback task requires an input (we created it).

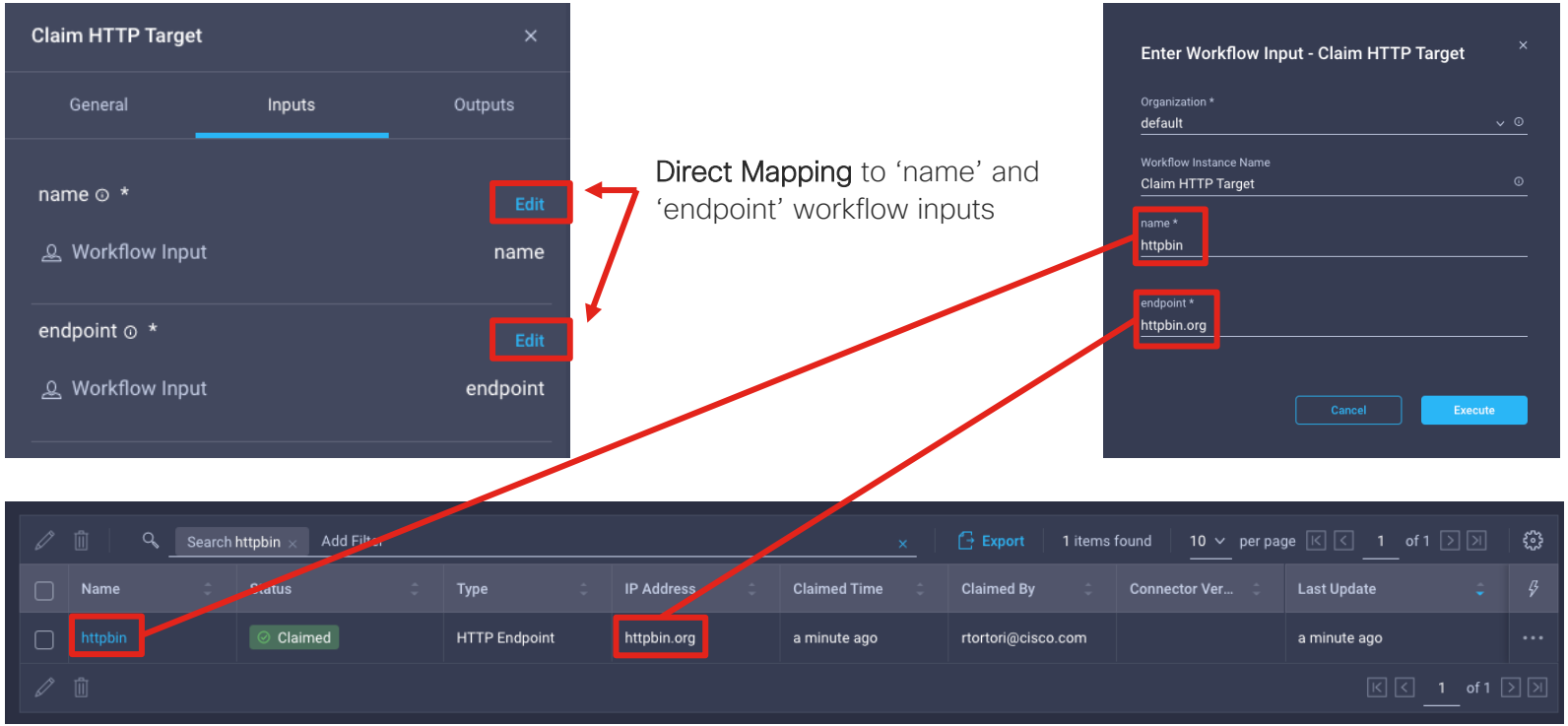
Click on **edit** than click on the 'Claim HTTP Target', 'moid' task output.

Create a Workflow to Claim an HTTP Target

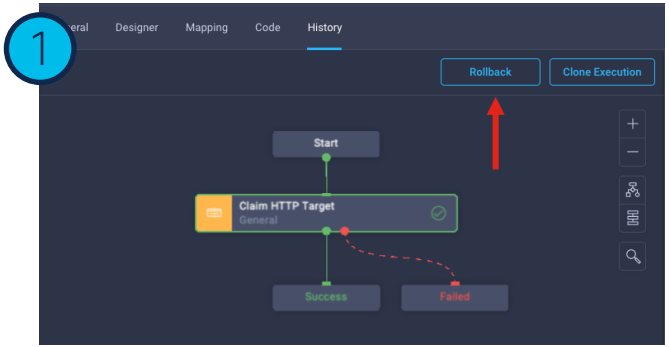


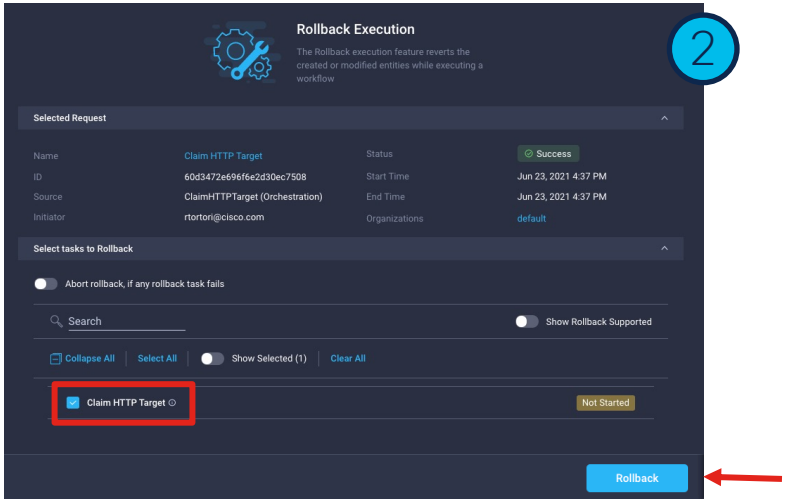
Configure Inputs for ‘name’ and ‘endpoint’ (String, Required)

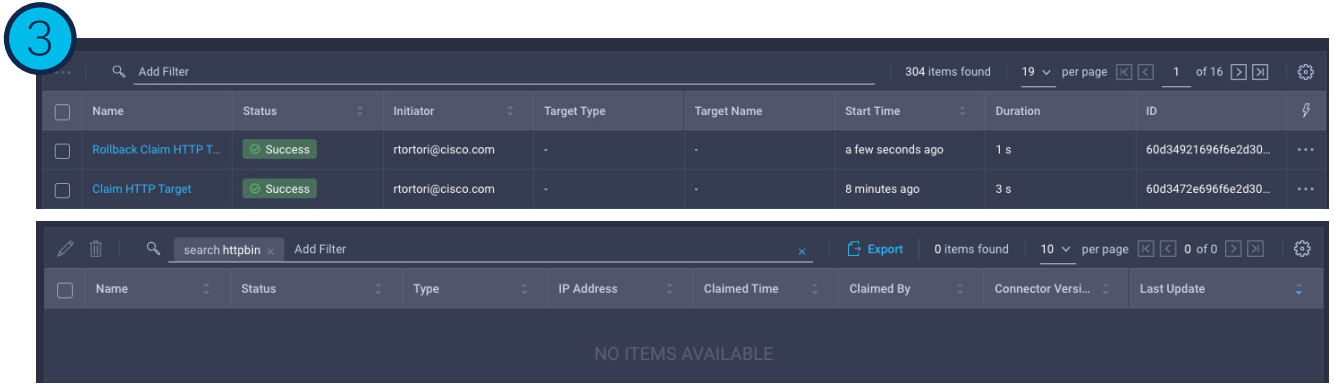
Create a Workflow to Claim an HTTP Target



Rollback Execution



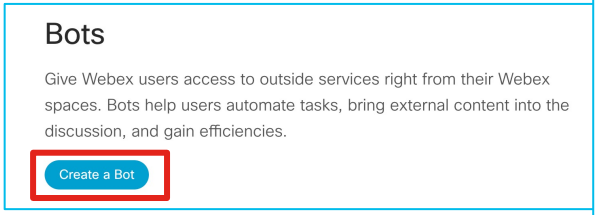


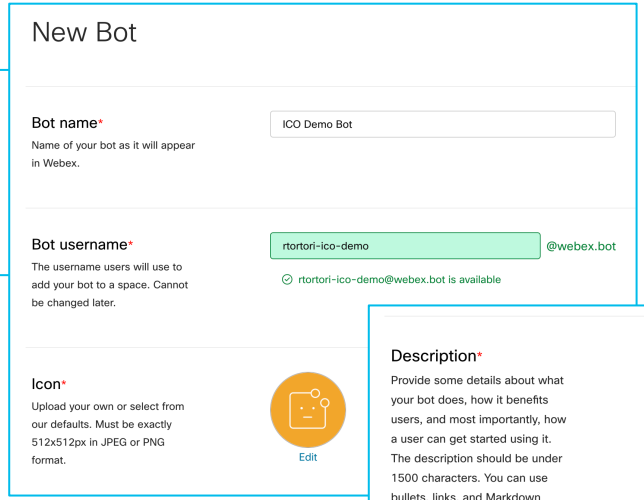


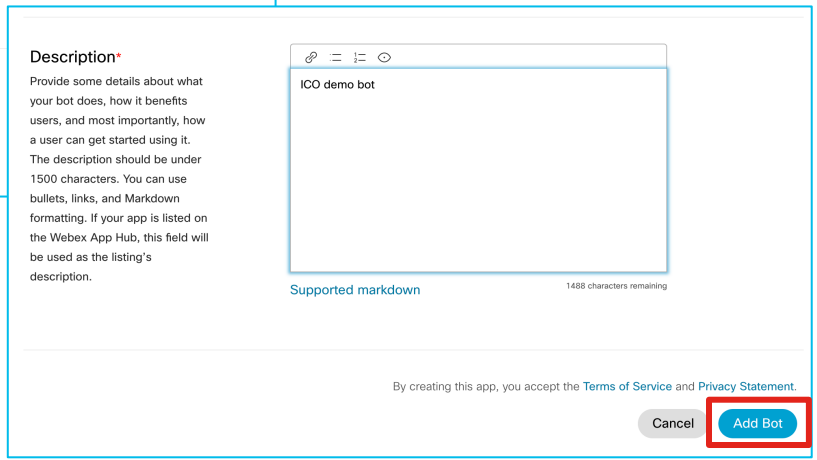
Create a Notification Task with Webex

Create a BOT in webex.com

https://developer.webex.com/docs/bots



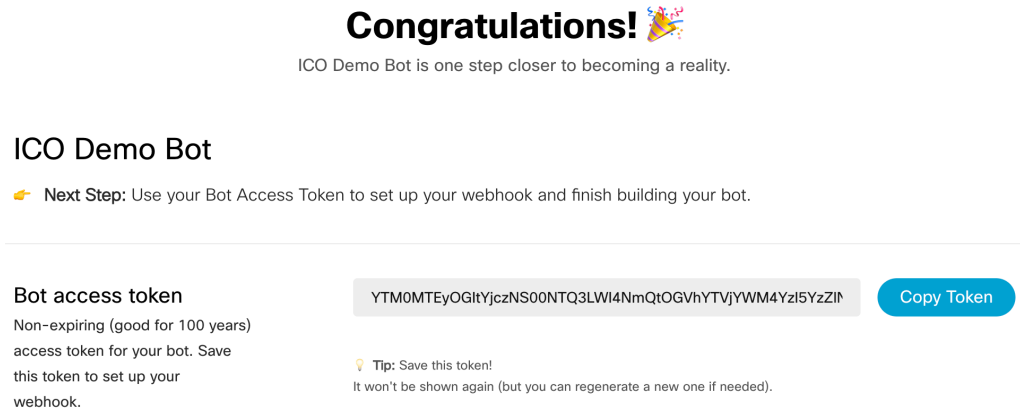




Provide the details necessary to create a bot, such as a name, username, icon, description.

Once the BOT is created, it will redirect to a page where details of the BOT are present; a bot access token is also provided

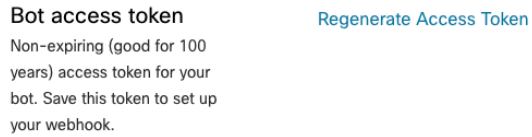
Copy the BOT Access Token



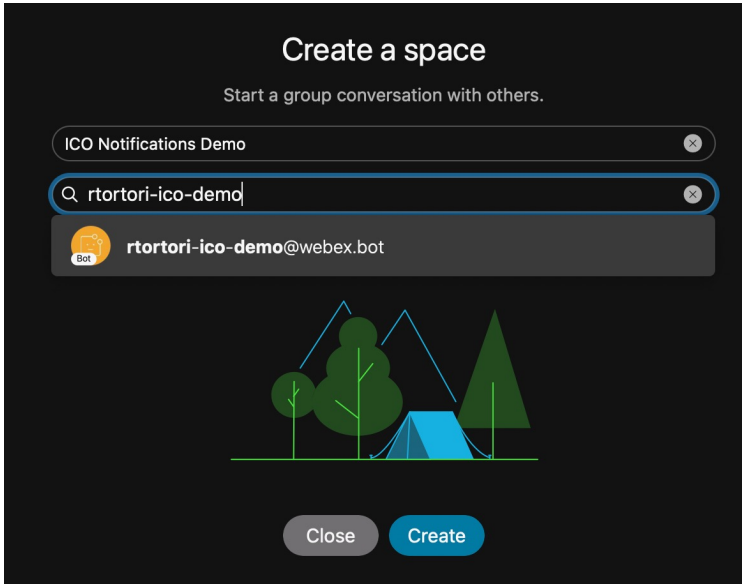
A generated token will be valid for 100 years

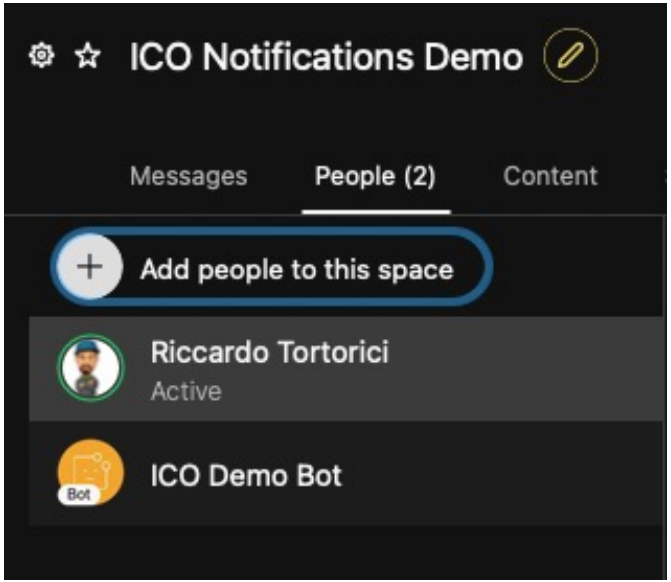
Copy the token as this is the only time you will access it

If needed, you can generate a new one.



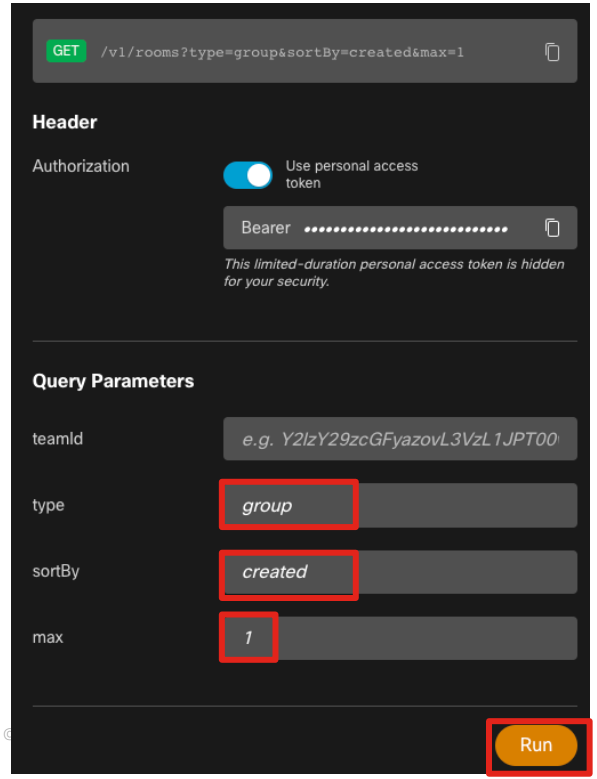
Create a Webex Space and Invite your BOT





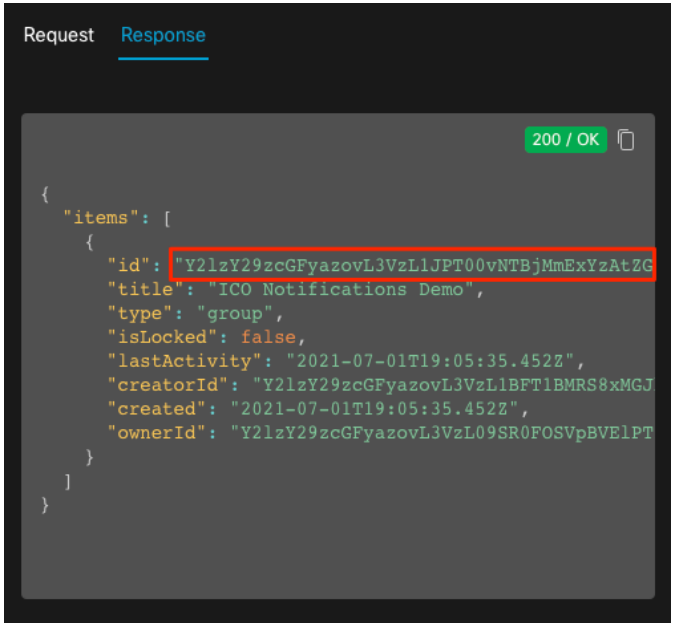
Get the Room ID

https://developer.webex.com/docs/api/v1/rooms/list-rooms



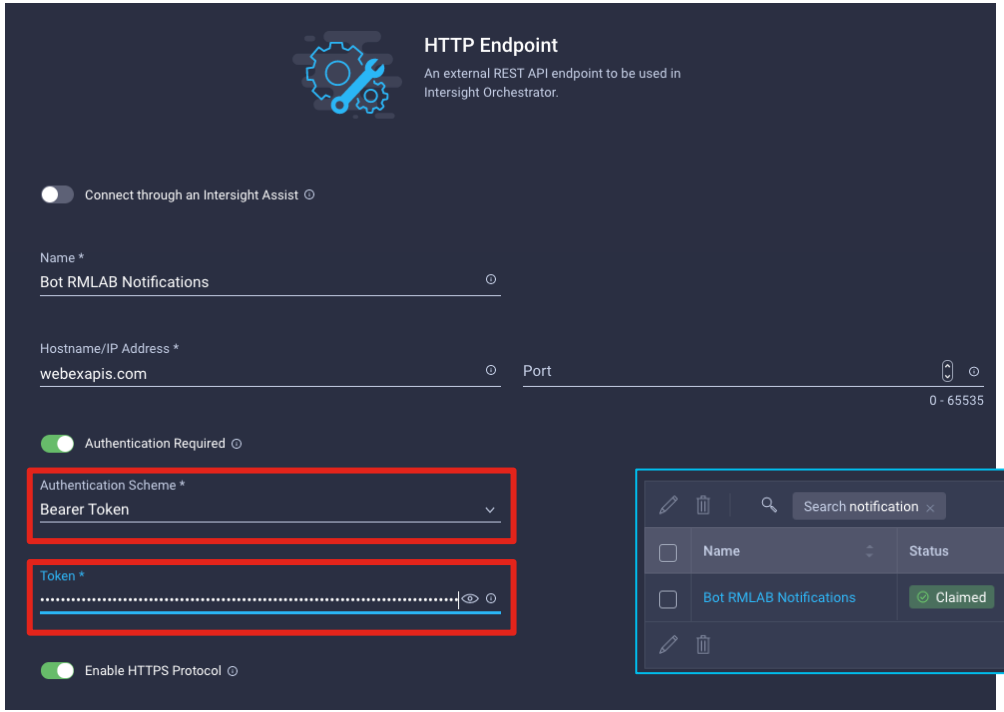
This will return the last group created. If you leave the default, you will get all rooms you are subscribed to. Feel free to experiment with **parameters** if needed.

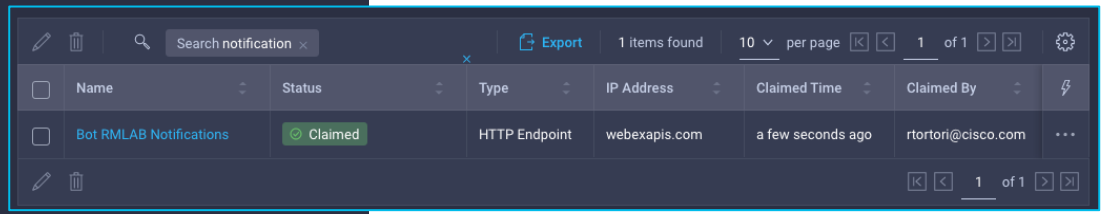
As an anternative, you can use the BOT access token to fetch the list of rooms where the BOT is present



Take note of the room **id**
*Note: the **id** in this example is truncated on purpose*

Claim Webex in Intersight



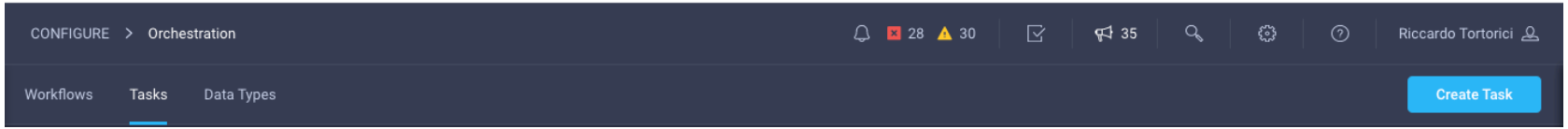


Hostname/IP Address: webexapis.com
Bearer Token **Authentication Scheme**
Enable HTTPS Protocol

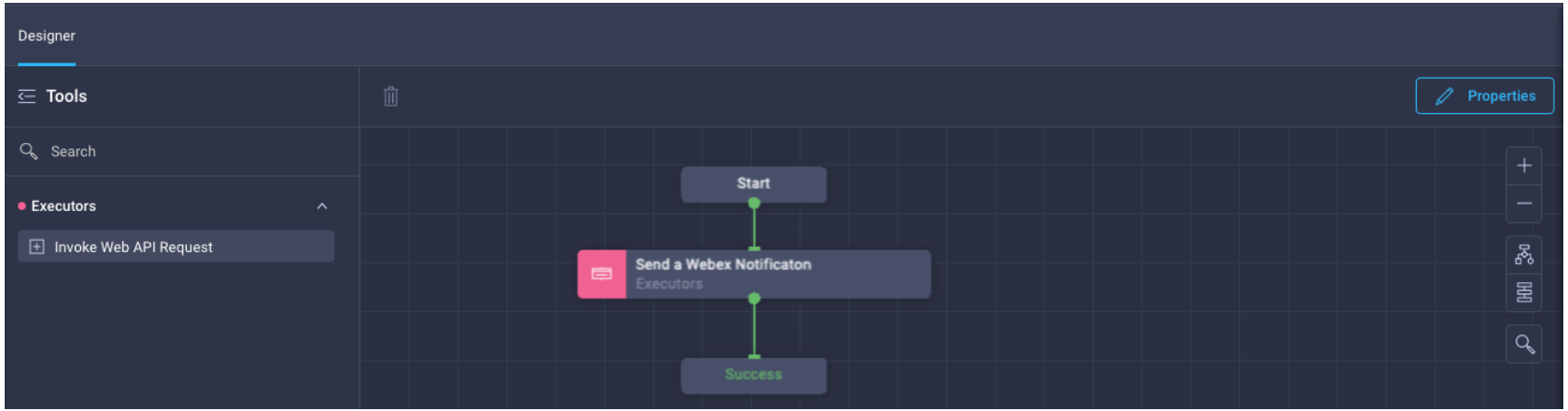
Copy the BOT access token in the **Token** field

Click **Claim** when done.

Create a Notification Task in the Task Designer

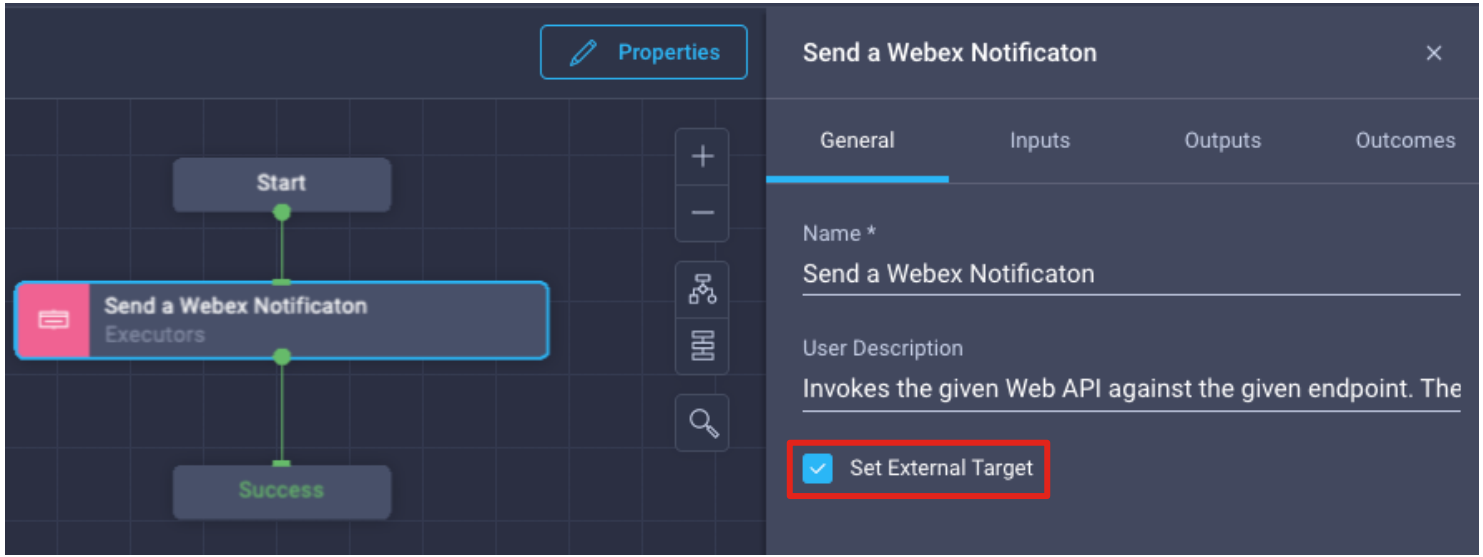


Orchestration -> Tasks -> **Create Task**



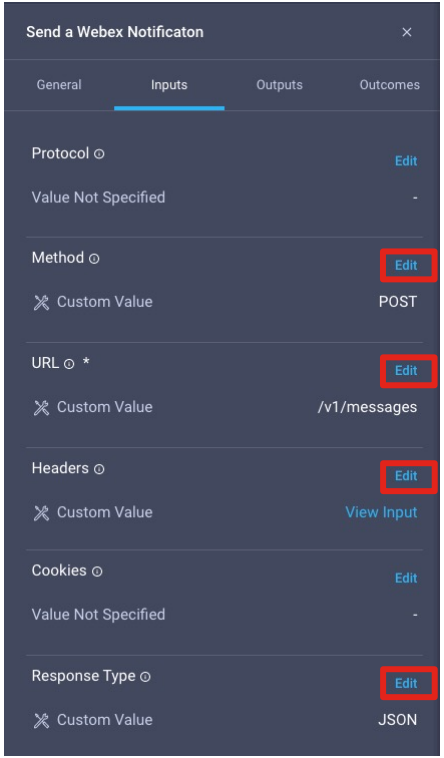
Drag the **Invoke Web API Request** in the designer

Executor Task Properties: General



Under **General**, **Set External Target** and give the task a **Name**

Executor Task Properties: Inputs



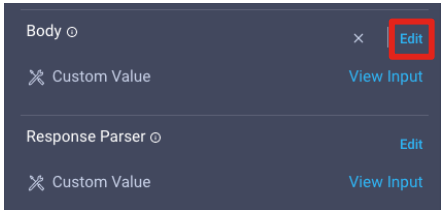
Method: POST

URL: /v1/messages

Headers:

```
{
"Content-Type": "application/json"
}
```

Response Type: JSON

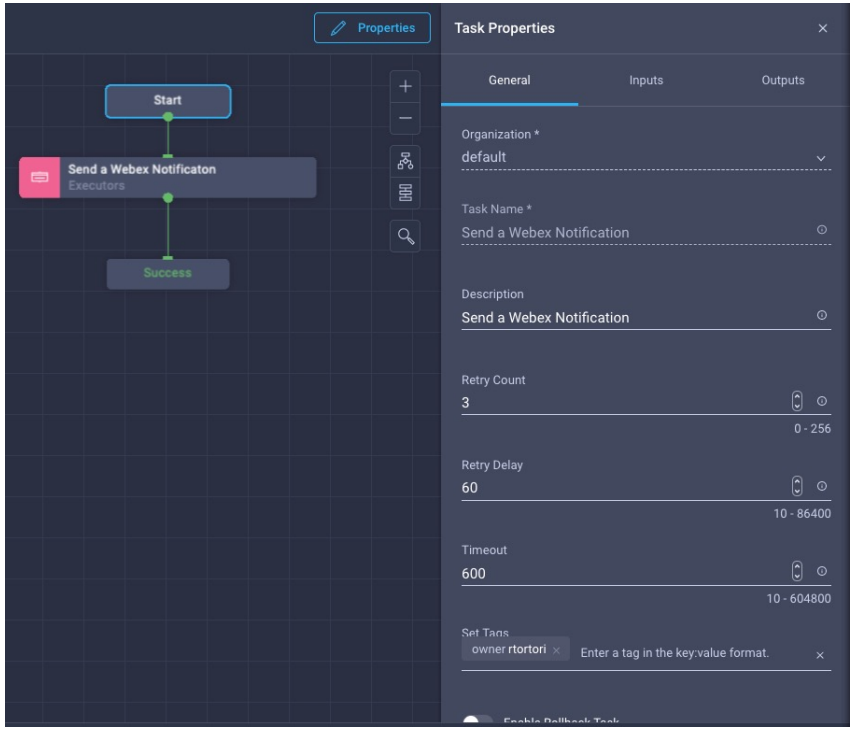


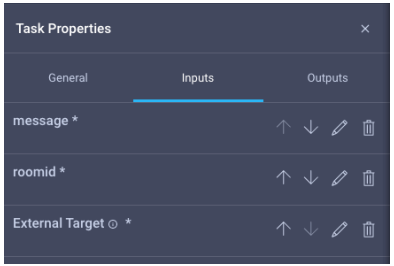
Body:

```
{
  "roomId": "{{.global.task.input.roomid}}",
  "markdown": "{{.global.task.input.message}}"
}
```

`.global.task.input.roomid` and `.global.task.input.message` map to task inputs (yet to be created, see next slides) defining the room id to post a message to (retrieved a few slides back from the Webex API) and the actual message respectively. With this configuration, we can send arbitrary messages base on other task input/outputs in our workflows)
Refer to the documentation to get more details on the many parameters you can use to send messages, attachments, etc : https://developer.webex.com/docs/api/v1/messages/create-a-message

Custom Task Properties





`.global.task.input.roomid` and `.global.task.input.message` map to task inputs (yet to be created, see next slides) defining the room id to post a message to (retrieved a few slides back from the Webex API) and the actual message respectively.

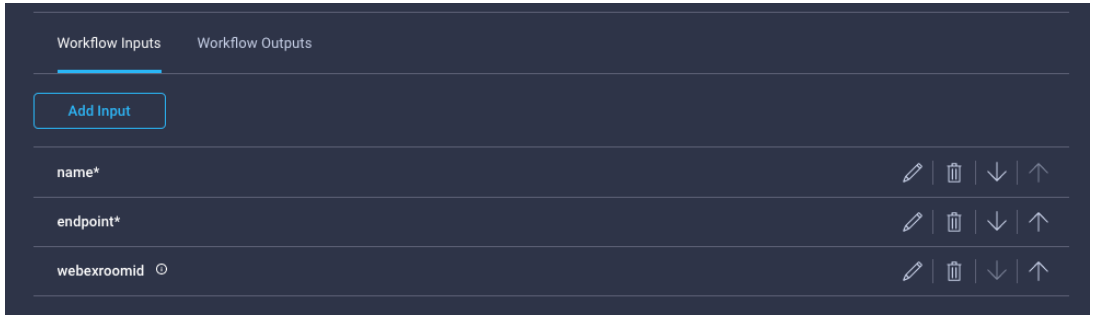
With this configuration, we can send arbitrary messages based on other task input/outputs in our workflows)
Refer to the Webex API documentation to get more details on the many parameters you can use to send messages, attachments, etc :
https://developer.webex.com/docs/api/v1/messages/create-a-message

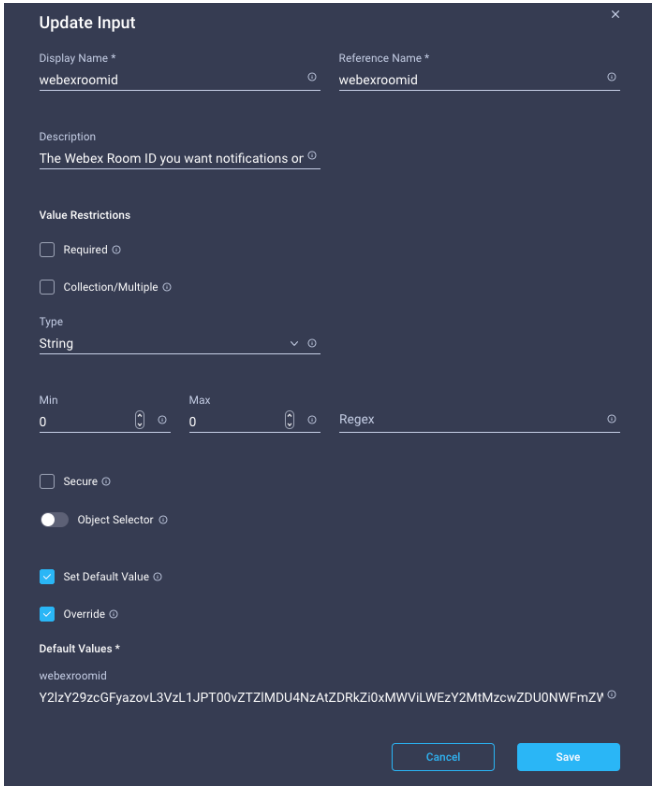
Use the Notification Task

You can create a new workflow or just use an existing one. For this example we will use the HTTP Target Claim task we created a few slides back.

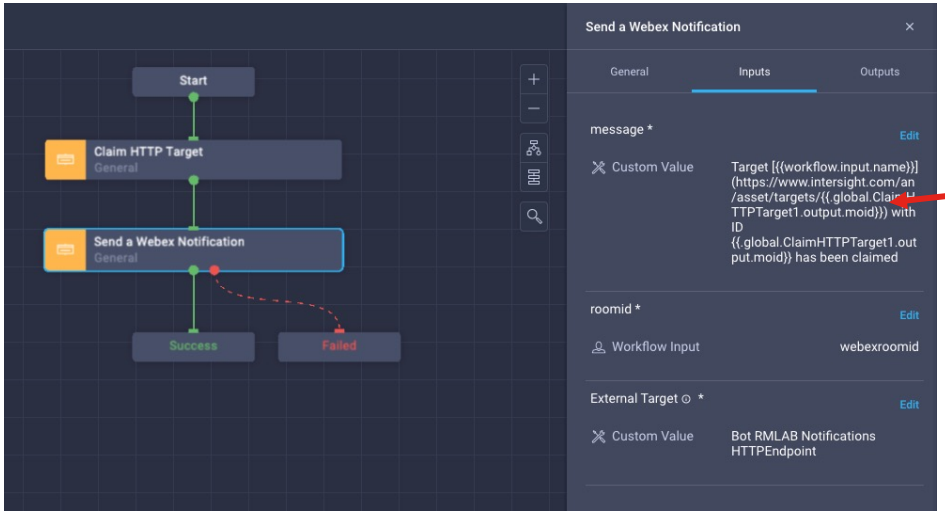
First, we add an additional **Workflow Input** called 'webexroomid' for the Webex Room ID we want our messages notified on. While this can be hardcoded, you may want to leverage different rooms based on the workflow you run.

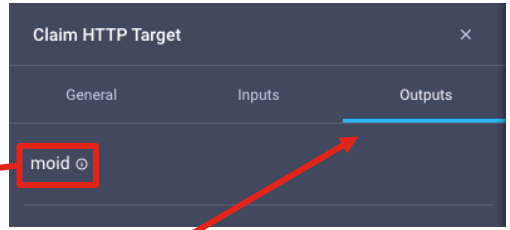
We set it as a **String**. You can optionally set a default value if you don't want to enter the room ID on each execution.





Use the Notification Task





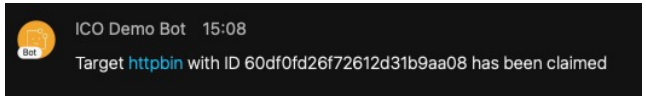
Recall the **Claim HTTP Target** task returns the target MOID as an output.

You can leverage this output to build a custom message in your BOT.

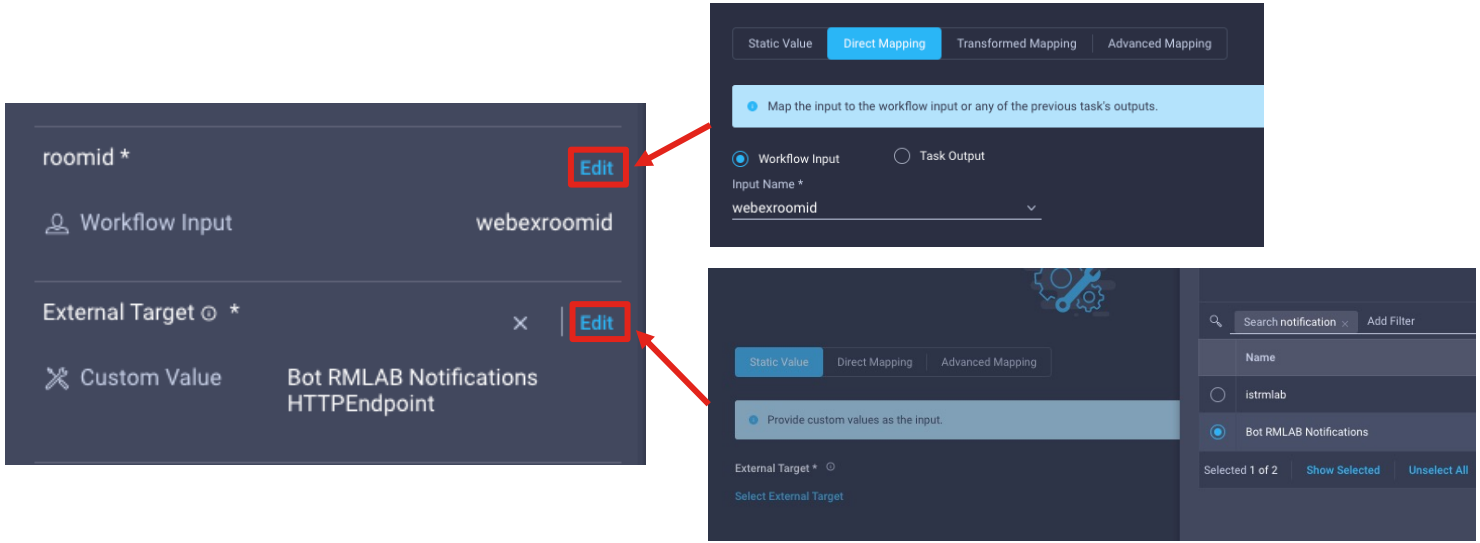
For the message input mapping we will leverage the **Advanced Mapping** as we use templates to render a custom message:

```
Target [{{.global.workflow.input.name}}](https://www.intersight.com/an/asset/targets/{{.global.ClaimHTTPTarget1.output.moid}}) with ID {{.global.ClaimHTTPTarget1.output.moid}} has been claimed
```

As soon as we claim a target, we will have our BOT sending a message like this. 'httpbin' is the name of the target, but that is actually an hyperlink that will bring you to the Intersight target just claimed. For further info on markdown syntax: https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf



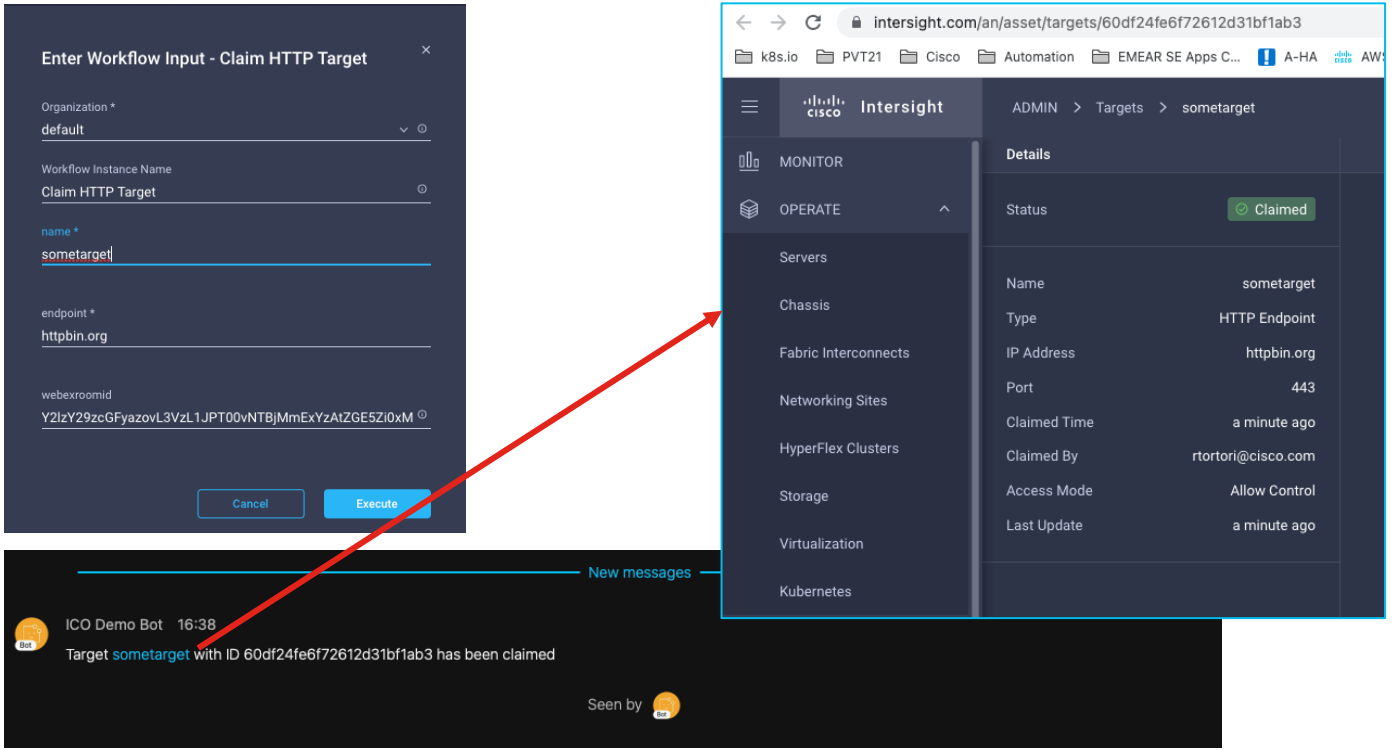
Use the Notification Task



roomid will be a **Direct Mapping** to the 'webxroomid' workflow input
External Target a **Static Value** pointing to the Webex endpoint we claimed at the beginning of the process.

Save and **Execute** the workflow

Run the Workflow



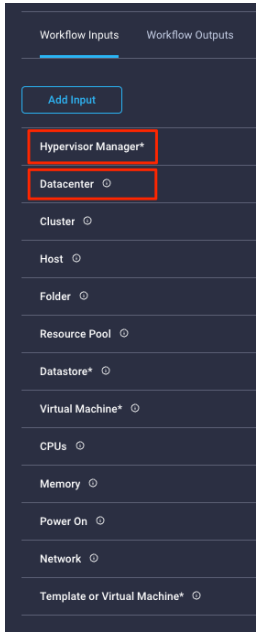
Parameter Set – Use Cases

Condition-based workflow input display

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:

I want to show VMware *Datacenter* workflow input **after** the *Hypervisor Manager* has been selected and **if** the selected manager is in state *Connected*



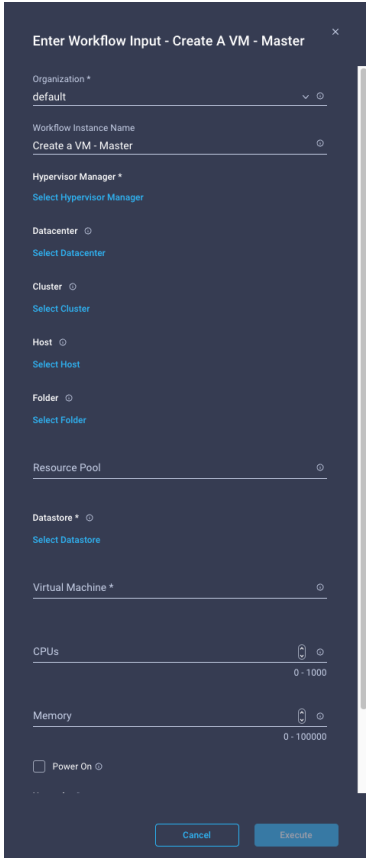
Default behavior:

All workflow inputs displayed at once unconditionally

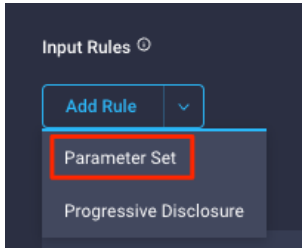
Drawbacks:

1) User may execute a workflow against a disconneted target (which will eventually fail)

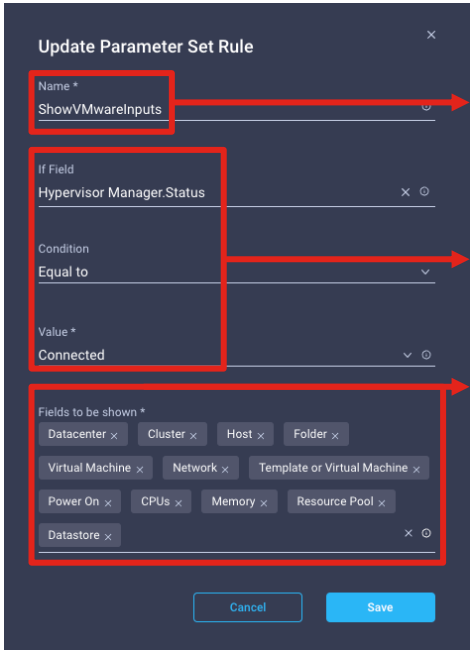
2) Unstructured inputs, clutter, missing a meaningful flow



Parameter Set – Implementation Example



Under **General** tab of the workflow in workflow designer, click on **Input Rules, Parameter Set** to add a new parameter set rule



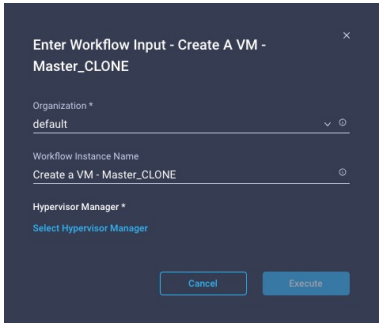
Arbitrary name

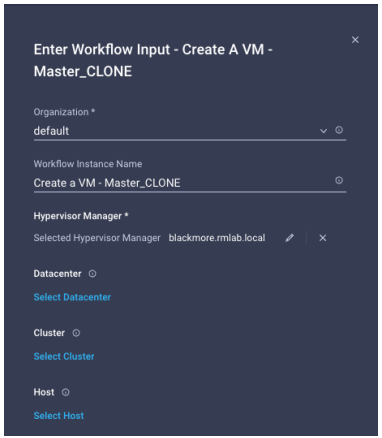
Only **Hypervisor Manager** is shown

Condition

Fields to show if condition is met

All other inputs will be disclosed after the condition is met (user has selected a connected hypervisor manager)





Parameter Set – Key Concepts

Documentation:
https://intersight.com/help/saas/resources/Workflow_Designer#workflow_input_parameter_set_and_progressive_disclosure_rules

The supported data types for Parameter Set rules are:

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

If an input is not shown because of a parameter set rule, it will have NO value even if a default has been configured

Example:

Workflow Input CPU has a default value of 4
- A parameter set rule prevents CPU to be shown to the user. CPU value will be *null*
- A parameter set rule shows CPU as the condition is met. CPU value will be *4*

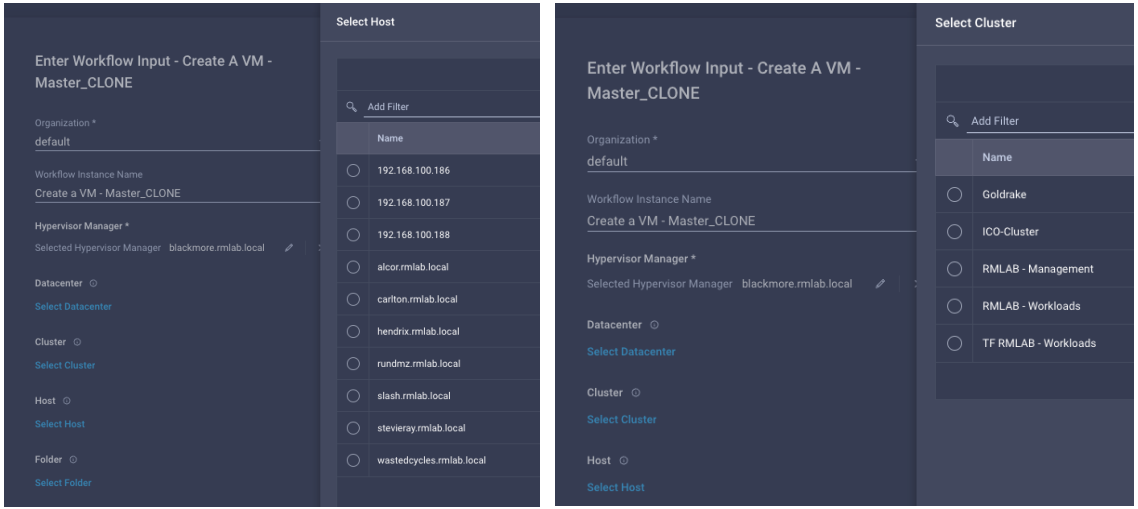
Progressive Disclosure – Use Cases

Filters data available in an input field based on another input selection

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:

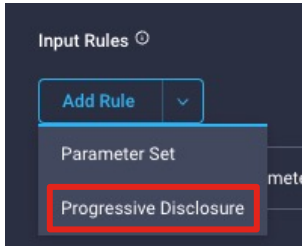
If I select VMware *Cluster* 'Alpha' in the *Cluster* input, I want users to select in the *Hosts* input only hosts belonging to that cluster



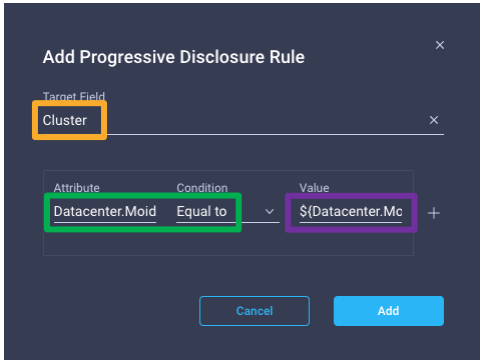
Default behavior:

User can potentially select incompatible data, like hosts not belonging to selected clusters

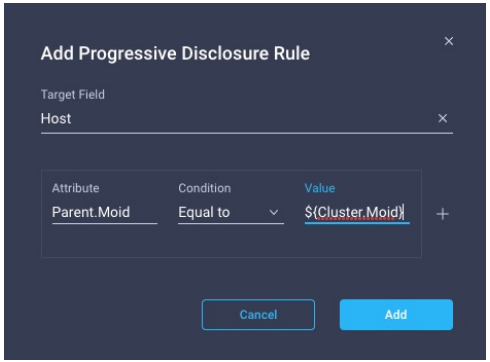
Progressive Disclosure– Implementation Example



Under **General** tab of the workflow in workflow designer, click on **Input Rules, Progressive Disclosure** to add a new rule



We add a second rule for the **Hosts**, in order to only show entries belonging to the selected **Cluster**



From apidocs, you can query the target resource, in the first case 'VmwareClusters', and look at the JSONPATH of the **Attribute** and **Value** you need for your rule.
In this example we want the **Cluster** input, to only show entries that have **Datacenter.Moid Equal to** the **Moid** value of the **Datacenter** workflow input (Reference Name)

Excerpt taken from:
https://intersight.com/apidocs/apirefs/api/v1/virtualization/VmwareClusters/get/

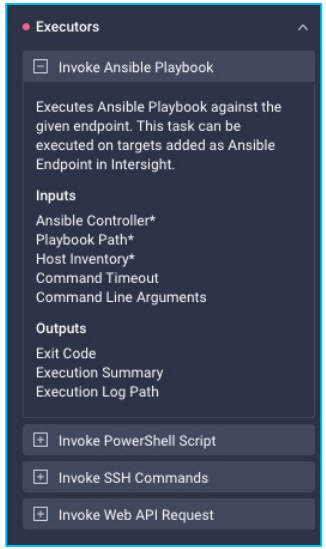
Progressive Disclosure– Key Concepts

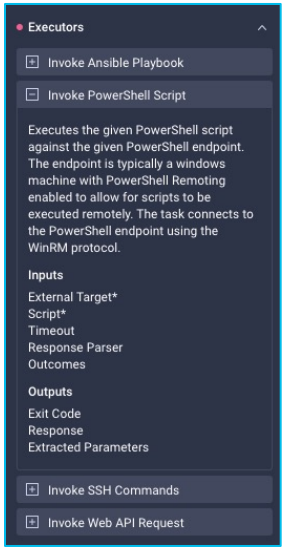
Documentation:
https://intersight.com/help/saas/resources/Workflow_Designer#workflow_input_parameter_set_and_progressive_disclosure_rules

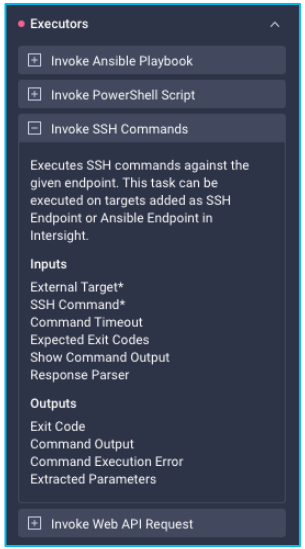
The supported data types for the Progressive Disclosure rules are:
- MoReference
- Target
- String Object Selector

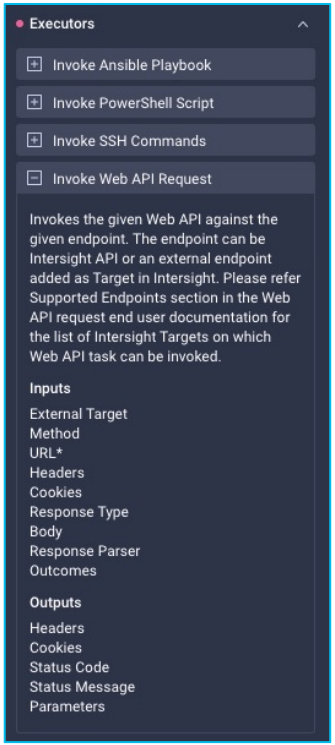
Executors

Executors









Executors are tasks that allow for custom actions in Intersight Cloud Orchestrator (ICO)

They can be used as **Embedded Executors** in the **Workflow Designer** to invoke one-off executions (scoped in the current workflow) or as **Reusable Tasks** in the **Task Designer** to create custom tasks that can be reused across multiple workflows.

When used in the Task Designer you have the option to also create outputs that will extract values from a response parser, the user experience will be similar to what you would have using native tasks.

Executors - Powershell

Powershell Executor

Scope:
Execute Powershell Script in a target Powershell endpoint as embedded task or a custom task

Requirements:
- The endpoint is reachable through Intersight Assist
- The endpoint target is in the Connected state
- PowerShell Remoting is enabled on the target endpoint to allow the endpoint to receive PowerShell remote commands
- As of March 2022, we only support WinRM and Windows Powershell Endpoint. SSH remoting is not supported

Documentation:
https://intersight.com/help/saas/resources/Executor_PowerShell#using_the_executor_powershell_script_embedded_task

Claim a Powershell Endpoint

Endpoint Configuration – Create an SSL certificate and WinRM HTTPS Listener

On the target endpoint, open a “Command Prompt” with Administrator privileges.

Create a new self-signed certificate and take note of the result **Thumbprint**.
Replace the placeholder below with your host DNS name or host IP:

```
New-SelfSignedCertificate -DnsName <IP or DNS name> -CertStoreLocation Cert:\LocalMachine\My
```

Create a WinRM HTTPS listener. Fill in with your IP or DNS name and the certificate **Thumbprint**:

```
winrm create winrm/config/Listener?Address=*+Transport=HTTPS @{Hostname=”<IP or DNS name>";
CertificateThumbprint=”<Thumbprint>"}
```

```
C:\Users\Administrator>winrm create winrm/config/Listener?Address=*+Transport=HTTPS @{Hostname="192.168.130.61"; Certifi
cateThumbprint="989E42FD13A5E0DF0F33E08D919321CE5104C283"}
ResourceCreated
    Address = http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    ReferenceParameters
        ResourceURI = http://schemas.microsoft.com/wbem/wsman/1/config/listener
        SelectorSet
            Selector: Address = *, Transport = HTTPS
```

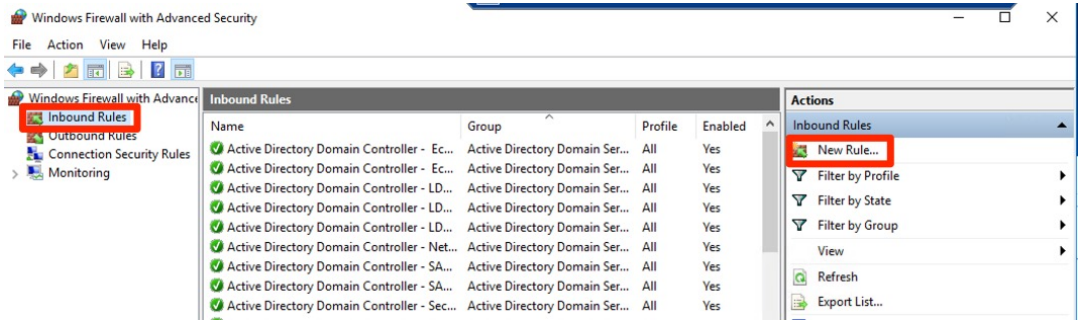
Force Powershell to use TLS 1.2 for web requests

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
```

Claim a Powershell Endpoint

Endpoint Configuration – Create a firewall rule to allow TCP port 5986

On Windows Firewall, select **Inbound Rules** and create a **New Rule**



Rule Type: Port
Protocol and Ports: TCP, Specific local ports: 5986
Action: Allow the connection
Profile: select all that apply
Name: Allow HTTPS Powershell

Claim a Powershell Endpoint

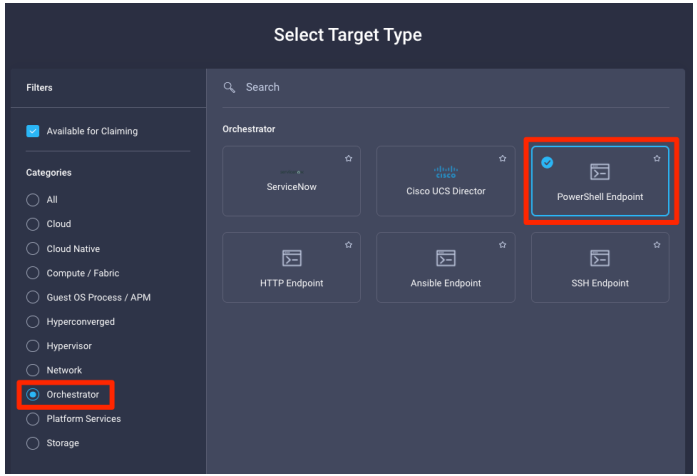
Endpoint Configuration – Create a firewall rule to allow TCP port 5986

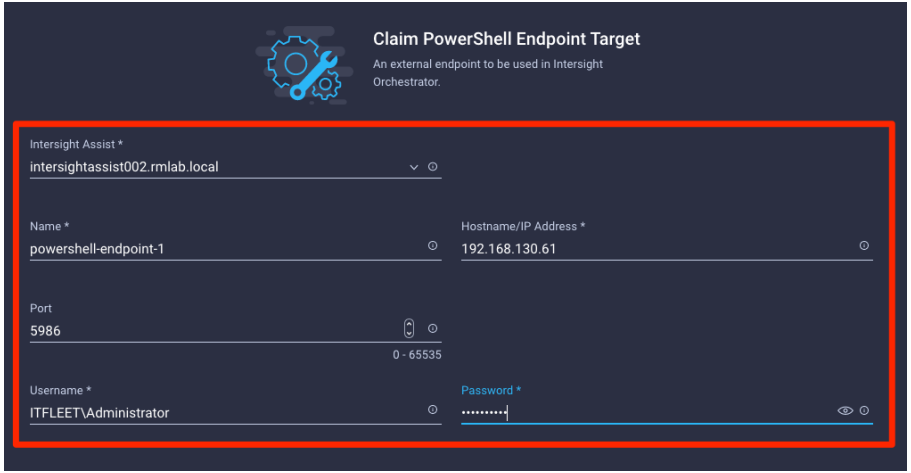
Create a powershell script (example: intersight.ps1), taking the content from:
https://intersight.com/help/saas/resources/Executor_PowerShell

This script will check everything is right and will create the required configuration if something is missing.
Execute Powershell script in the endpoint, using the **Powershell** command line

```
PS C:\Users\Administrator> .\intersight.ps1
Basic authentication is already disabled
Firewall rule already exists for WinRM
PS Remoting has been successfully configured for Intersight.
PS C:\Users\Administrator>
```

Claim a Powershell Endpoint
Intersight Configuration – Claim the target



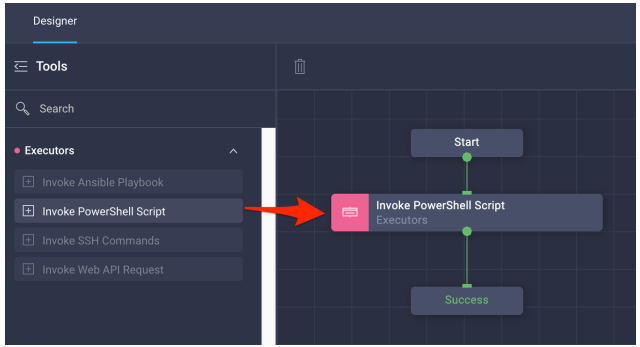


Select one **Intersight Assist** (must be able to reach the target endpoint), specify a **name**, the target **IP** or **Hostname** as well as the credentials.

Hit **Claim.**

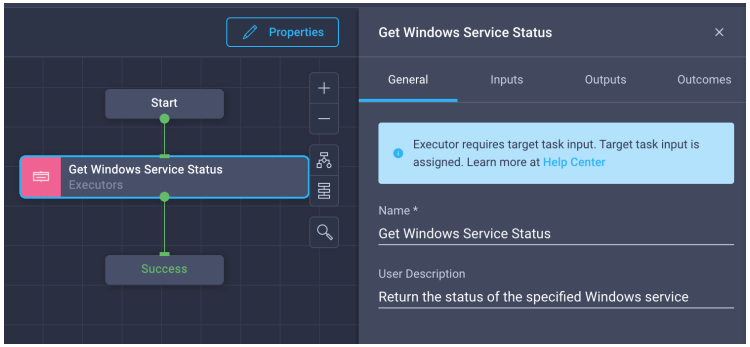
Create a Powershell Custom Task 1/6

Sample Scenario: Get the status of the specified Windows Service



Drag the **Invoke Powershell Script** executor in the designer.

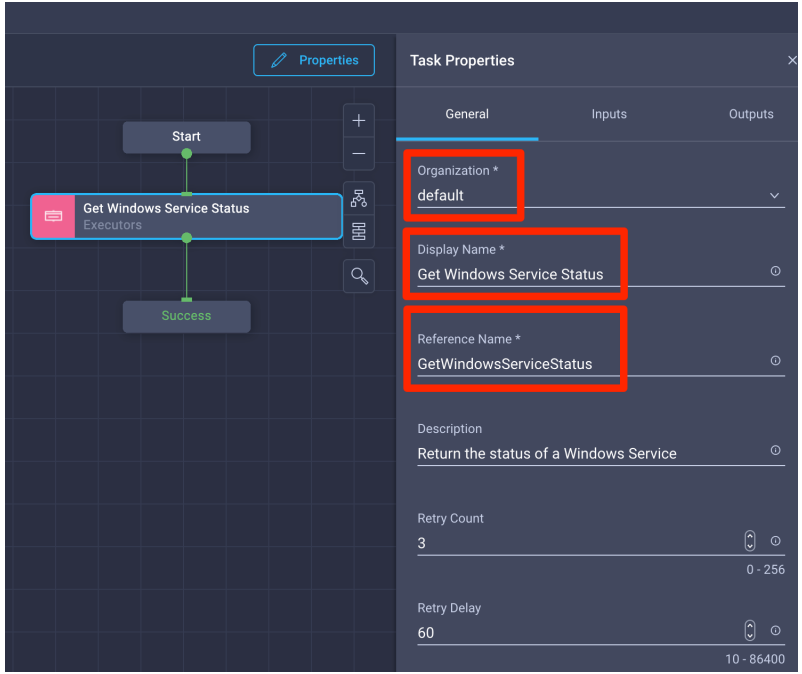
Note: you can't use more than one executor type in a single custom task. However, you can use multiple executors of the same type.



Optionally, assign a name and a description

Create a Powershell Custom Task 2/6

Sample Scenario: Get the status of the specified Windows Service

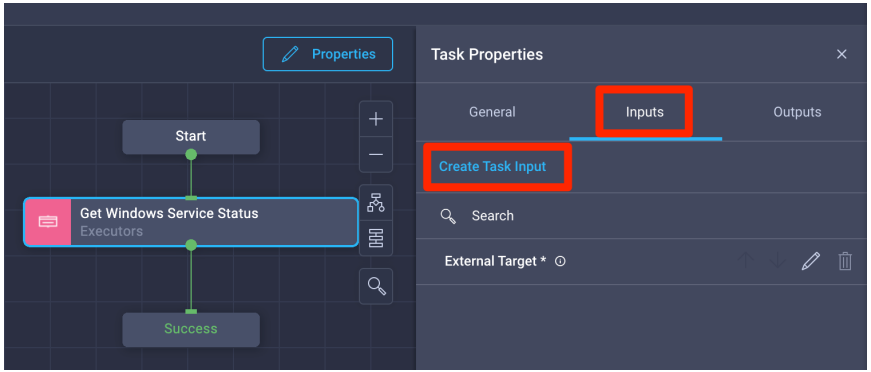


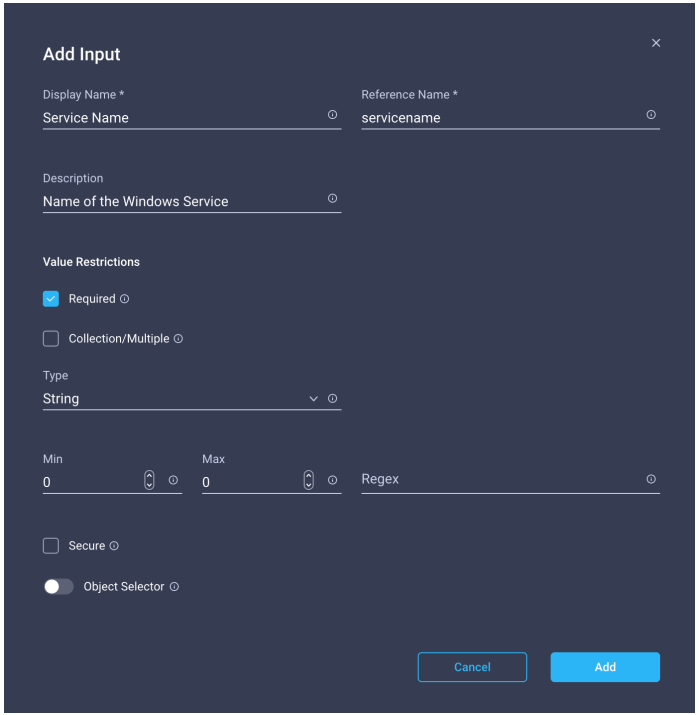
Click **Properties** to access the **General** custom task properties.

Select an **Organization**, assign a **Display Name**, a **Reference Name** and a **Description** (optional)

Create a Powershell Custom Task 3/6

Sample Scenario: Get the status of the specified Windows Service





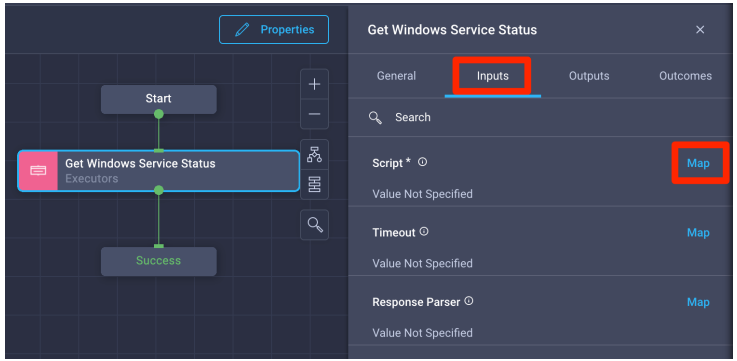
Create a **Task Input.**
This will be a required input for the custom task representing the name of the service you want to get the status for.

The **Reference Name** will be used when you reference this input in the Powershell script

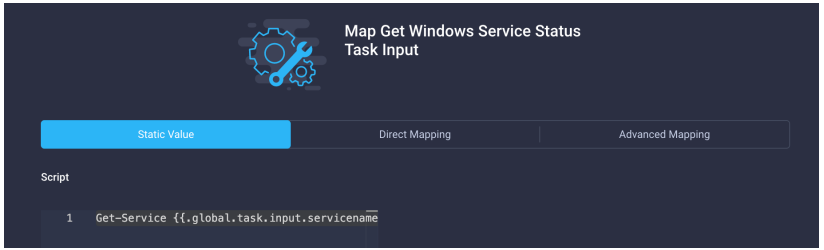
Reference Name: servicename
Type: String

Create a Powershell Custom Task 4/6

Sample Scenario: Get the status of the specified Windows Service



Map a Powershell **Script** under the executor **Inputs**



Write or copy a Powershell Script. In this example, it's just a simple command that requests information on a given Windows Service.

```
Get-Service {{.global.task.input.servicename}}
```

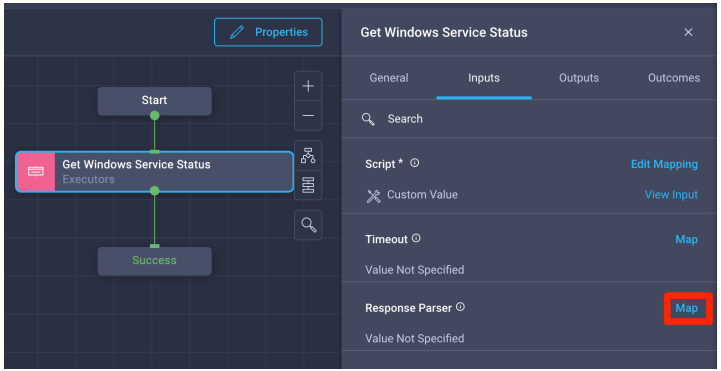
We are using the templating syntax to pass the **Service Name** input (Reference name: **servicename**) as a parameter of the `Get-Service` command

You can also use Direct Mapping to a Workflow Input or a Task Output if necessary. Advanced Mapping may also be used if you want to use gotemplates to apply transformations, etc.

Hit **Map** to confirm.

Create a Powershell Custom Task 5/6

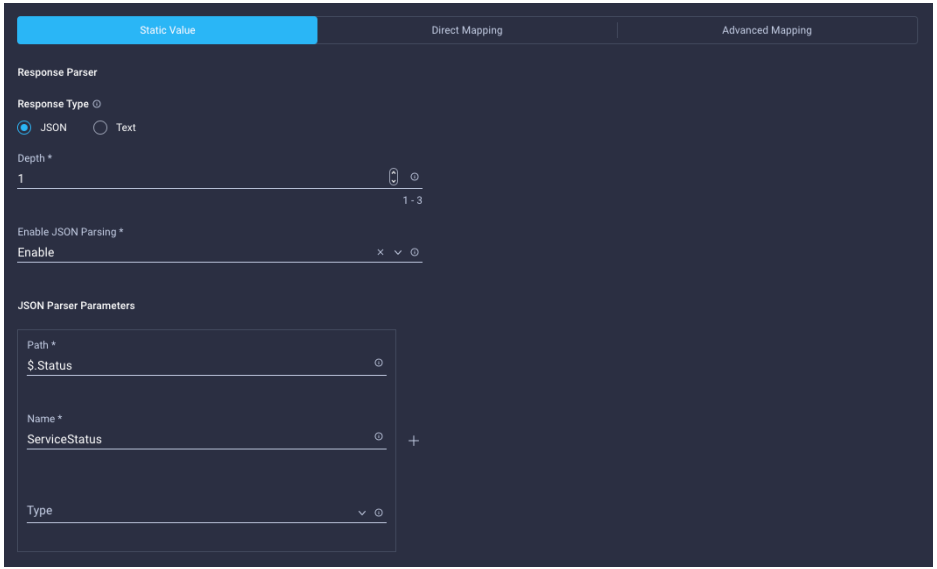
Sample Scenario: Get the status of the specified Windows Service



Map a **Response Parser**, it will be used to extract data from the server response.

In this example, we will extract the value of key **“Status”**, we give it a name **ServiceStatus** using **JSON Parsing** and create a task **Output** with this value.
To do this we use the jsonpath syntax $.Status

For additional info on jsonpath syntax:
https://lzone.de/cheat-sheet/JSONPath



Note: When you specify the PowerShell script statement and enable the Response Parser Type as JSON, at the time of execution of the workflow, Intersight pipes `ConvertTo-Json` *to the script to get the corresponding responses as JSON output. This means that you should* ***not*** *explicitely convert your script output to JSON.*

Create a Powershell Custom Task 6/6

Sample Scenario: Get the status of the specified Windows Service

```
PS C:\Users\Administrator> Get-Service W32Time | ConvertTo-Json
{
    "CanPauseAndContinue":  false,
    "CanShutdown":  true,
    "CanStop":  true,
    "DisplayName":  "Windows Time",
    "DependentServices":  [

                          ],
    "MachineName":  ".",
    "ServiceName":  "W32Time",
    "ServicesDependedOn":  [

                           ],
    "ServiceHandle":  {
                          "IsInvalid":  false,
                          "IsClosed":  false
                      },
    "Status":  4,
    "ServiceType":  32,
    "StartType":  2,
    "Site":  null,
    "Container":  null,
    "Name":  "W32Time",
    "RequiredServices":  [

                         ]
}
```

Sample Response from a Windows Server 2012

Fields

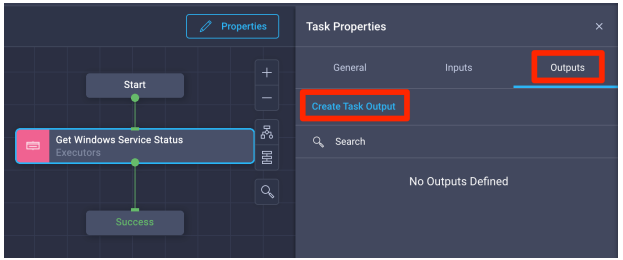
ContinuePending	5	The service continue is pending. This corresponds to the Win32 `SERVICE_CONTINUE_PENDING` constant, which is defined as 0x00000005.
Paused	7	The service is paused. This corresponds to the Win32 `SERVICE_PAUSED` constant, which is defined as 0x00000007.
PausePending	6	The service pause is pending. This corresponds to the Win32 `SERVICE_PAUSE_PENDING` constant, which is defined as 0x00000006.
Running	4	The service is running. This corresponds to the Win32 `SERVICE_RUNNING` constant, which is defined as 0x00000004.
StartPending	2	The service is starting. This corresponds to the Win32 `SERVICE_START_PENDING` constant, which is defined as 0x00000002.
Stopped	1	The service is not running. This corresponds to the Win32 `SERVICE_STOPPED` constant, which is defined as 0x00000001.
StopPending	3	The service is stopping. This corresponds to the Win32 `SERVICE_STOP_PENDING` constant, which is defined as 0x00000003.

Status code mappings of the ServiceControllerStatus. Reference: https://docs.microsoft.com/en-us/dotnet/api/system.serviceprocess.servicecontrollerstatus?view=dotnet-plat-ext-6.0

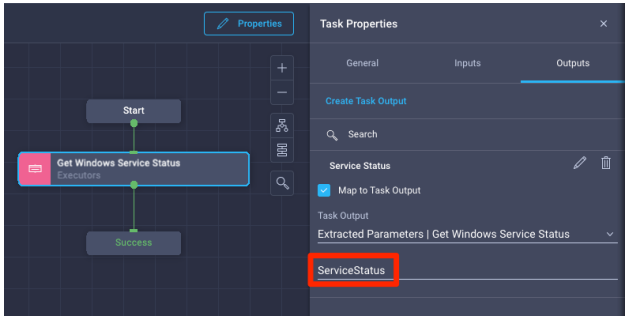
Create a Powershell Custom Task 7a/7

Sample Scenario: Get the status of the specified Windows Service

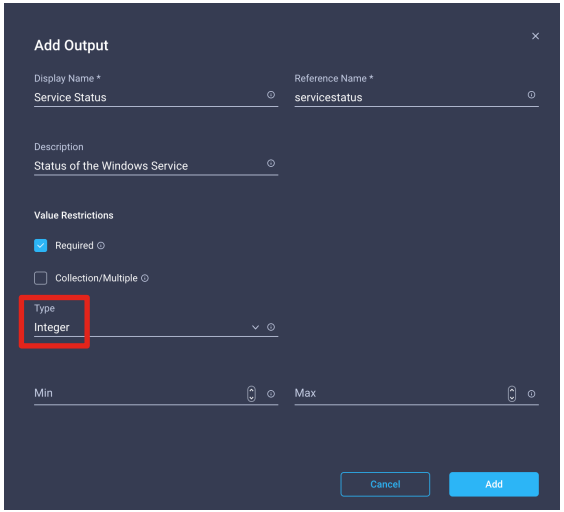
Approach 1: Map the response parser extracted parameter to the task output



Create a **Task Output** from the **Output** tab of the general custom task **Properties**



Map to Task Output specifying the **Extracted Parameters path** of the **Get Windows Service Status** task (in this case **ServiceStatus**)

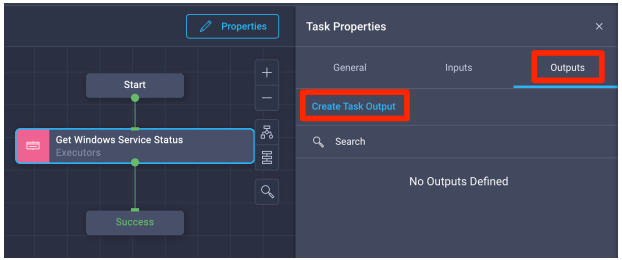


Note: As of February 14 2022, this approach won't work due to defect ***CSCwa37847***

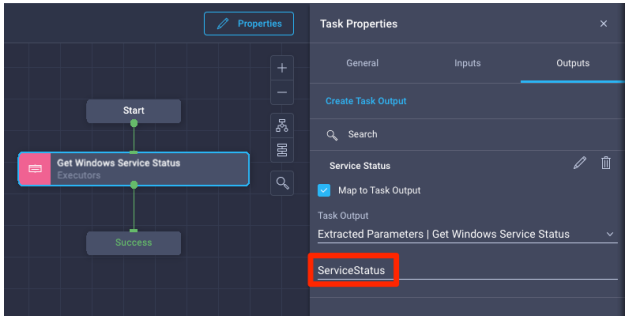
Create a Powershell Custom Task 7b/6

Sample Scenario: Get the status of the specified Windows Service

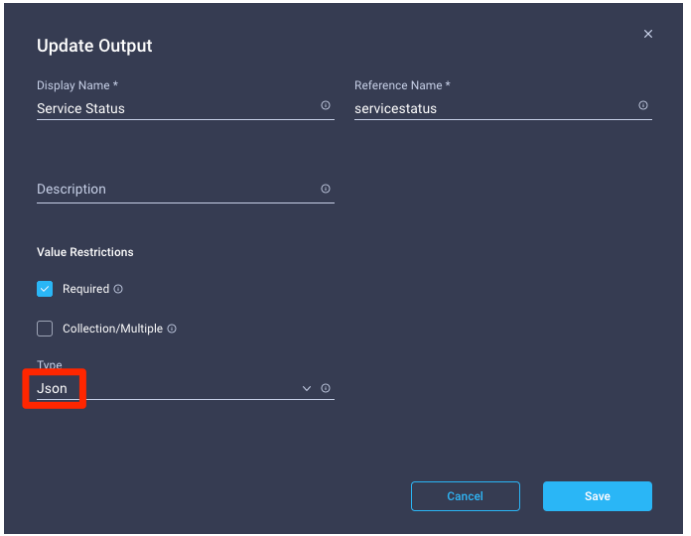
Approach 2: Map the full response to the task output (Response parser will be ignor



Create a **Task Output** from the **Output** tab of the general custom task **Properties**

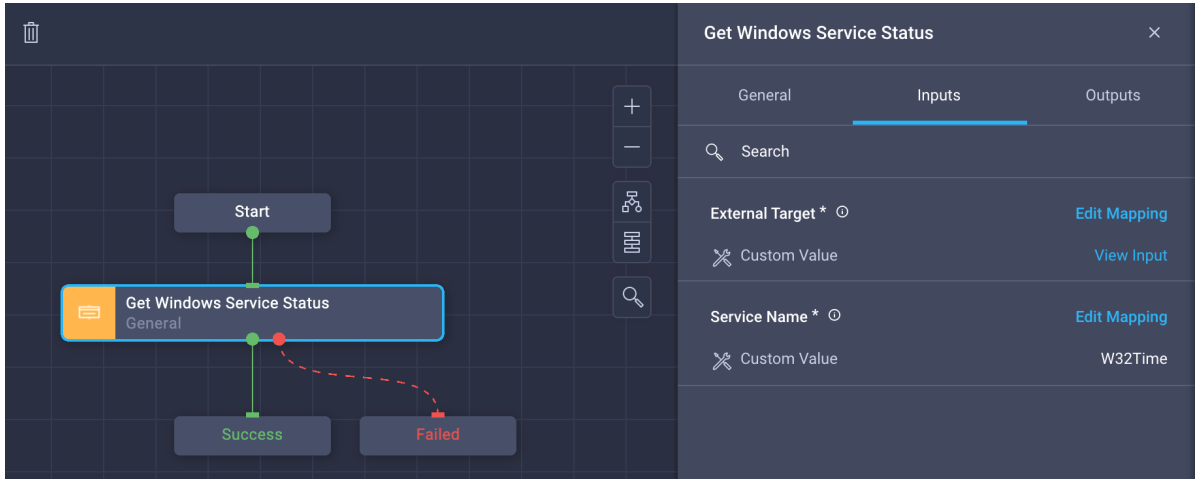


Map to Task Output specifying the **Response** of the **Get Windows Service Status** task



Execute a Workflow with the Powershell Custom Task

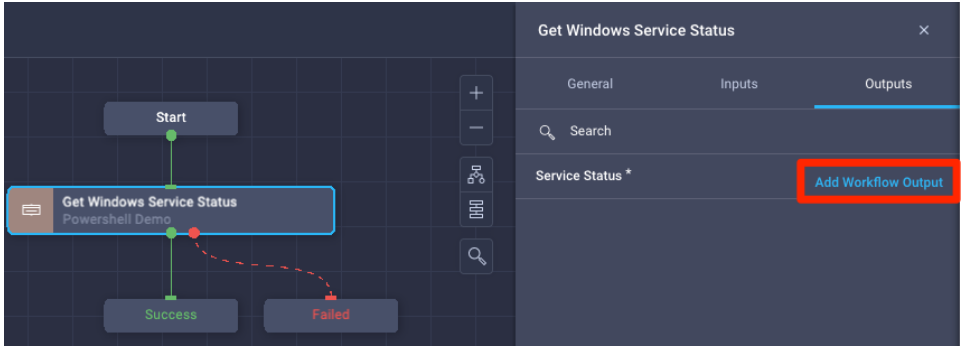
Sample Scenario: Get the status of the specified Windows Service



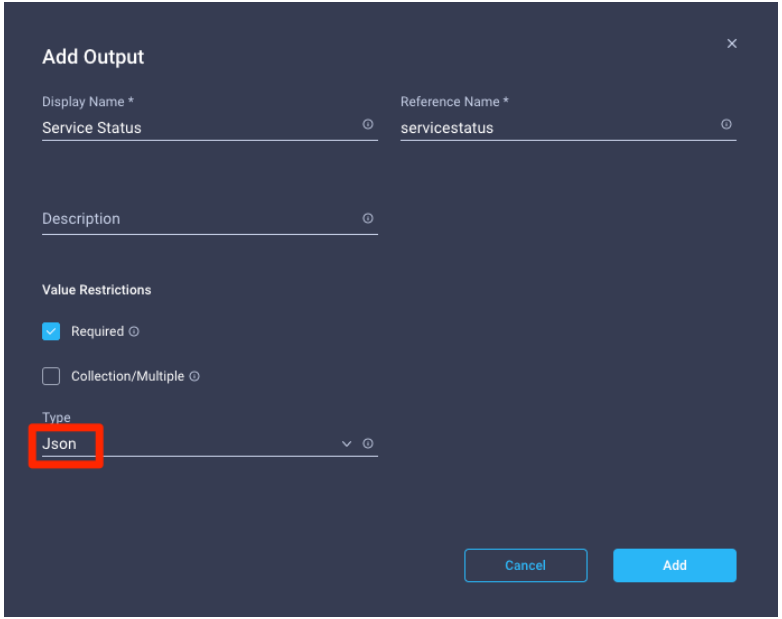
Drag the new **Get Windows Service Status** custom task from the task library.
Map the **External Target** to your Powershell Endpoint.
Map a **Service Name** to a Workflow Input or set it statically for testing purposes.
An example Windows Service for testing may be the **W32Time** service (Windows Time)

Execute a Workflow with the Powershell Custom Task

Sample Scenario: Get the status of the specified Windows Service

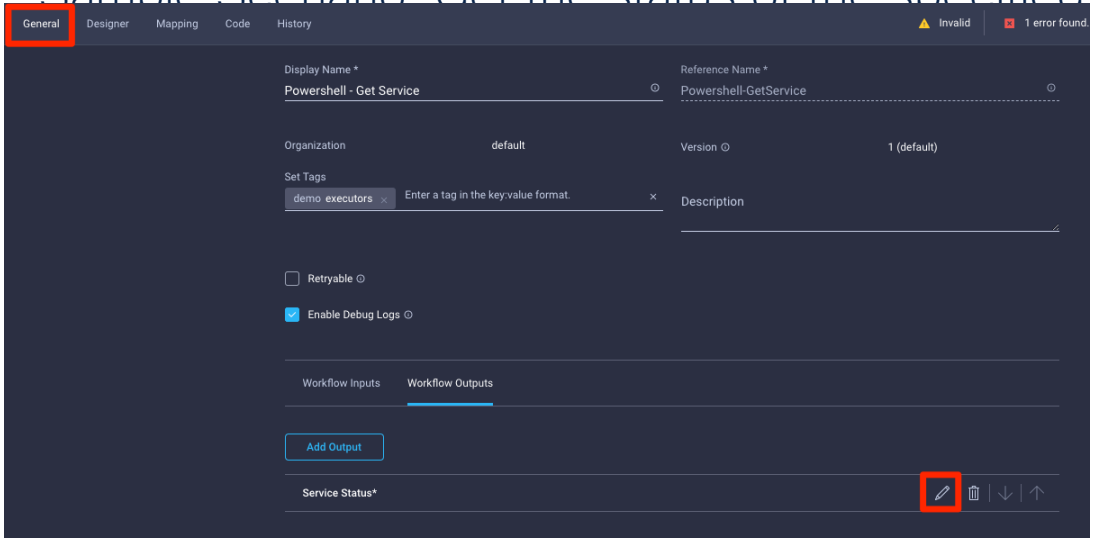


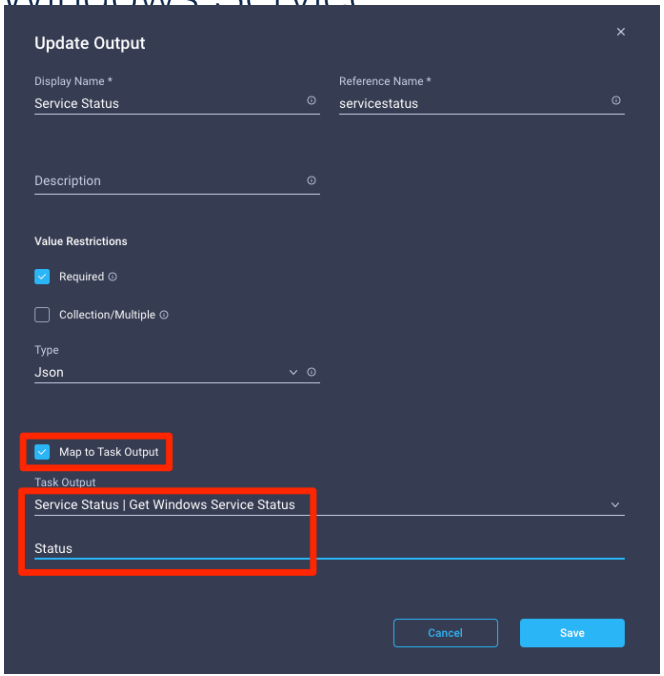
On the task output, select **Add Workflow Output** to create a workflow output.
The new Workflow is going to have automatically the same data type of the task output, click **Add**



Execute a Workflow with the Powershell Custom Task

Sample Scenario: Get the status of the specified Windows Service



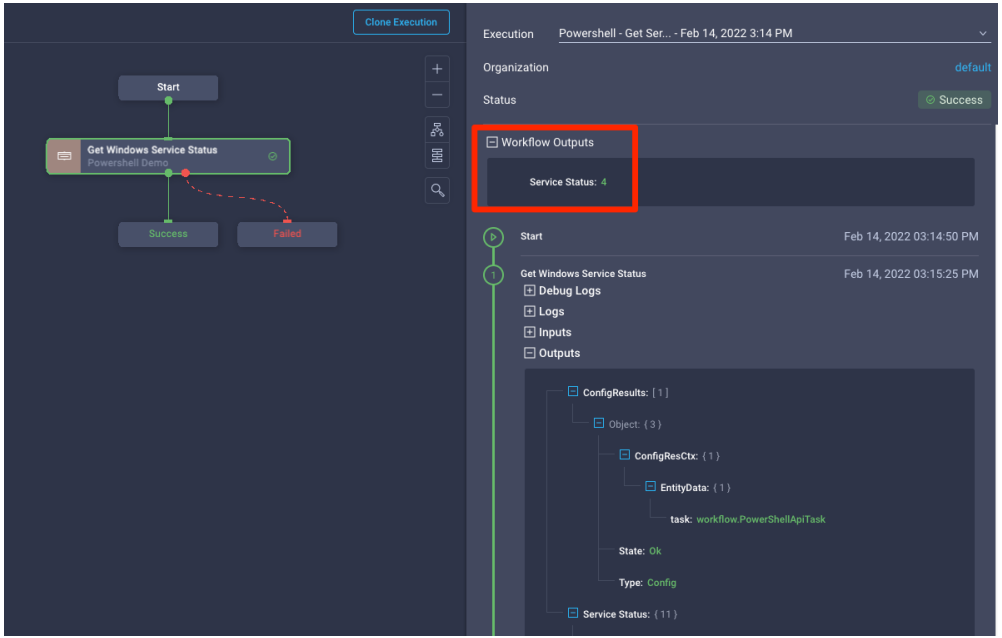


In the **General** tab, edit the Workflow Output **Service Status** and **Map to Task Output** the **Status** path of the **Service Status** output of **Get Windows Service Status** task. Save and execute the workflow

Execute a Workflow with the Powershell Custom Task

Sample Scenario: Get the status of the specified Windows Service

Verification



Executors - SSH

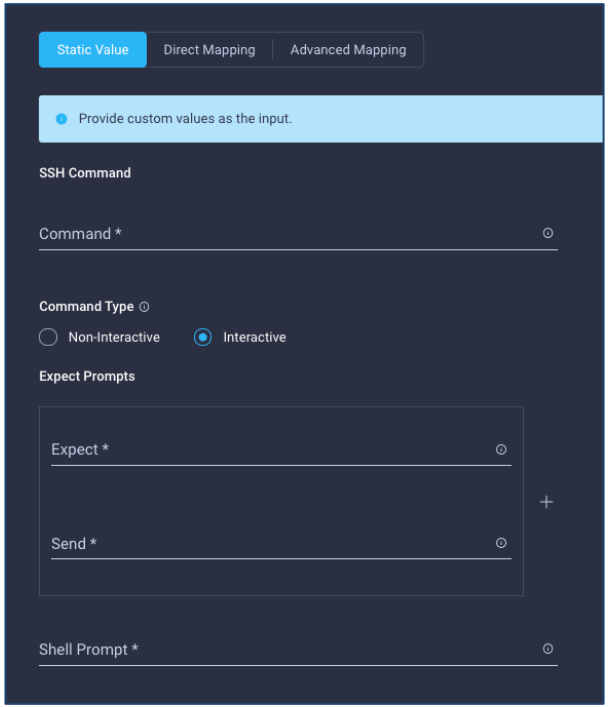
SSH Executor

Scope:
Execute CLI commands over SSH in a target endpoint as embedded task or a custom task

Requirements:
- The endpoint is reachable through Intersight Assist
- The endpoint target is in the Connected state
- SSH server is enabled and running on the target endpoint
- Username/Password or private key for authentication

Documentation:
https://www.intersight.com/help/saas/resources/Executor_SSH

Supported CLI Command Types



Command Types:

Non-Interactive

Commands that exit as soon as it's executed

Example: sh /path/to/bin -A -B

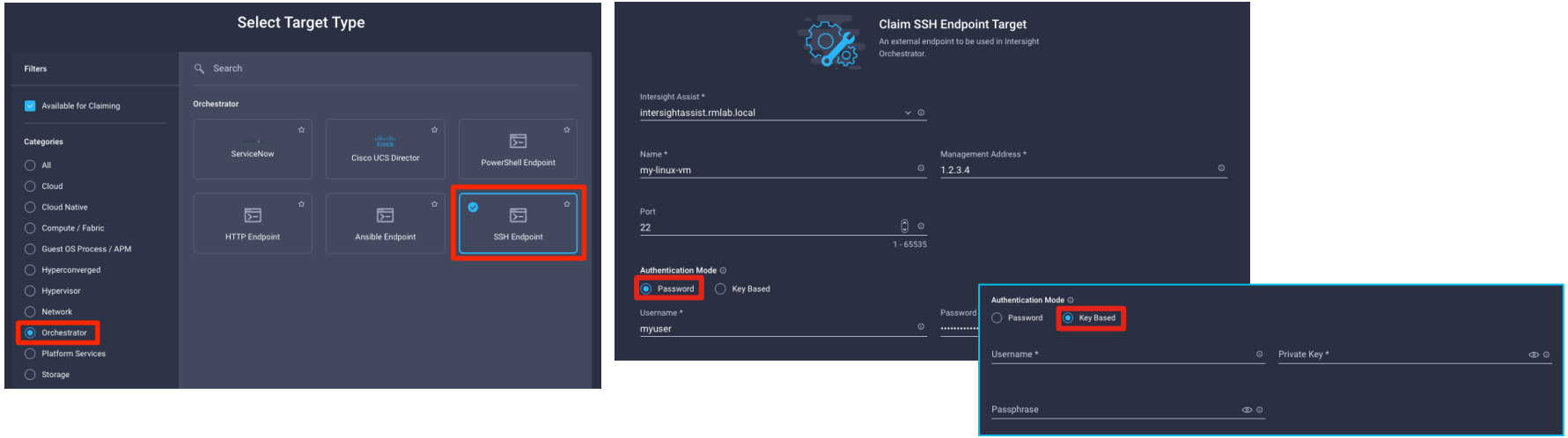
Interactive

Commands that requires user interaction (i.e. asking questions)

Example: sudo yum install -> y

Claim a SSH Endpoint

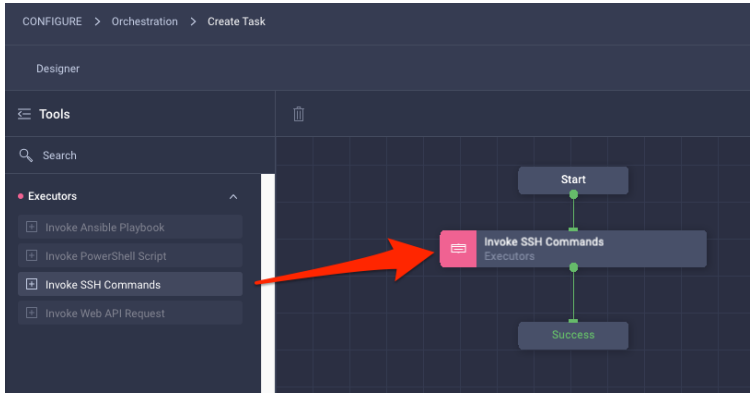
Intersight Configuration – Claim the target



Select one **Intersight Assist** (must be able to reach the target endpoint), specify a **name**, the target **IP** or **Hostname** as well as the credentials. For SSH, you can select between **Password** authentication mode and **Key Based**
Hit **Claim.**

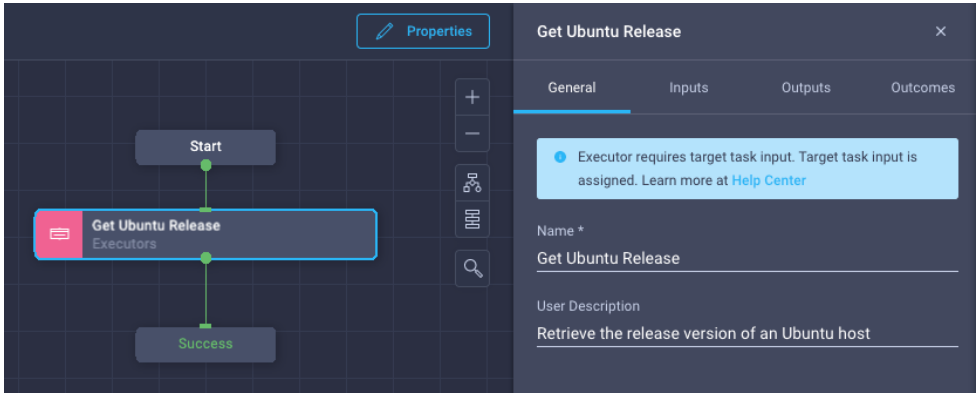
Create a SSH Custom Task - Non-Interactive 1/5

Sample Scenario: Get the release number of an Ubuntu Linux Host



Drag the **Invoke SSH Command** executor in the designer.

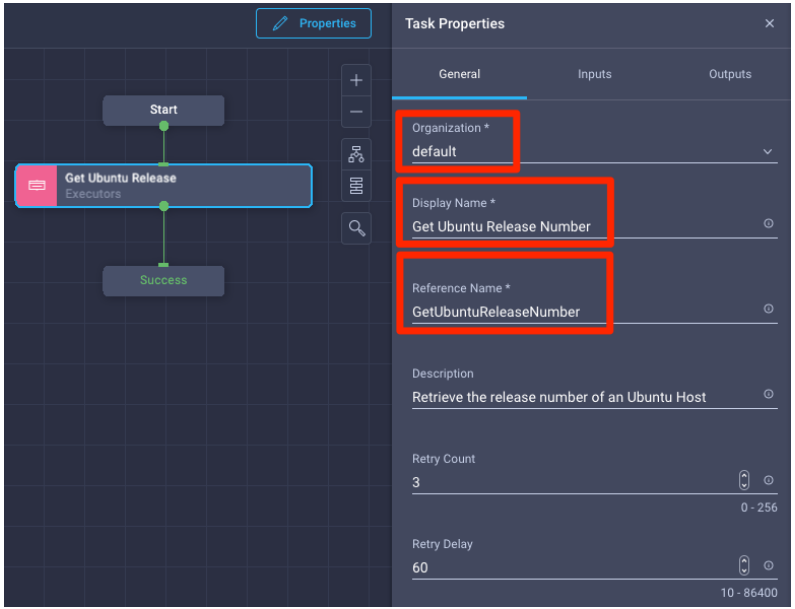
Note: you can't use more than one executor type in a single custom task. However, you can use multiple executors of the same type.



Optionally, assign a name and a description

Create a SSH Custom Task - Non-Interactive 2/5

Sample Scenario: Get the release number of an Ubuntu Linux Host

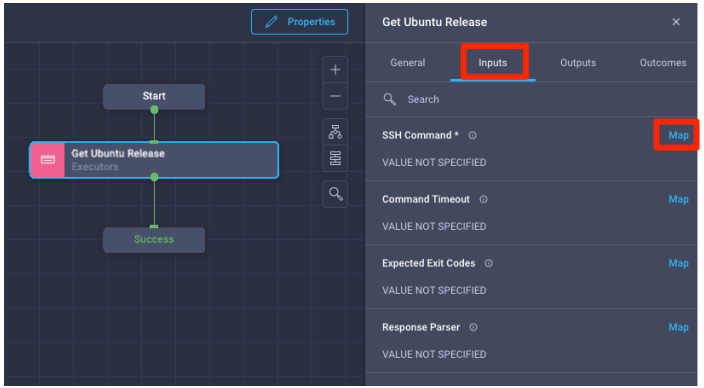


Click **Properties** to access the **General** custom task properties.

Select an **Organization**, assign a **Display Name**, a **Reference Name** and a **Description** (optional)

Create a SSH Custom Task - Non-Interactive 3/5

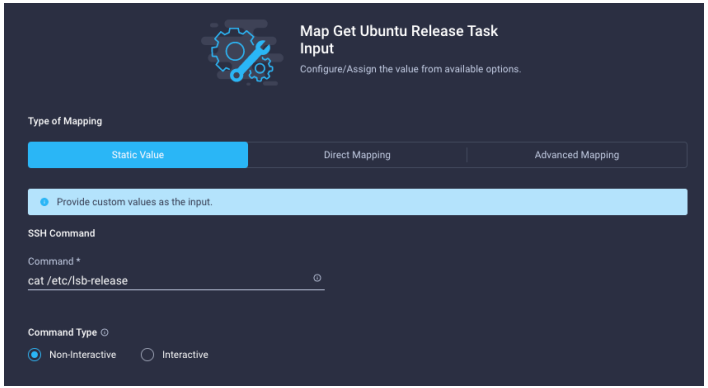
Sample Scenario: Get the release number of an Ubuntu Linux Host



As mentioned, we have two ways to execute SSH commands:

1. Non-interactive Script
2. Interactive Script

In this example, we will use a non-interactive command.
Map a **SSH Command** under the executor **Inputs**



The command `cat /etc/lsb-release` will return the details of the installed Ubuntu release. Here's an example of the output:

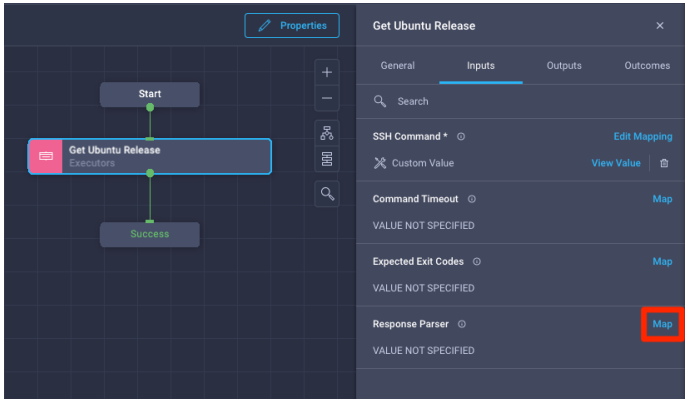
```
rtortori@rtortori-ubuntu-jh:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.6 LTS"
```

You can also use Direct Mapping to a Workflow Input or a Task Output if necessary. Advanced Mapping may also be used if you want to use gotemplates to apply transformations, etc.

Hit **Map** to confirm

Create a SSH Custom Task – Non-Interactive 4/5

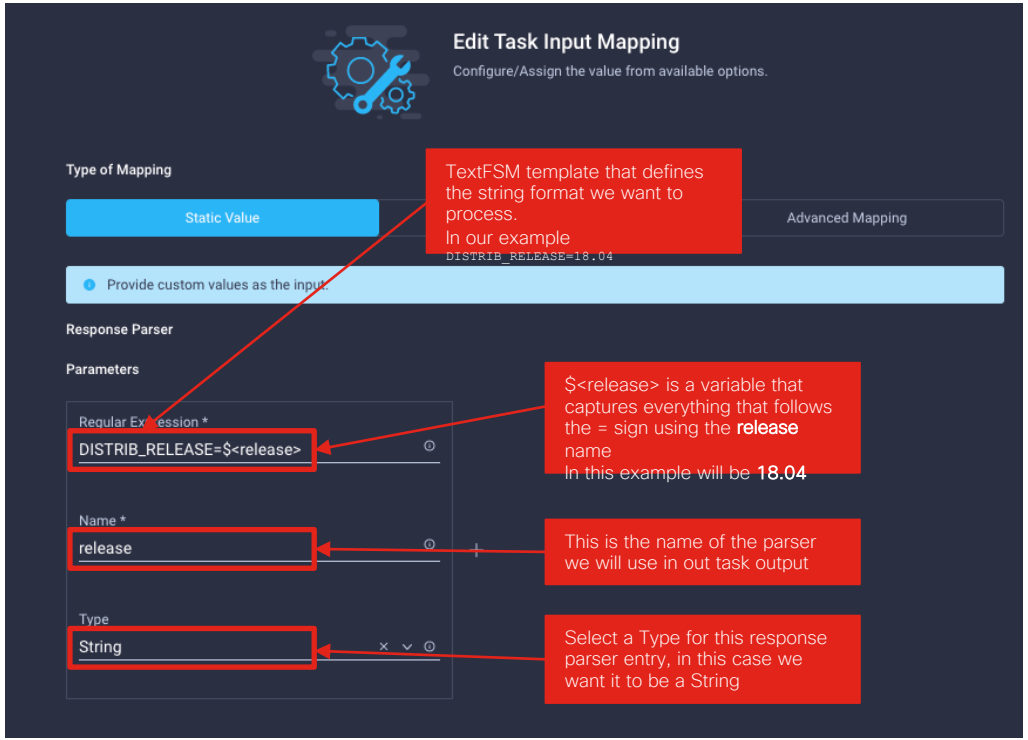
Sample Scenario: Get the release number of an Ubuntu Linux Host



Map a **Response Parser**, it will be used to extract data from the server response.

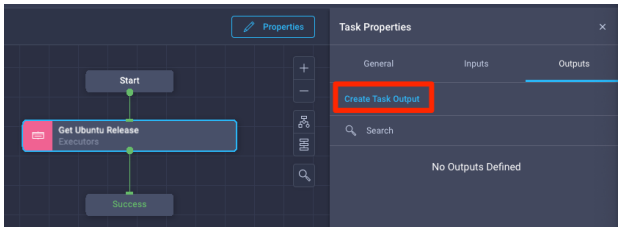
The SSH Executor uses the TextFSM which is a parser for semi-formatted text.

In this example, we will extract the **release** output from the output of the `cat /etc/lsb-release` command. Hit **Save** once done to save the task.

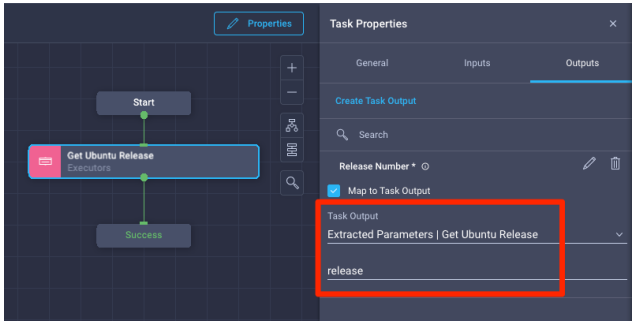


Create a SSH Custom Task – Non-Interactive 5/5

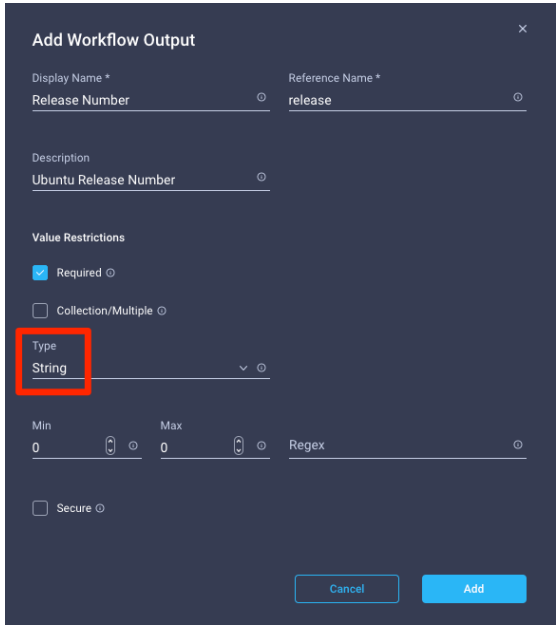
Sample Scenario: Get the release number of an Ubuntu Linux Host



Create a **Task Output** from the **Output** tab of the general custom task **Properties**

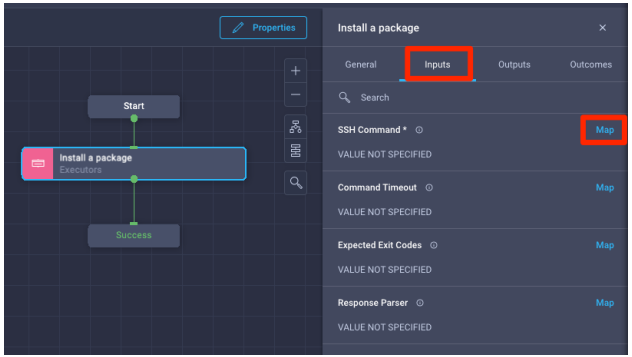


Map to Task Output specifying the **Extracted Parameters path** of the **Get Ubuntu Release** task (in this case **release**, the response parser entry we created earlier)



Create a SSH Custom Task – Interactive

Sample Scenario: Install a package using APT

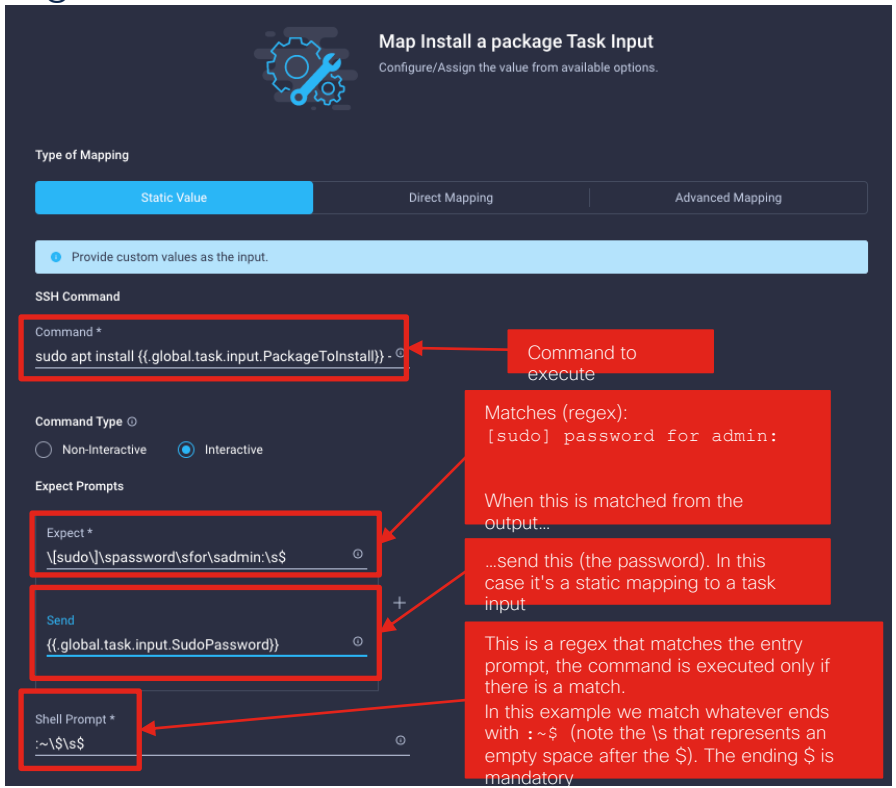


If you want to leverage an interactive command, **Map** a SSH **Command** under the executor **Inputs** and select **Interactive**

Our goal in this example will be to install a package which name will be passed as a workflow input (Ref name: **PackageToInstall**) using the command:

```
sudo apt install {{.global.task.input.PackageToInstall}} -y
```

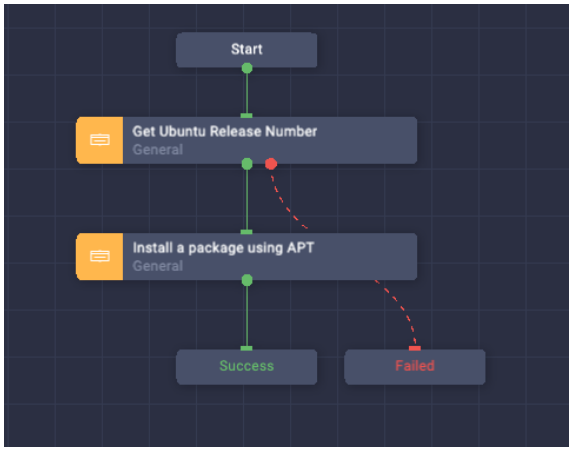
This is an interactive command as the system will ask for the sudo password as we are connected as the user **admin**. We need to intercept that request and send the password, then continue the execution



*Note: all regular expressions **must** end with the $ sign.*
The creation of task inputs is not covered for this example

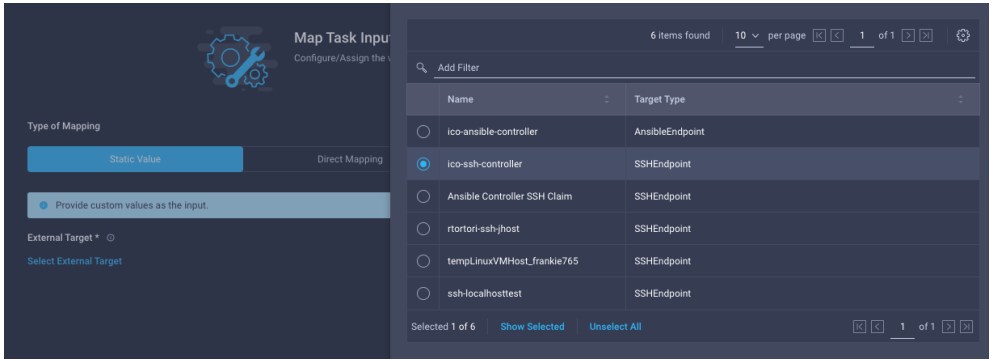
Execute a Workflow with the SSH Custom Task

Sample Scenario: Get release version and install a package



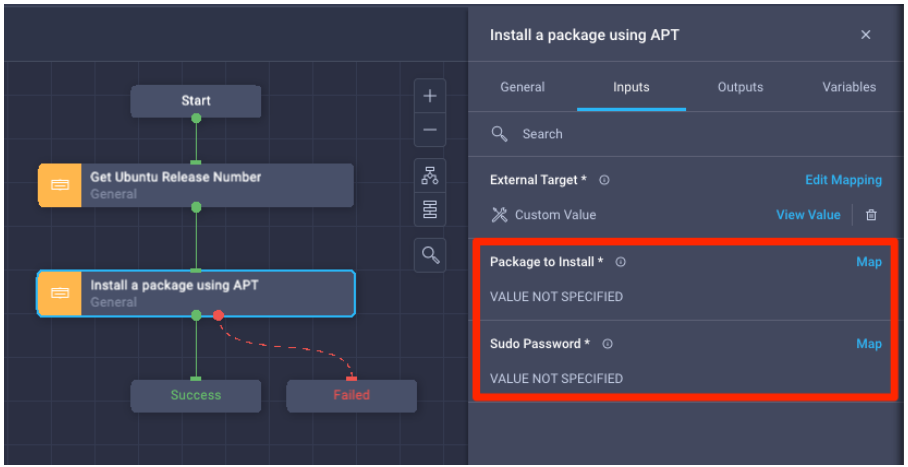
In this example we will run both SSH tasks (interactive and non-interactive). Drag both tasks from the task library

Map the **External Targets** to the SSH hosts we claimed.
All commands will be executed on this host



Execute a Workflow with the SSH Custom Task

Sample Scenario: Get release version and install a package

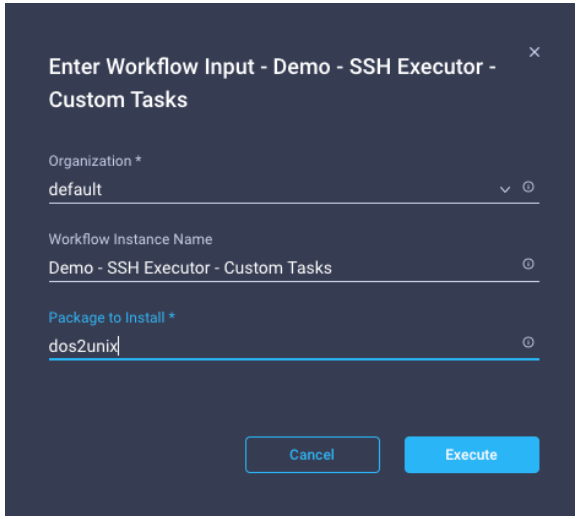


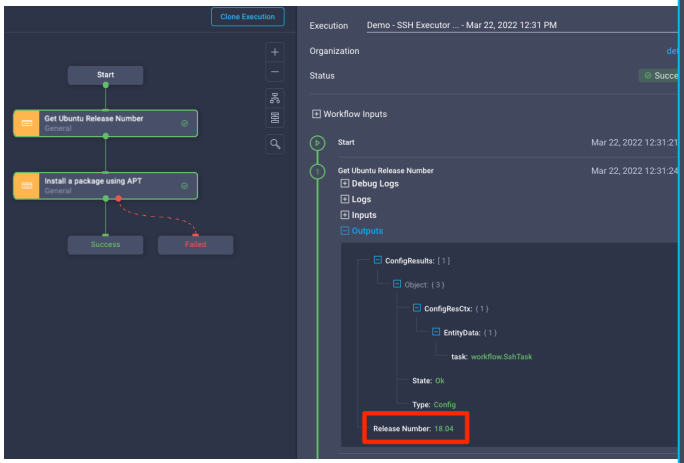
For the **Install a package using APT** task we configured two inputs, map them accordingly. Either statically or using a workflow input.

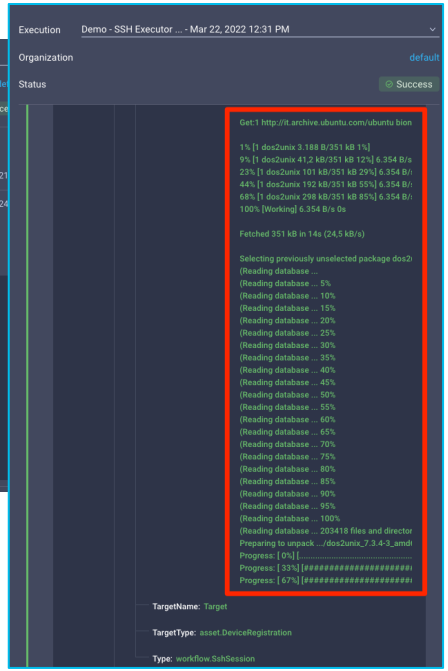
In this example the **Package to Install** input will be a direct mapping to a workflow input while the **Sudo Password** will be statically assigned

Execute a Workflow with the SSH Custom Task

Sample Scenario: Get release version and install a package







Execute the workflow and inspect the outputs (Note: **Enable Debug Logs** is turned on for this workflow)

Executors - Ansible

Ansible Executor

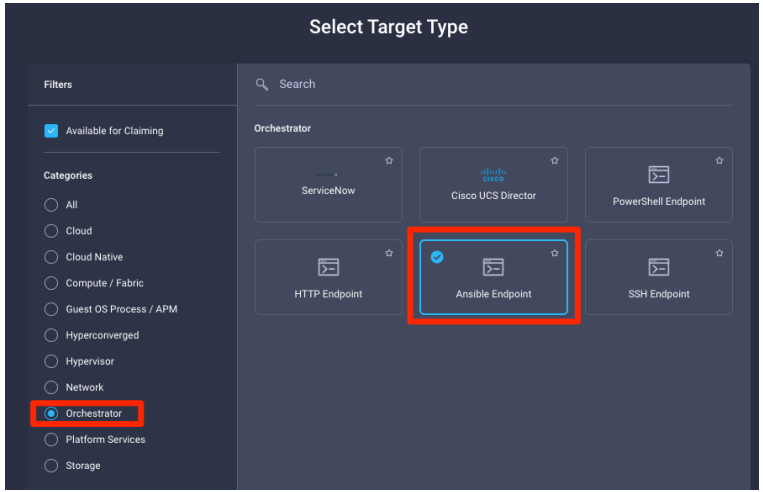
Scope:
Execute Ansible Playbooks in a target control node as embedded task or a custom task

Requirements:
- The Ansible Control node (Intersight Target) is reachable through Intersight Assist
- The Ansible Playbook modules and executables (i.e. `ansible-playbook`) required to run the playbooks are already installed on the Ansible Target
- The Ansible Control Node has password-less SSH access to the hosts (endpoints)
- The Ansible Control Node can resolve endpoints hostname if the Host Inventory file specified the hostnames

Documentation:
https://intersight.com/help/appliance/resources/Executor_Ansible

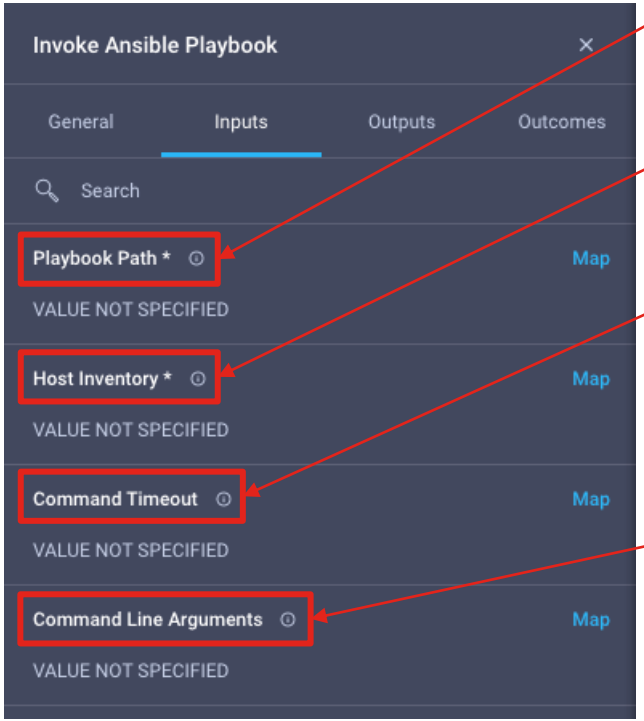
Claim an Ansible Target

Intersight Configuration – Claim the target



Select one **Intersight Assist** (must be able to reach the target endpoint), specify a **name**, the target **IP** or **Hostname** as well as the credentials. For Ansible target controller, you can select between **Password** authentication mode and **Key Based**
Hit **Claim.**

Anatomy of the Ansible Playbook Executor



The full path of the Ansible Playbook in the claimed Ansible Control Node
Example: `/home/ansible/deploy-nginx.yaml`

This can have two formats:
1. The full path of an existing inventory file on the host which will be used by the ansible playbook to retrieve the targets
2. Comma separated IPs or hostnames of the targets.
   Note: it is required to **always** have a comma at the end, even if there's only one entry.
   Example: root@1.2.3.4
admin@myswitch.company.com

Timeout of the playbook execution. Default is 600 seconds. If this timer expires the task will fail

The command line arguments for running the Ansible playbook against the given endpoint.
Note: The escape character backslash(\) needs to be used when the command line arguments contain double quotes(") in them.

The following command line options are not supported:

1. -vvv
2. -vvvv
3. -k, -K
4. -c
5. --connection
6. --sftp-extra-args
7. --scp-extra-args
8. --ask-vault-password
9. --step

For more information on the Ansible playbook documentation and the list of other supported command line options, see:

https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html

End to end Ansible Scenario
Sample Scenario: Deploy NGINX Webserver

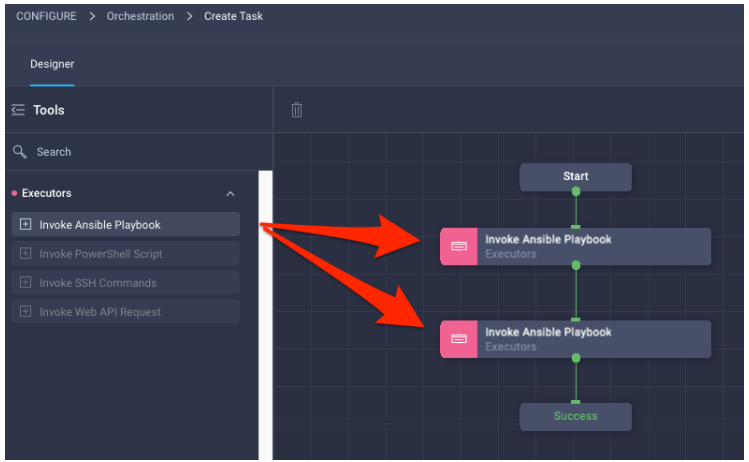
In this sample scenario, we will deploy an NGINX webserver in a target host.
The actual Ansible implementation consists in running two playbooks:

1. Install NGINX
2. Sync Site Content

We will create a custom task named `Install NGINX with Ansible` with two Ansible Playbook executors to invoke those two runs.
As the Ansible playbook executor requires password-less SSH access to the endpoints, in the complete workflow we will also create an additional task with the SSH Executor to copy the ssh public key of the Ansible controller to the target host.

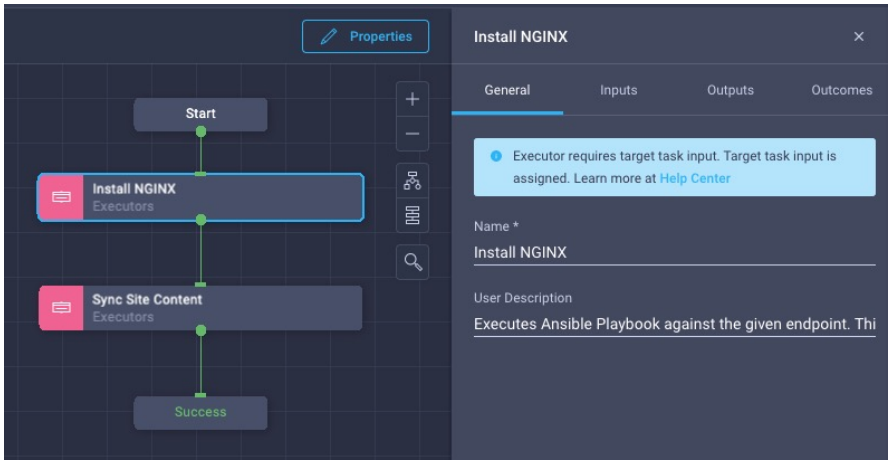
The Playbooks used can be found here: <PLACEHOLDER>

Create an Ansible Custom Task – 1/7
Sample Scenario: Deploy NGINX Webserver



Drag the **Invoke Ansible Playbook** executor in the designer.

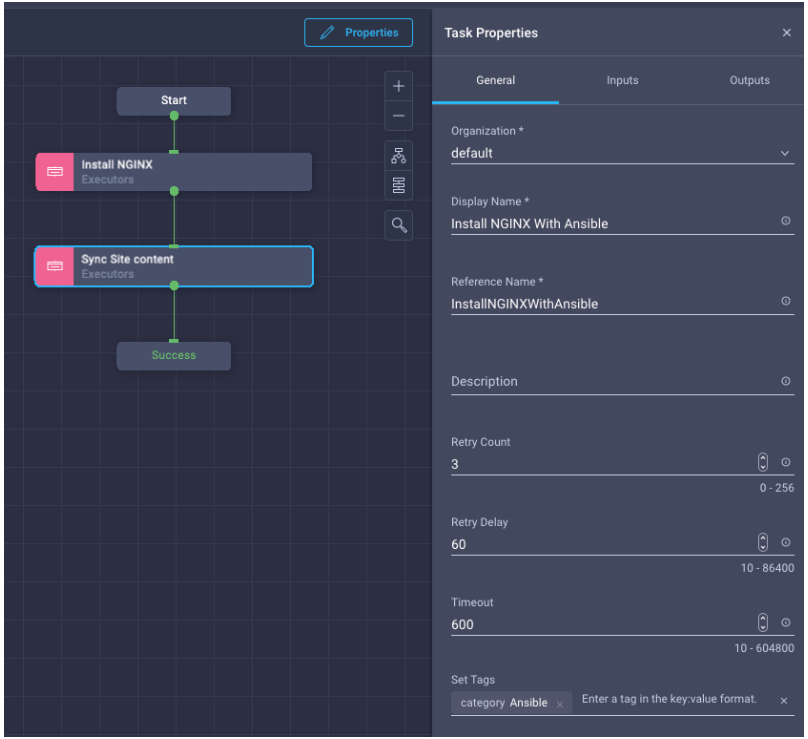
Note: you can't use more than one executor type in a single custom task. However, you can use multiple executors of the same type.



Assign a name and a description to both executors

Create an Ansible Custom Task - 2/7

Sample Scenario: Deploy NGINX Webserver. Custom Task Properties, General

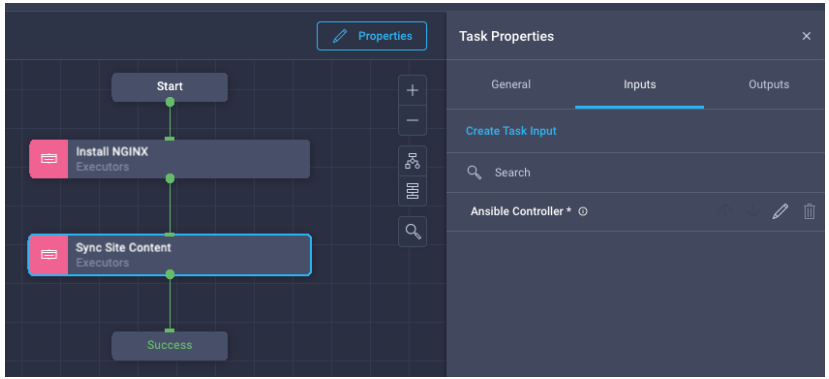


Click on **Properties** to access the custom task properties.
Under the **General** tab, select an **Organization** and specify a **Display Name.**

Optionally, set a category for this custom task using the **Tag** `category:Ansible`

Create an Ansible Custom Task – 3/7

Sample Scenario: Deploy NGINX Webserver. Custom Task Properties, Inputs



Click on **Create Task Input** to create inputs as we want to pass their values in the **Command Line Arguments** inputs of the executors.

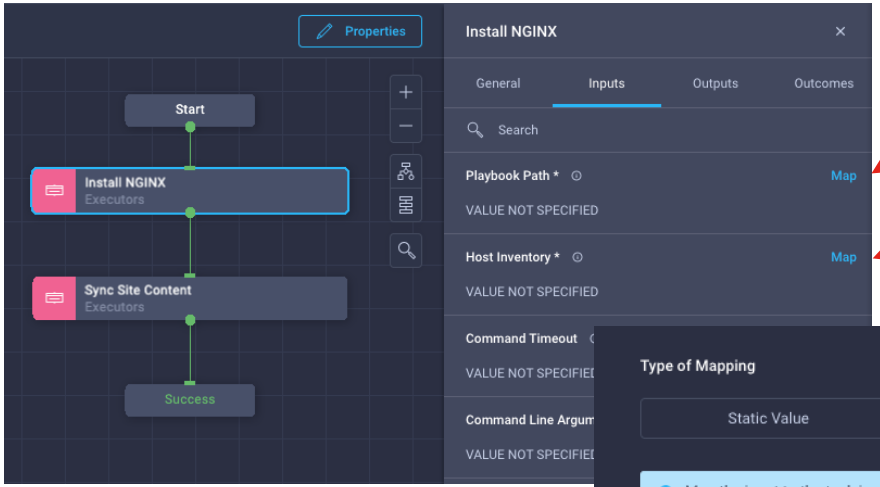
The following table recaps the input required, their names and types

Note: these are specific to this example, if you want users to pass the ansible variables you will need to create inputs based on your implementation

Display Name	Reference Name	Type
Server Domain Name	domain	string
Target Endpoint Ansible User	ansuser	string
SSH Private Key File	sshprivkey	string
Target Endpoint Ansible SUDO Password	anssudopwd	string

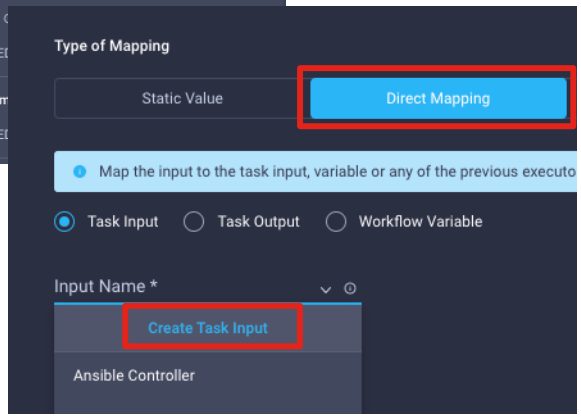
Create an Ansible Custom Task – 4/7

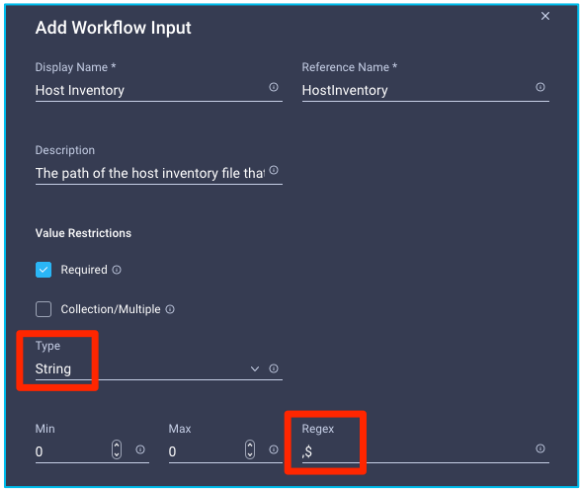
Sample Scenario: Deploy NGINX Webserver. Executor: Install NGINX



Static Mapping of the playbook path in the claimed Ansible control node:
`/home/admin/ansible/nginx/nginx.yml`

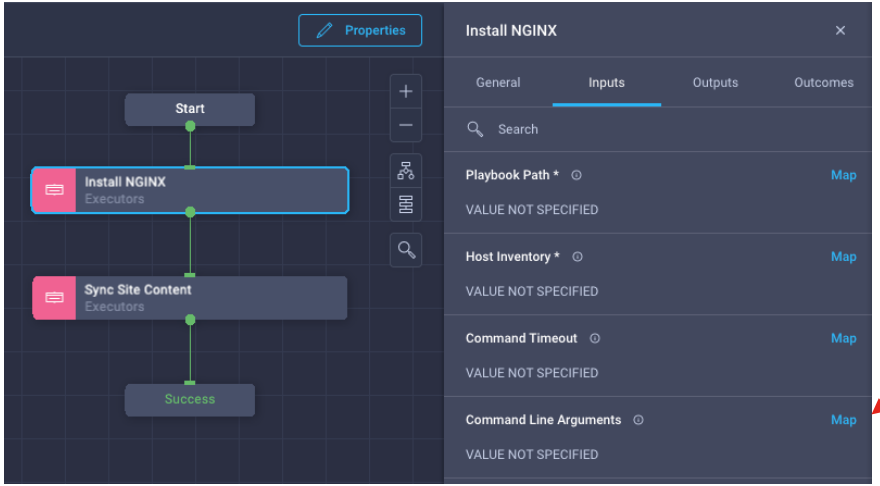
Direct Mapping (create Task Input) as we want to run this playbook to an endpoint or list of endpoint specified in the parent workflow. We set the input type to be a **String** and enforce regex validation with ,$ to be sure we always have the trailing comma (see requirements slide)





Create an Ansible Custom Task - 5/7

Sample Scenario: Deploy NGINX Webserver. Executor: Install NGINX



Static Mapping.

This executor input accepts the command line arguments as you would pass to the ansible-playbook CLI command.
In this case the target playbook accepts variables so we will pass them using the **Command Line Arguments:**

```
-e "domain={{.global.task.input.domain}}
ansible_user={{.global.task.input.ansuser}}
ansible_ssh_private_key_file={{.global.task.inp
ut.sshprivkey}}
ansible_sudo_pass={{.global.task.input.anssudop
wd}}
ansible_python_interpreter=/usr/bin/python3"
```

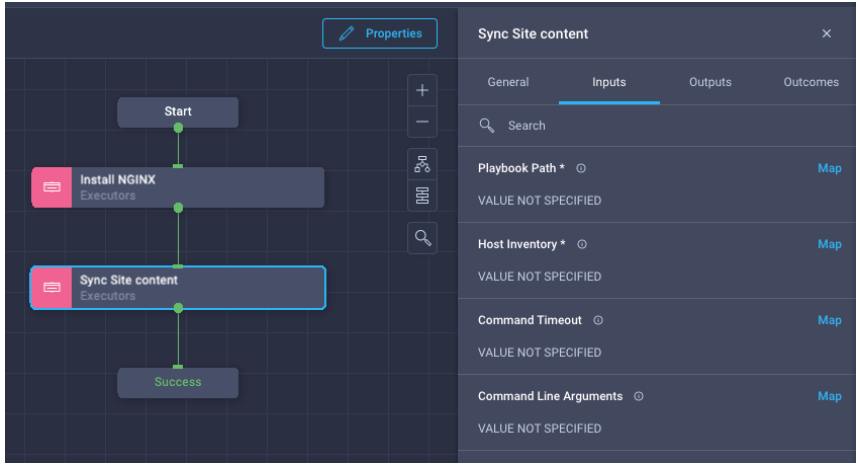
Note that this is specific to the playbook you are using.

Also note that if you are using the Ansible Playbook Executor as an embedded task you are required to escape double quotes with the \ character

Please refer to the following documentation about using variables in Ansible Playbooks:
https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

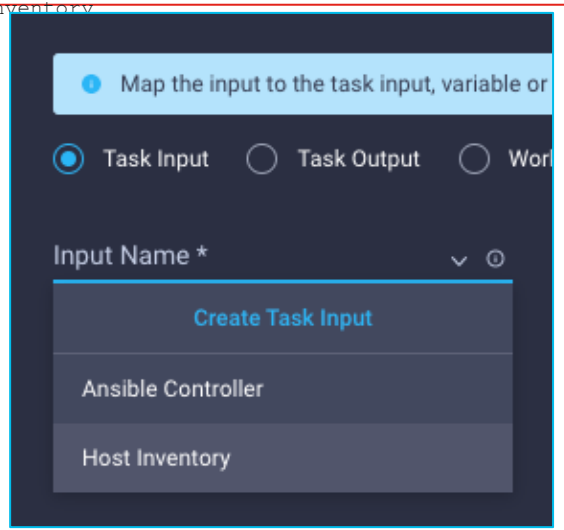
Create an Ansible Custom Task - 6/7

Sample Scenario: Deploy NGINX Webserver. Executor: Sync Site Content



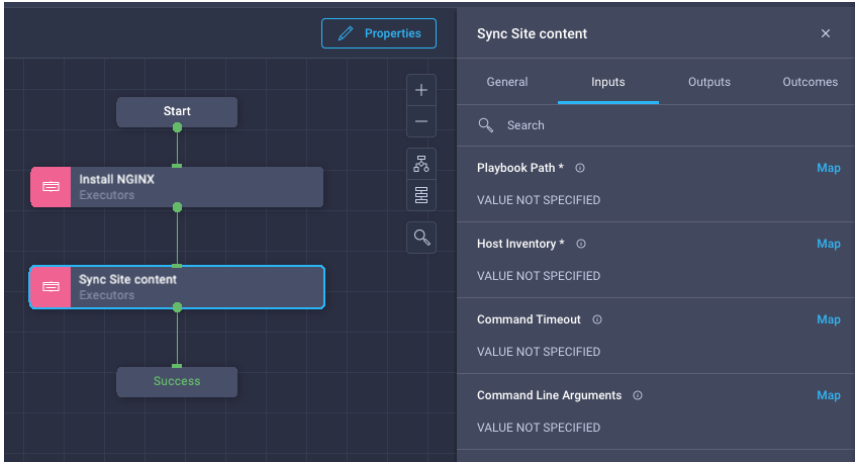
Static Mapping of the playbook path in the claimed Ansible control node:
`/home/admin/ansible/nginx/sync.yml`

Direct Mapping to the created workflow input `Host Inventory`



Create an Ansible Custom Task - 7/7

Sample Scenario: Deploy NGINX Webserver. Executor: Sync Site Content



Static Mapping.

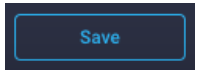
This executor input accepts the command line arguments as you would pass to the ansible-playbook CLI command.
In this case the target playbook accepts variables so we will pass them using the **Command Line Arguments:**

```
-e "domain={{.global.task.input.domain}}
ansible_user={{.global.task.input.ansuser}}
ansible_ssh_private_key_file={{.global.task.inp
ut.sshprivkey}}
ansible_sudo_pass={{.global.task.input.anssudop
wd}}
ansible_python_interpreter=/usr/bin/python3"
```

Note that this is specific to the playbook you are using.

Also note that if you are using the Ansible Playbook Executor as an embedded task you are required to escape double quotes with the \ character

Please refer to the following documentation about using variables in Ansible Playbooks:
https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html



Save the task

Create a Custom Task to configure password-less SSH 1/2

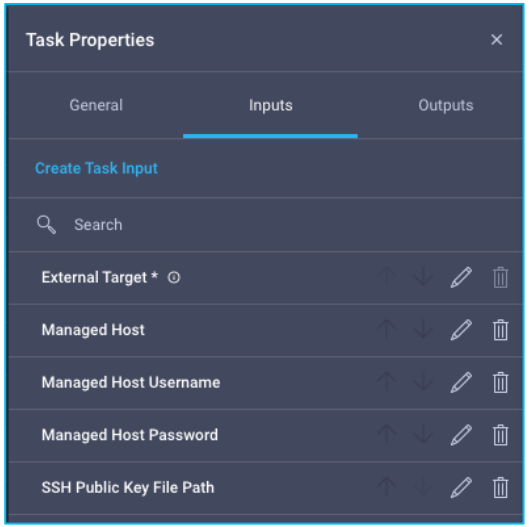
Sample Scenario: Deploy NGINX Webserver

Recall that in order to work, Ansible requires password-less SSH access to the targets it's executing against.
If the Ansible controller already has password-less SSH access to the target, no additional steps are needed. However, there are cases where we want to target freshly created hosts that may not have granted access to the Ansible controller yet.

To overcome this issue, we will create a custom task using the SSH executor that injects the Ansible target public key into the endpoint hosts (refer to the SSH executor section to get more info on creating SSH custom tasks)

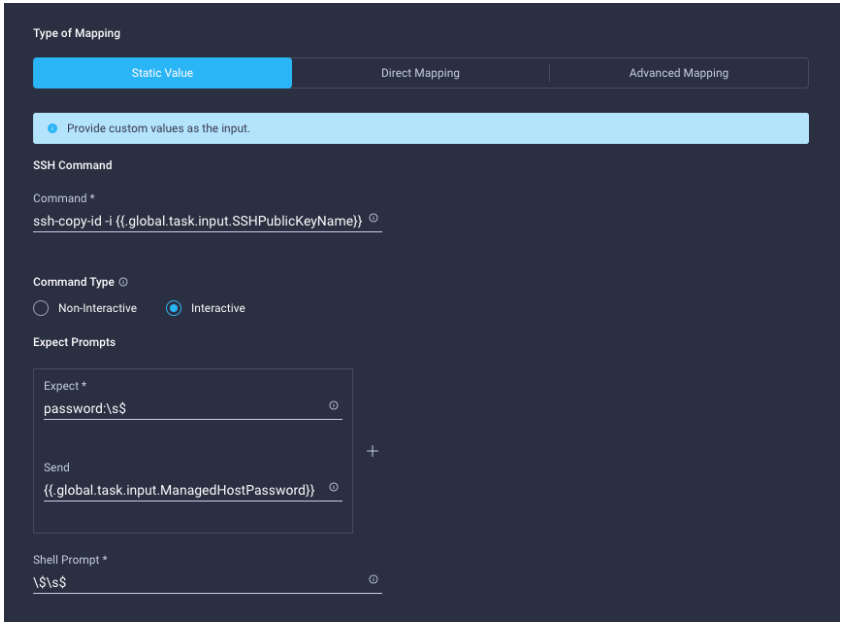
`Inject SSH Key` custom task inputs:

Display Name	Reference Name	Type
Managed Host	ManagedHost	string
Managed Host Username	ManagedHostUsername	string
Managed Host Password	ManagedHostPassword	string
SSH Public Key File Path	SSHPublicKeyName	string



Create a Custom Task to configure password-less SSH 2/2

Sample Scenario: Deploy NGINX Webserver



Command:

```
ssh-copy-id -i
{{.global.task.input.SSHPublicKeyName}}
{{.global.task.input.ManagedHostUsername}}@{{.g
lobal.task.input.ManagedHost}} -o
"StrictHostKeyChecking no"
```

Command-Type: **Interactive**

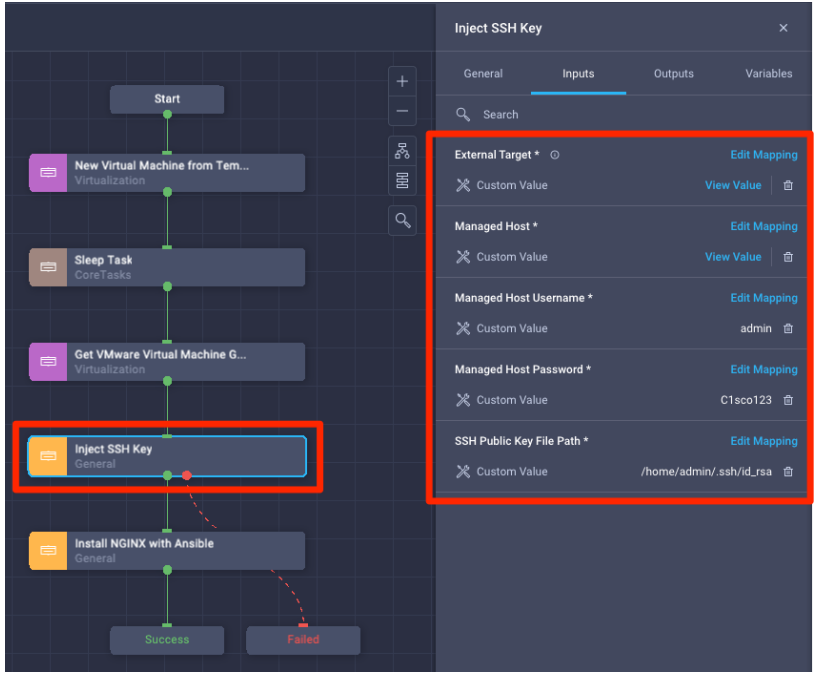
Expect:
`password:\s$`

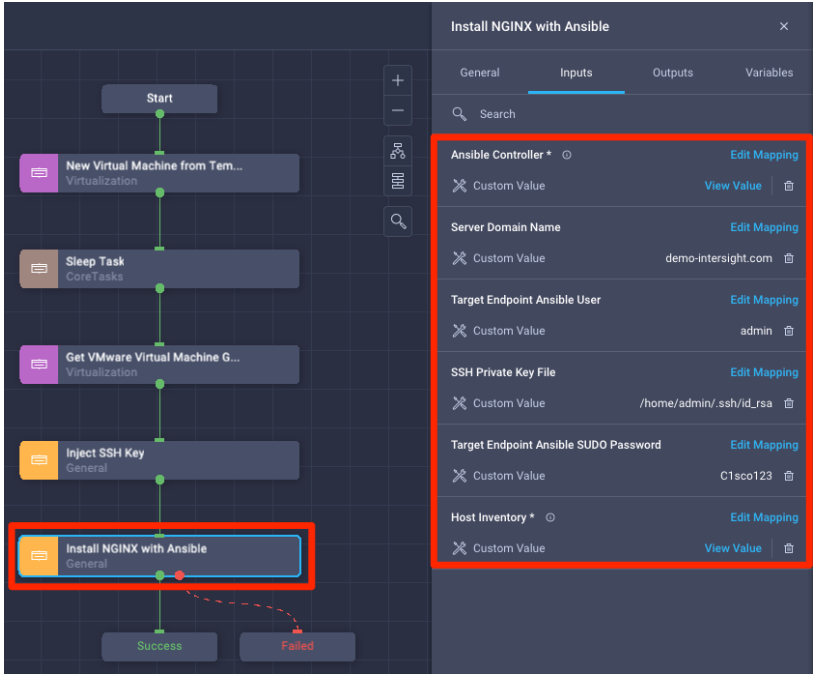
Send:
`{{.global.task.input.ManagedHostPassword}}`

Shell Prompt:
`\$\s$`

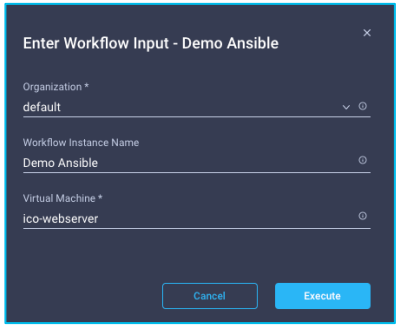
This task will simply execute the ssh-copy-id command in the Ansible target against another target host. From that point on, it will be able to have password-less SSH access. **Save** the task once done

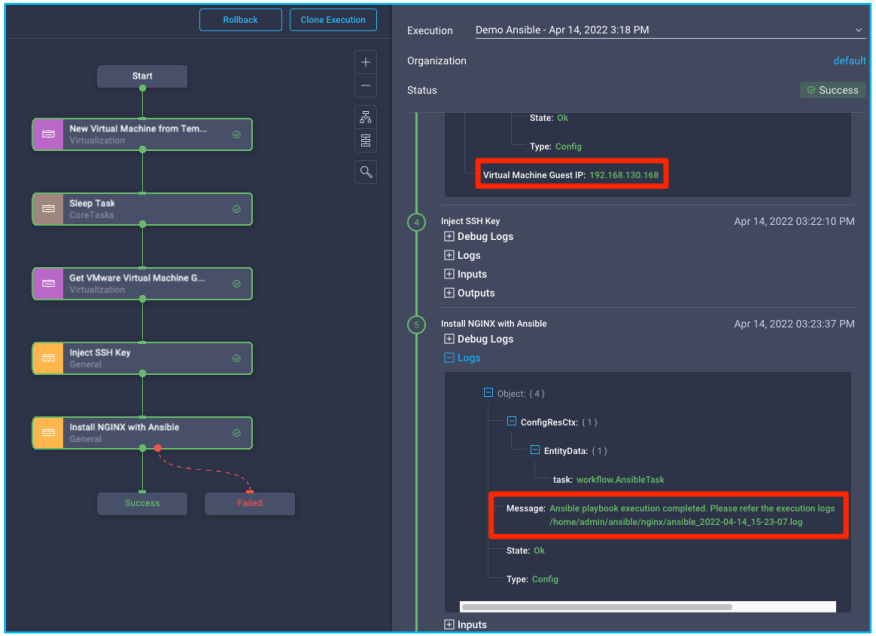
Sample End to End Workflow
Sample Scenario: Deploy NGINX Webserver

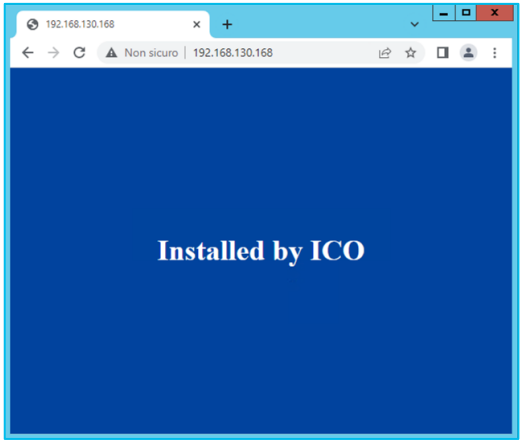




Sample End to End Workflow
Sample Scenario: Deploy NGINX Webserver







Go Template Cheat Sheet

Referencing variables in the Workflow Designer

Usage	Description	Code
Workflow Input	You can reference a workflow input by replacing NAME with the the reference name you have given it inside the workflow designer	{{.global.workflow.input.NAME}}
Task Output	You can reference a task output by replacing TASK with the name of the task (from the Code view), and by replacing NAME with the name of the output	{{.global.TASK.output.NAME}}
Org Moid	The Moid of the Intersight org used to execute this workflow	{{.security.OrganizationMoid}}

Referencing variables in the Task Designer

Usage	Description	Code
Task Input	You can reference a task input by replacing NAME with the the reference name you have given it inside the task designer	{{.global.task.input.NAME}}
Sub-Task Output	You can reference a sub-task output by replacing SUBTASK with the reference name of the sub-task, and by replacing NAME with the name of the output	{{.global.SUBTASK.output.NAME}}

Advanced functions 1/3

Usage	Description	Code
Conditional	Will allow you to select what option you want to execute based on a conditional. Replace VALUE with the specific value you want to compare against.	{{if (eq .global.workflow.input.example VALUE)}} IF TRUE {{else if eq .global.workflow.input.example VALUE}} ELIF TRUE {{else}} ELSE {{end}}
Optional variable	Only executes a block of the template if the variable was provided in the input. Inside of the with block, you refer to the name of the original variable as $x (or whatever variable you use)	{{with $x := .global.workflow.input.example}} {{$x}} {{end}}

Advanced functions 2/3

Usage	Description	Code
Regex	Apply a regex pattern to a string to filter out a sub-string. Replace REGEX with your regex pattern. This can be combined with the index feature below to filter out the n-th sub-string.	{{FindAllString .global.workflow.input.example “REGEX”}}
Get n-th entry	Allows you to get the n-th entry from a list. Replace NUMBER with the position of the list item you want to select.	{{index .global.workflow.input.example NUMBER}}
Sub-parameter	Allows you to select the subparameter of a previous operation. Replace SUBELEMENT with the parameter you are looking for	{{(index .global.workflow.input.example NUMBER).SUBELEMENT}}

Advanced functions 3/3

Usage	Description	Code
Loop	Allows you to loop through a list and execute a template for each entry in the list. If you want to prepend something to your list, you can use if $index to only apply on the first pass of the loop	{{range $i, $element := .global.workflow.input.example}} {{$element}} {{end}}
List length	Get the length of a list	{{len .global.workflow.input.example}}

ICO Intersight API

Intersight APIs

Intersight APIs are documented here:
https://intersight.com/apidocs/introduction/overview/

The Intersight API is based on the OpenAPI Specification standard:
https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.3.md

Requirements:

- Basic knowledge of REST APIs and JSON
- Basic knowledge of Postman
- Create an API and Secret Key in Intersight
- APIs can be invoked/tested also from the embedded REST client here:
  https://intersight.com/apidocs/apirefs/aaa/AuditRecords/model/
- Intersight APIs requires all requests to be signed. The following GitHub repo provides a Postman collection that will take care of it automatically
  https://github.com/CiscoDevNet/intersight-postman
- Examples in this document use the following postman collection (fork from the DevNet repo):
  https://github.com/rtortori/intersight-postman/tree/ico-workflows

ICO Intersight APIs

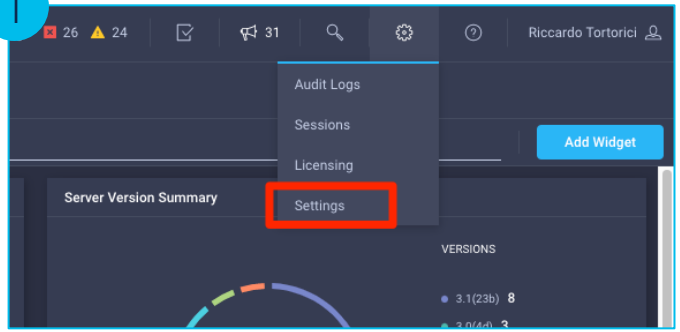
ICO APIs are genererally under the **workflow** path : /api/v1/workflow
They can be explored and tested from
https://intersight.com/apidocs/apirefs/aaa/AuditRecords/model/
You will need to be logged in to use the REST Client.

In this document, the following APIs will be covered:

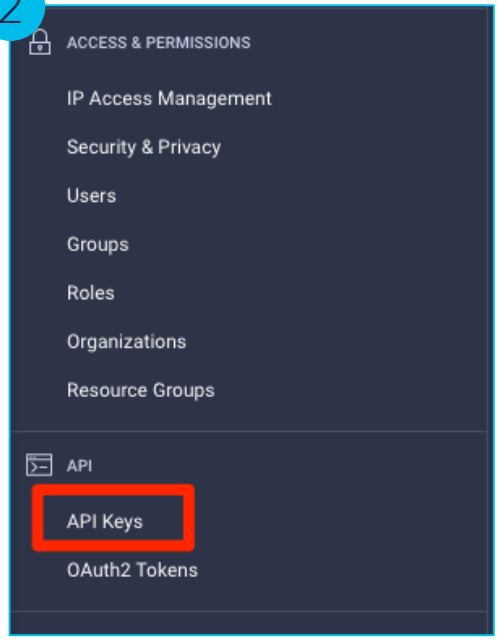
Scope	Description	Path
Workflows	Workflows Definitions	/api/v1/WorkflowDefinitions
Workflows	Workflow Executions (Requests)	/api/v1/WorkflowInfos
Workflows	Executions Rollbacks	/api/v1/RollbackWorkflows
Tasks	Task info on Workflow Executions (i.e. outputs)	/api/v1/TaskInfos
Tasks	Task Definitions	/api/v1/TaskDefinitions

Create an API and Secret Key in Intersight 1/2

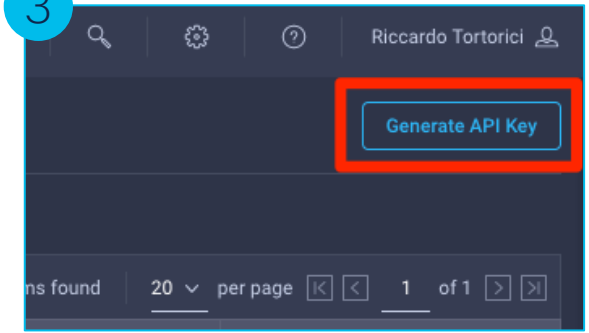
1



2



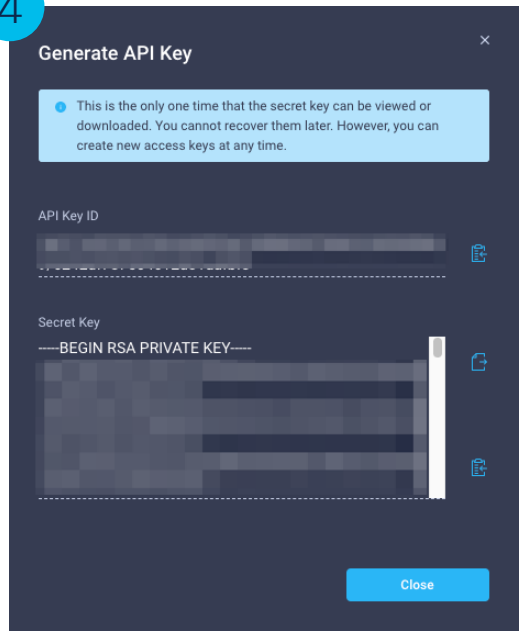
3



Create an API and Secret Key in Intersight 2/2

3

4



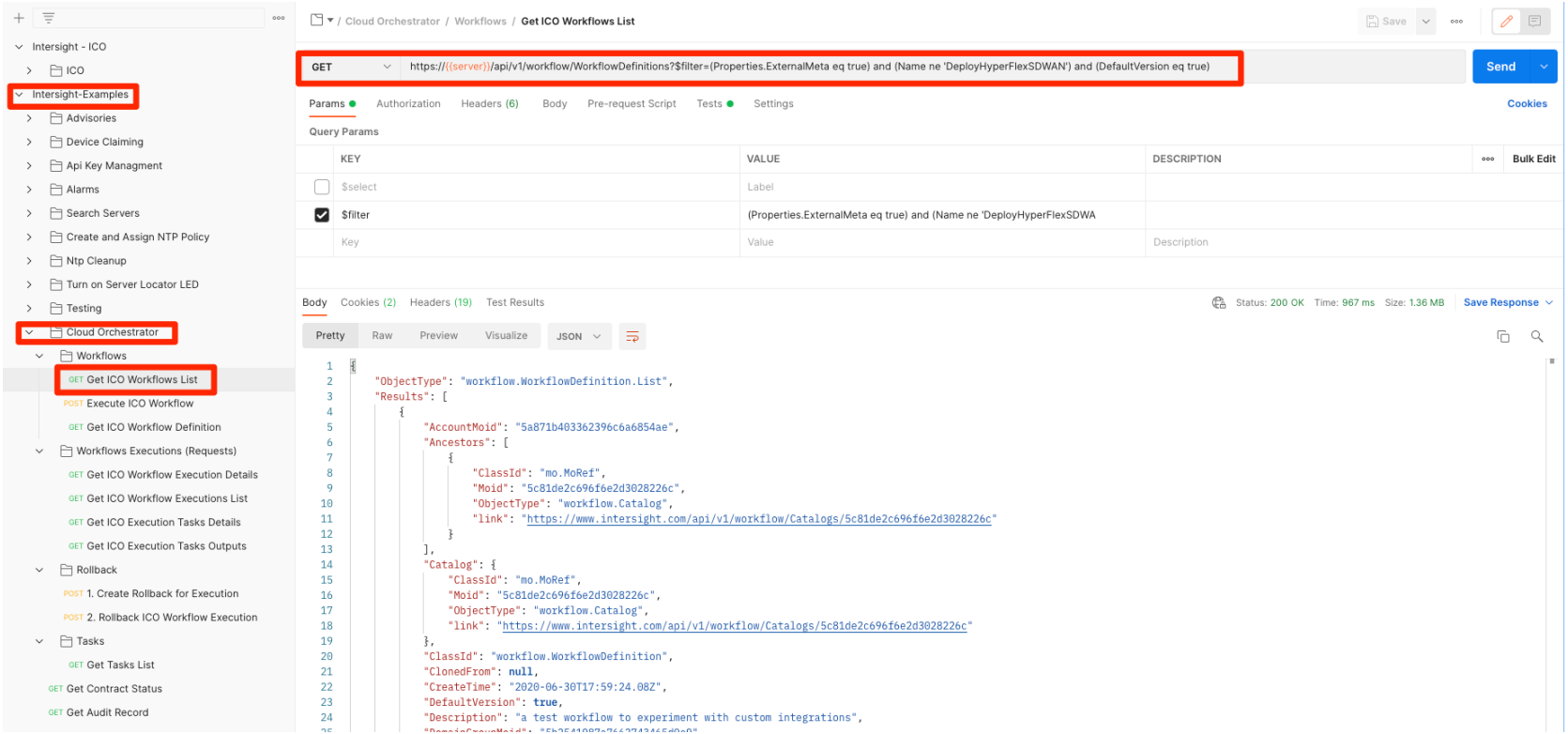
Take note of the **API Key ID** and **Secret Key**

Note:
This will be the only time the **Secret Key** will be shown. If you lose it, you'll need to generate another API key

Postman Setup

1. Download and Import the collection in Postman
2. Set the Api Key and Secret Key in the **Intersight** Postman Environment and **Save**

Example: Get ICO Workflow List

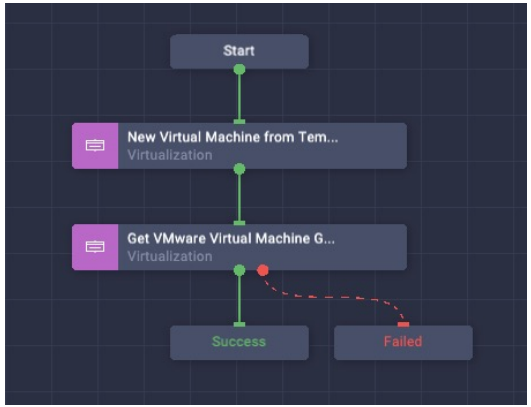


Example: Execute a Workflow 1/4

A popular use case is to execute a workflow using the API from a 3rd party application, orchestrator or service catalog.

While the payload structure is always the same, depending on which workflow you want to execute, you may have different inputs.
A possible strategy would be to invoke your workflow manually and capture the POST call done by the browser. This is an action you only have to do once and will help you shaping your call (i.e. a service catalog item creation).

In this case, we are executing a Workflow named Deploy a Virtual Machine



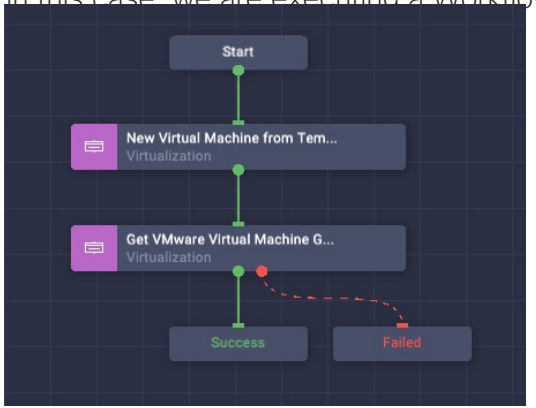
This workflow has two tasks:
1. Deploys a Virtual Machine
2. Query the vCenter to extract the IP of the VM

Example: Execute a Workflow 1/5

A popular use case is to execute a workflow using the API from a 3rd party application, orchestrator or service catalog.

While the payload structure is always the same, depending on which workflow you want to execute, you may have different inputs.
A possible strategy would be to invoke your workflow manually and capture the **POST** call done by the browser. This is an action you only have to do once and will help you shaping your call (i.e. a service catalog item creation).

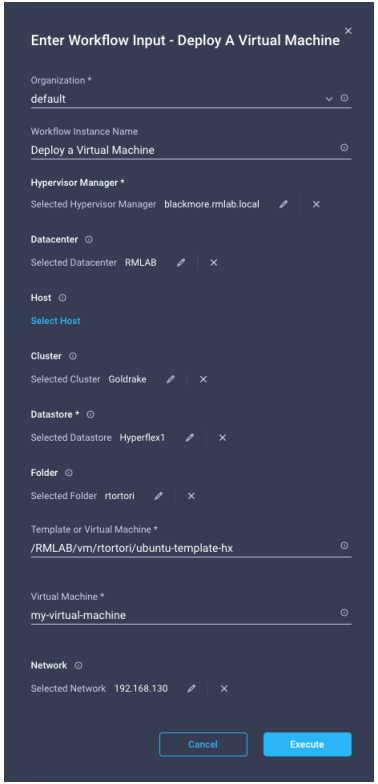
In this case, we are executing a Workflow named **Deploy a Virtual Machine**



This workflow has two tasks:

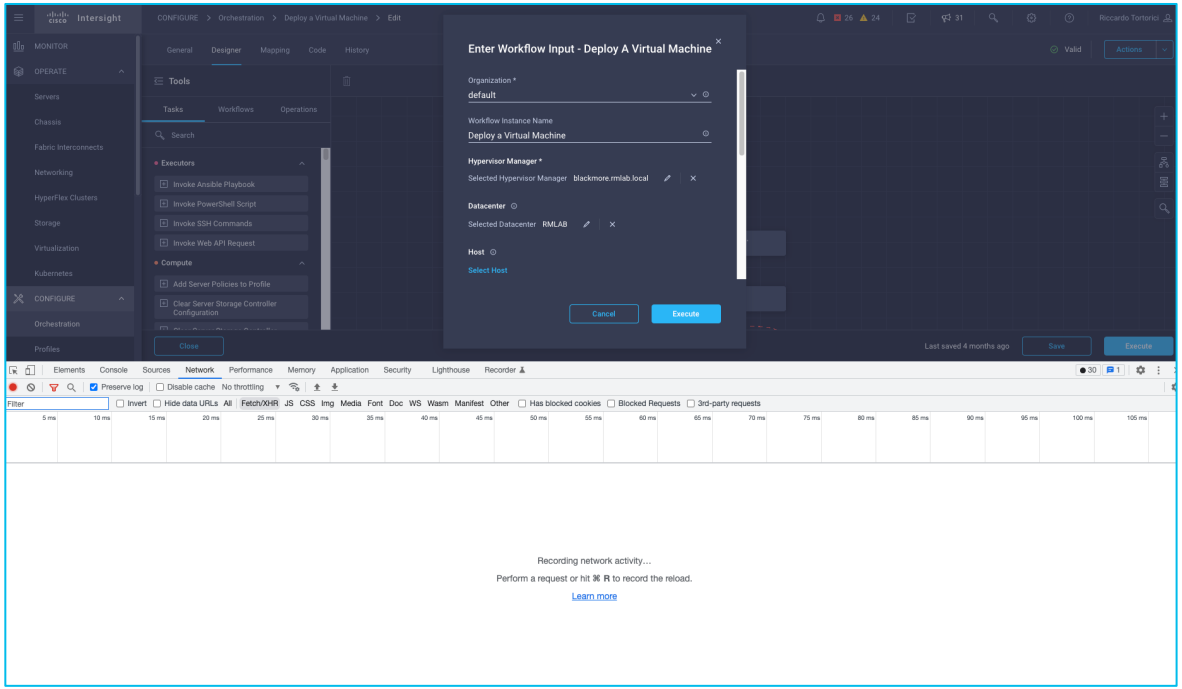
1. Deploys a Virtual Machine
2. Query the vCenter to extract the IP of the VM

Click on **Execute** and fill in with your inputs as you were deploying the workflow manually.
DO NOT launch the workflow execution once done but keep the window opened

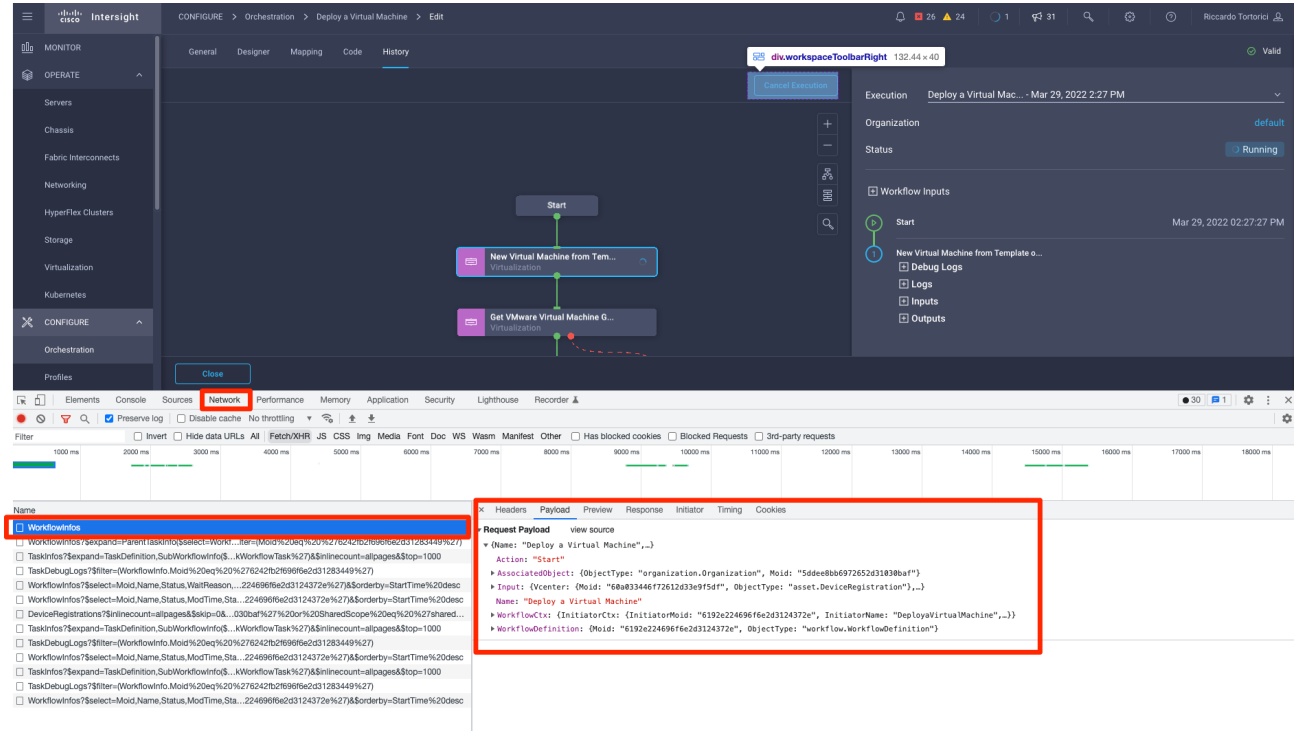


Example: Execute a Workflow 2/5

Open the developer tools of your browser and go to the **Network** tab



Example: Execute a Workflow 3/5



Open the developer tools of your browser and go to the **Network** tab

Copy the payload

Example: Execute a Workflow 4/5

This is the payload to be used to invoke the workflow:

```
{
    "Name": "Deploy a Virtual Machine",
    "AssociatedObject":
    {
        "ObjectType": "organization.Organization",
        "Moid": "aabbccddeeffgghhjjkk"
    },
    "Action": "Start",
    "Input":
    {
        "Vcenter":
        {
            "Moid": "aabbccddeeffgghhjjkk",
            "ObjectType": "asset.DeviceRegistration"
        },
        "Datacenter": "/RMLAB",
        "Cluster": "/RMLAB/host/Goldrake",
        "Datastore": "/RMLAB/datastore/Hyperflex1",
        "Folder": "/RMLAB/vm/rtortori",
        "Template": "/RMLAB/vm/rtortori/ubuntu-template-hx",
        "Name": "my-virtual-machine",
        "Network": "192.168.130"
    },
    "WorkflowDefinition":
    {
        "Moid": "aabbccddeeffgghhjjkk",
        "ObjectType": "workflow.WorkflowDefinition"
    }
}
```

Name of the Workflow to be executed

Moid (Managed Object ID) of your organization. In this example, "default"

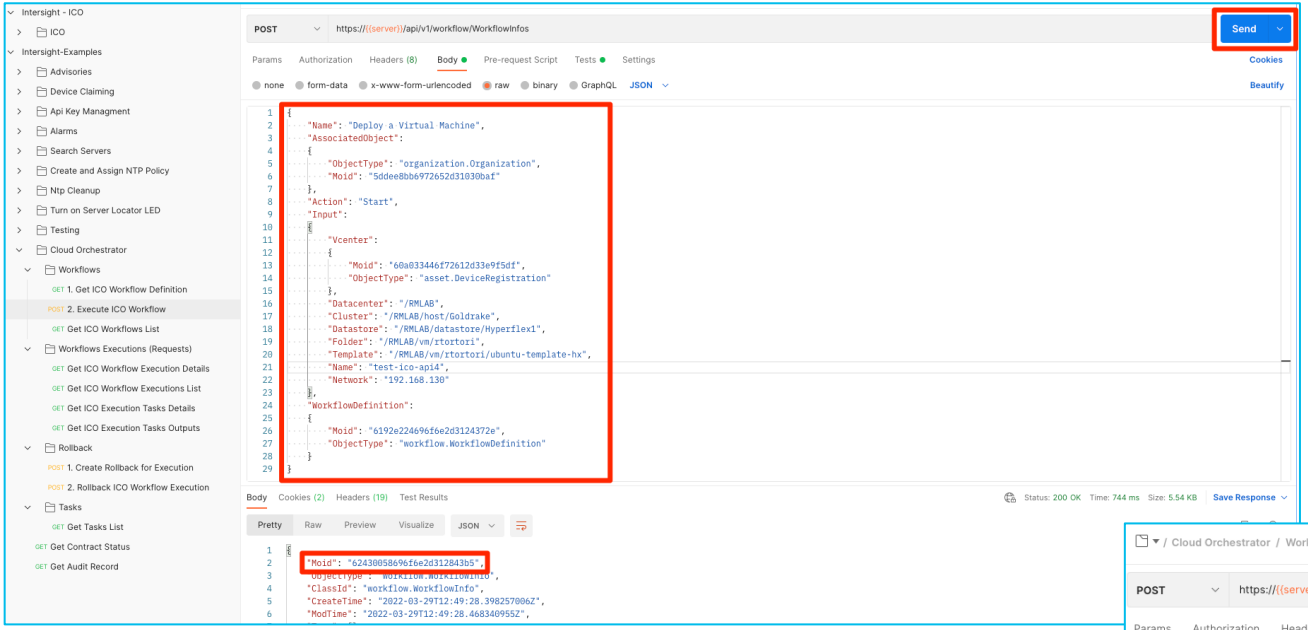
Inputs

Moid of the workflow

Replace inputs based on what you want to create. For instance, change the Virtual Machine **Name**

Example: Execute a Workflow 5/5

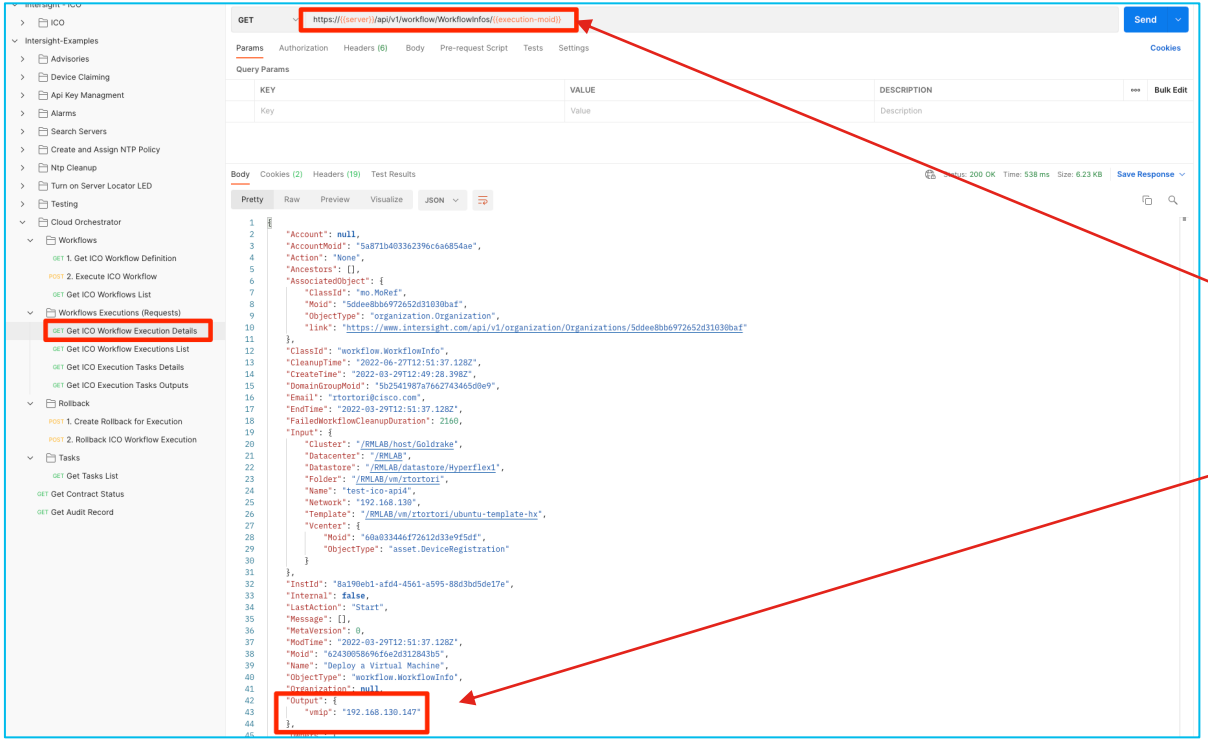
Copy the payload in the **Body** tab of the request **Intersight-Examples -> Cloud Orchestrator -> Workflows -> 2. Execute ICO Workflow**
Click **Send** to execute the workflow



In this postman collection, we will capture the **Moid** of the execution and save it in a global variable named **execution-moid**

We can then use this Moid to invoke other actions, such as get the workflow tasks output and rollback the request (next slides)

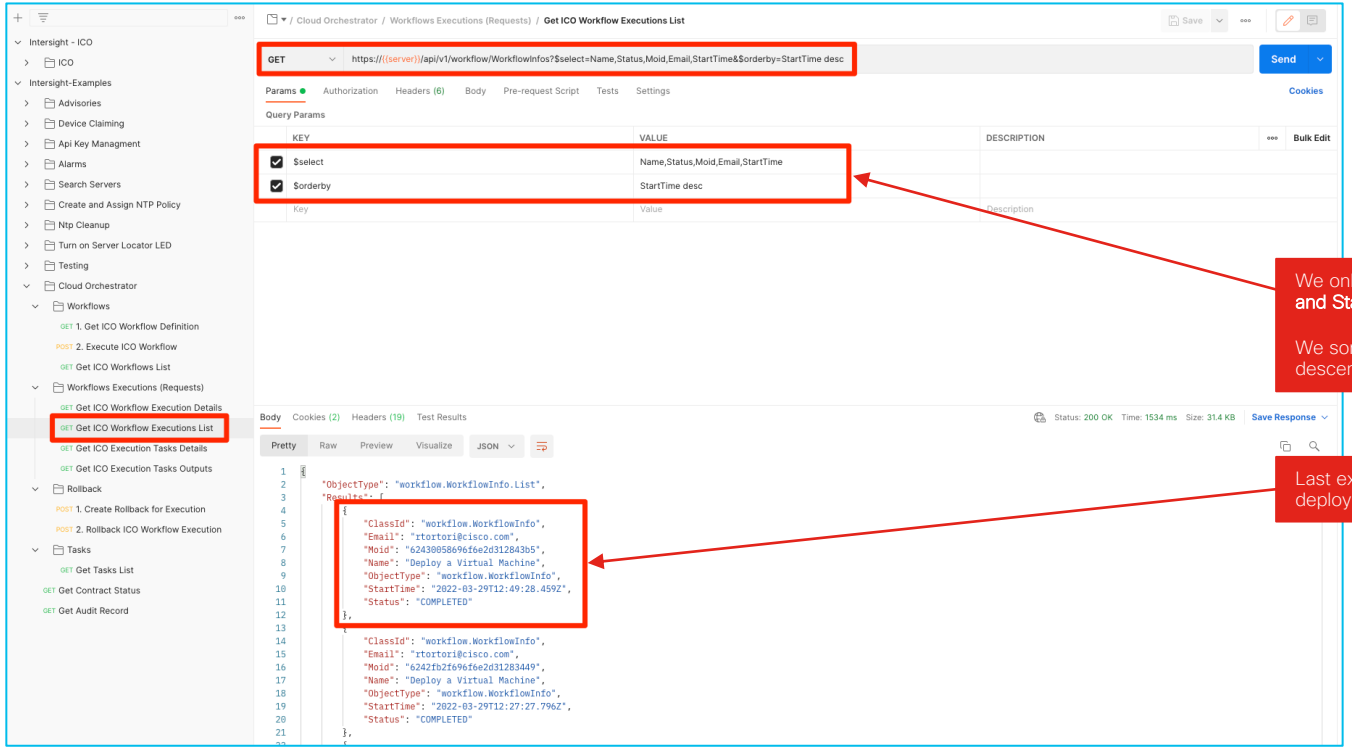
Example: Get Workflow Outputs



We use the **execution-moid** extracted from the workflow execution call

We fetch the output of the workflow, in this case the Virtual Machine IP

Example: Get the List of the Workflow Executions

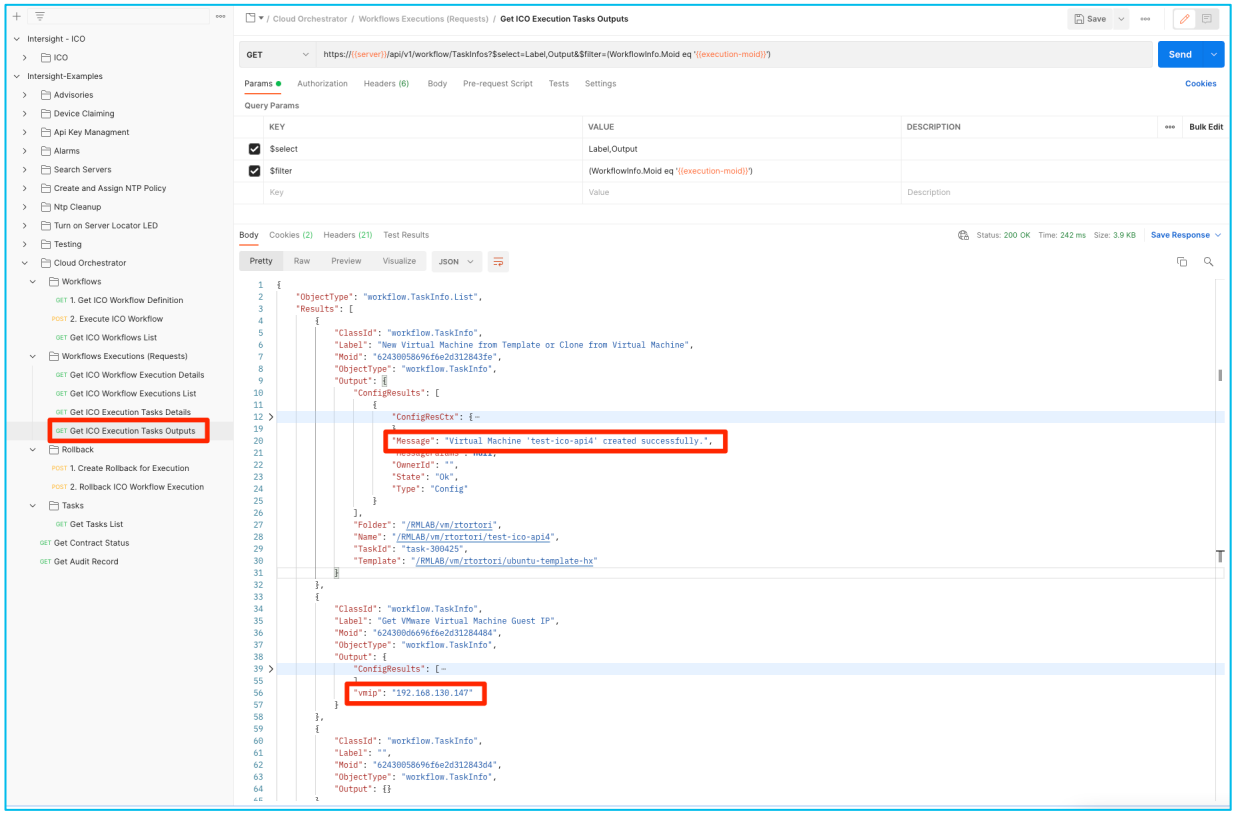


We only show the **Name, Status, Moid, Email and StartTime** ($select)

We sort the results by **StartTime** in descending (**desc**) order

Last execution, the virtual machine deployment we triggered earlier

Example: Get Workflow Tasks Outputs



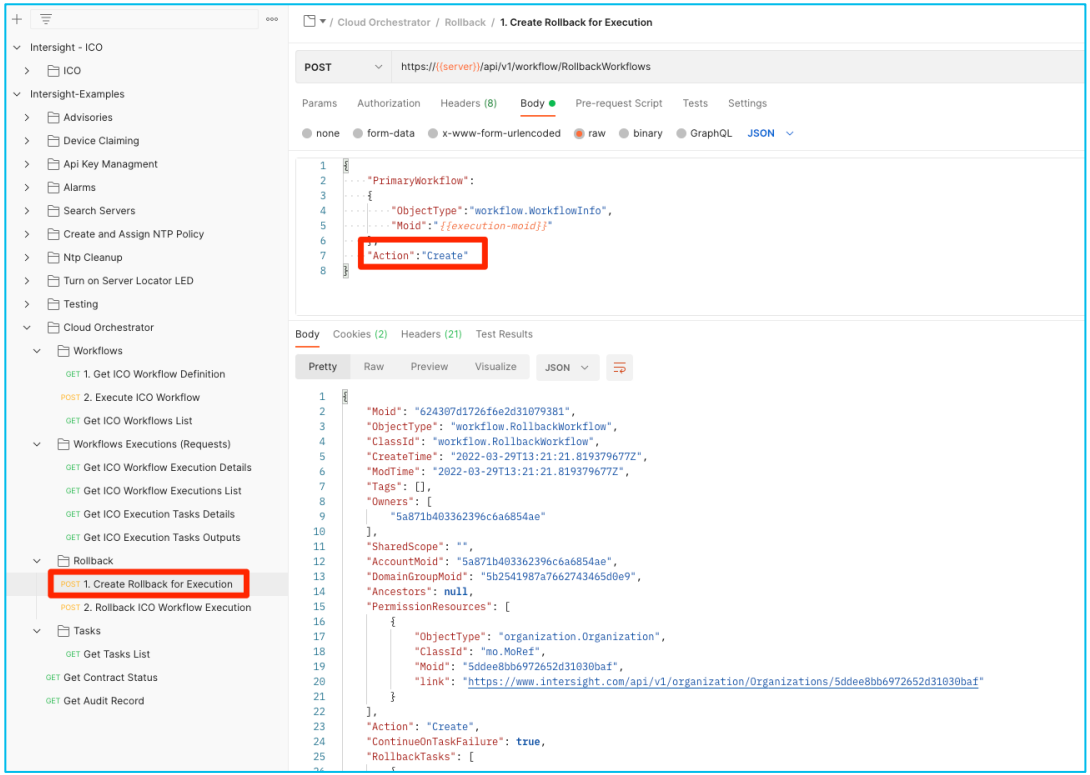
Example: Rollback an Execution (Request) 1/3

In order to rollback an execution you need to know the **Moid** of the request. You can use the **execution-moid** variable extracted earlier or you can use the **Get ICO Workflow Execution Details** postman request to select the execution you want to rollback.

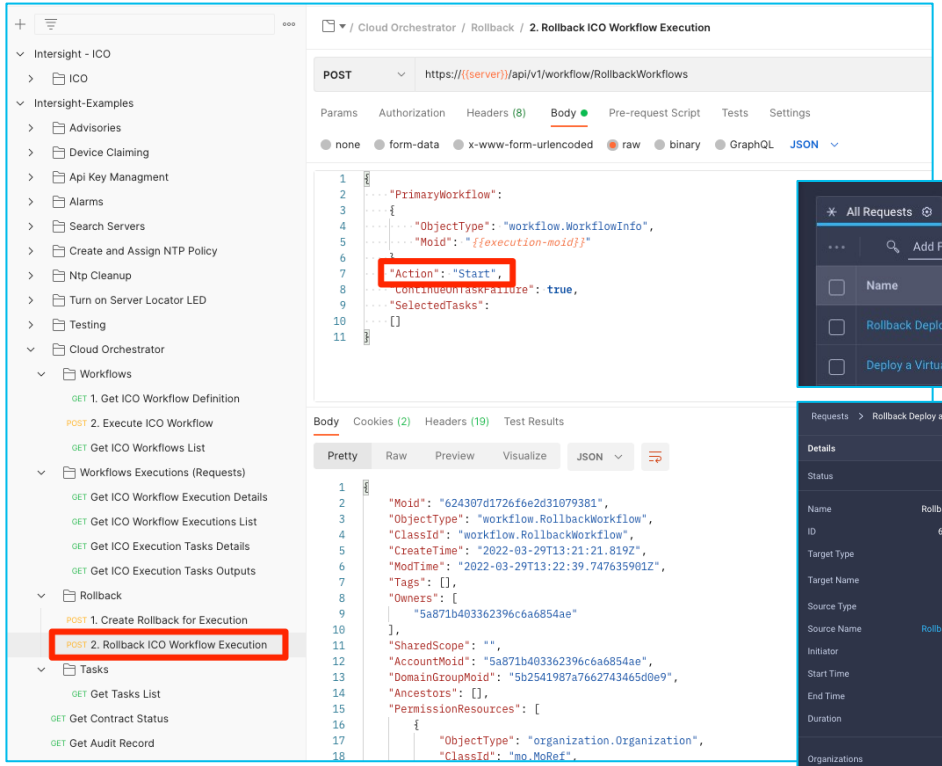
To rollback, you need to make two API Calls:

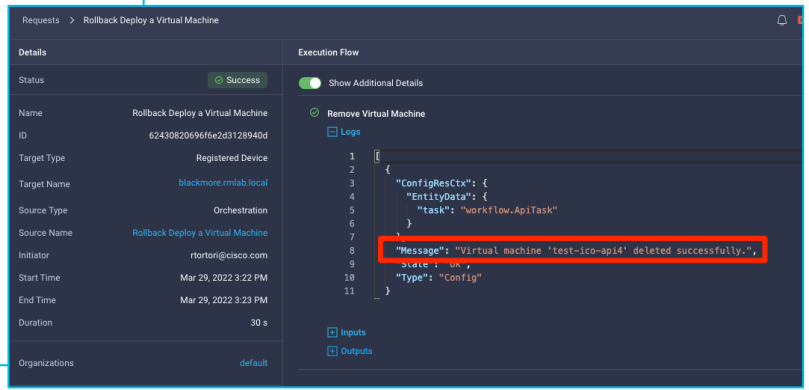
1. Create the rollback
2. Execute the rollback

Example: Rollback an Execution (Request) 2/3



Example: Rollback an Execution (Request) 3/3

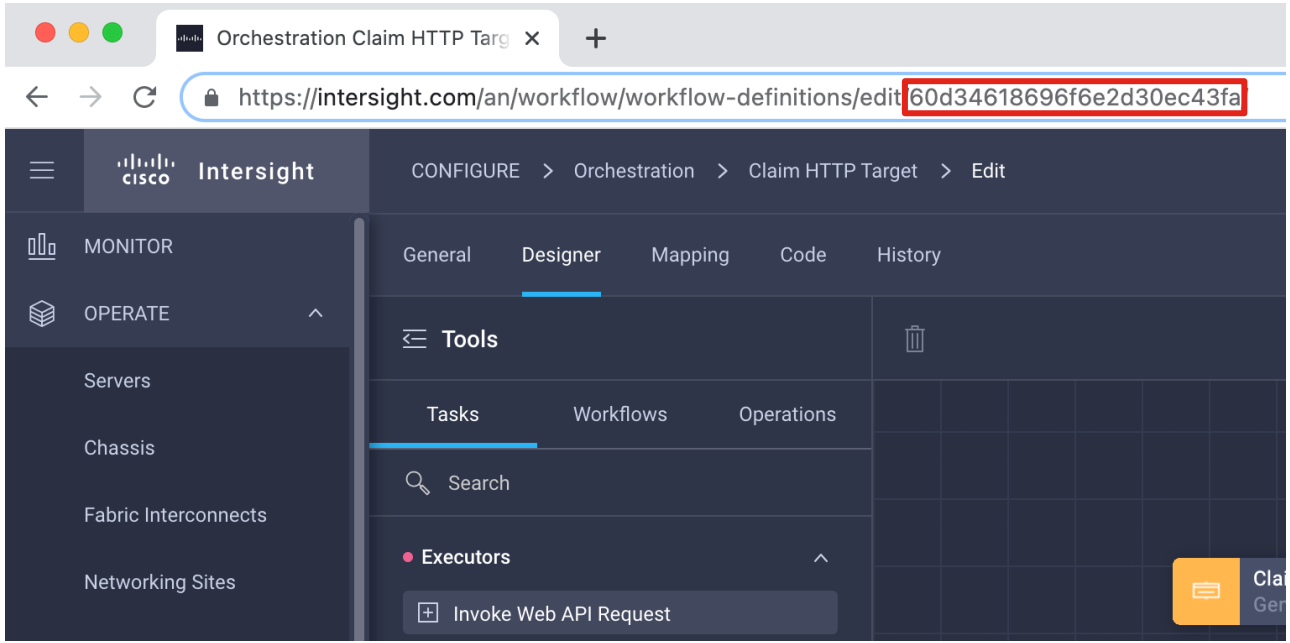




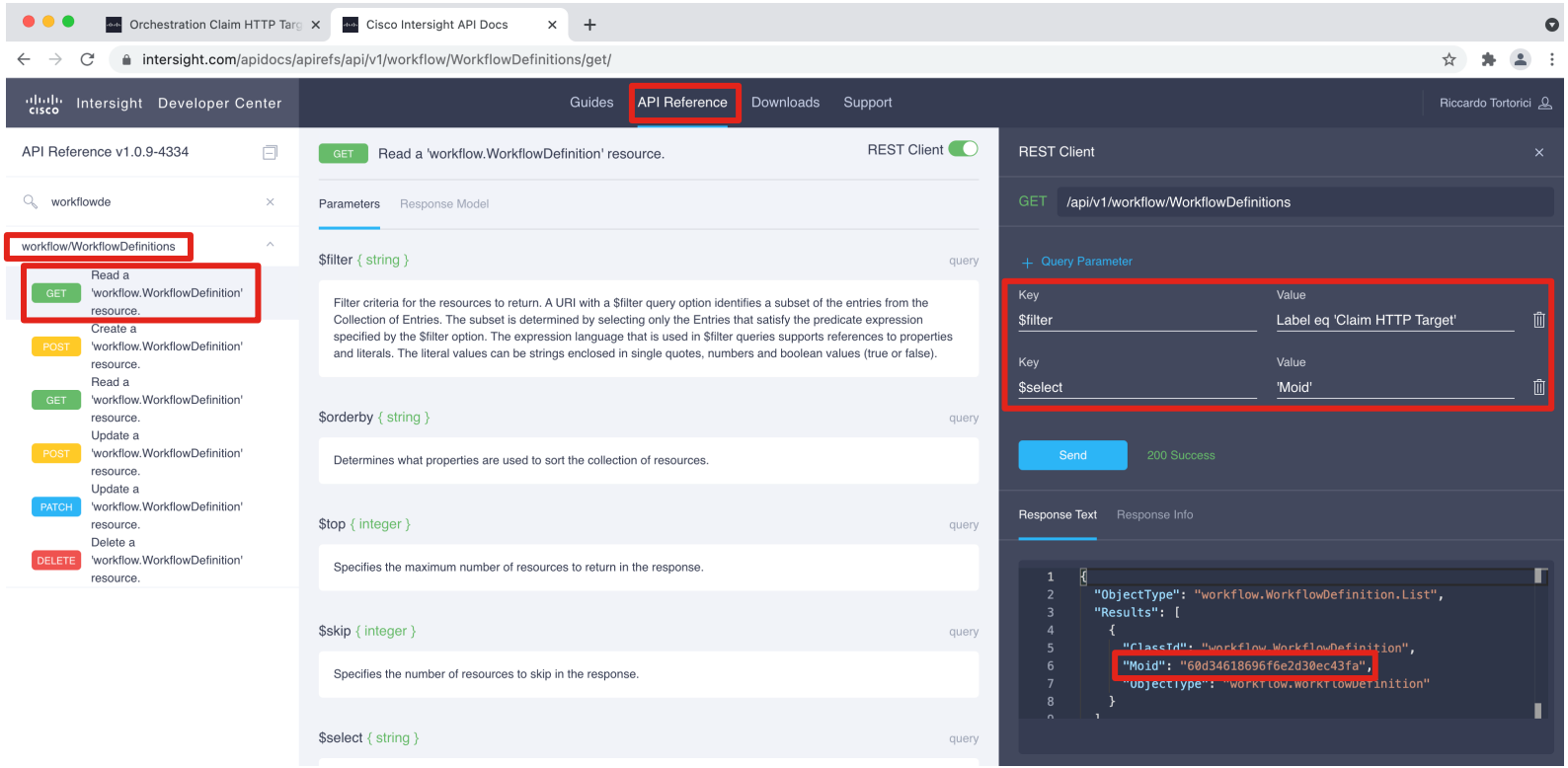
Execute an ICO Workflow from Terraform

(Provider version 1.0.11)

Get the Target Workflow MOID (The Easiest Way)

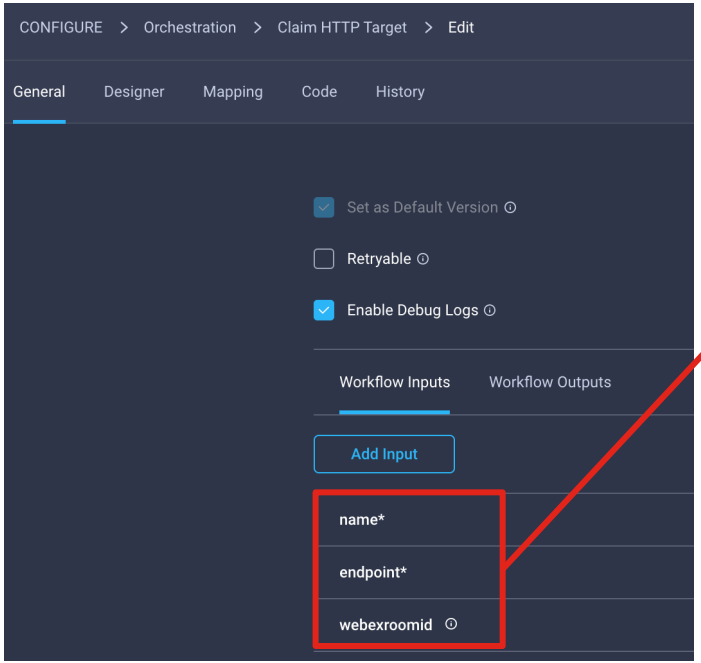


Get the Target Workflow MOID (The API Way)



API Docs: https://www.intersight.com/apidocs

Terraform Resource



Sample code and instructions here:
https://github.com/rtortori/intersight-terraform/tree/main/workflows

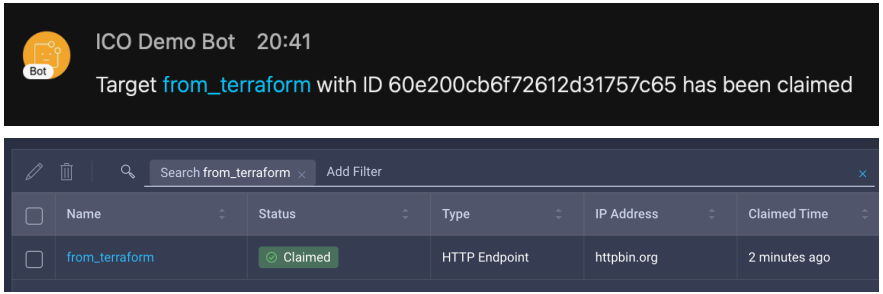
```
resource "intersight_workflow_workflow_info" "myworkflow" {
  name   = var.workflowname
  action = "Start"

  input = {
    "name"        = "from_terraform"
    "endpoint"    = "httpbin.org"
    "webexroomid" = var.roomid
  }

  organization {
    object_type = "organization.Organization"
    moid        = data.intersight_organization_organization.myorg.id
  }
  workflow_definition {
    object_type = "workflow.WorkflowDefinition"
    moid        = var.workflow_moid
  }
}
```

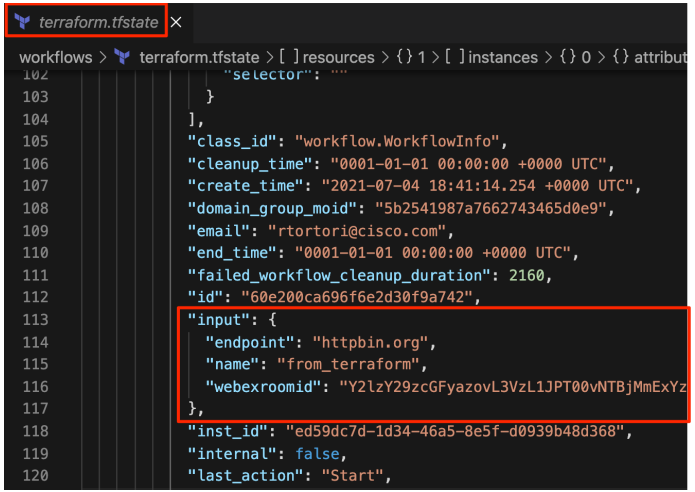
Specify your input variables as if you were executing the workflow from the UI. Ensure you respect types and constraints.

Provider documentation:
https://registry.terraform.io/providers/CiscoDevNet/intersight/latest/docs/resources/workflow_workflow_info

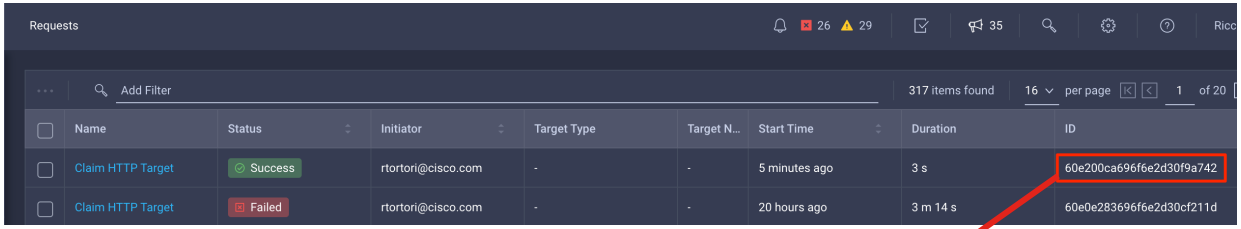
Terraform Apply

```
➜  workflows git:(main) > terraform apply --auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # intersight_workflow_workflow_info.myworkflow will be created
  + resource "intersight_workflow_workflow_info" "myworkflow" {
[...]

Plan: 1 to add, 0 to change, 0 to destroy.
intersight_workflow_workflow_info.myworkflow: Creating...
intersight_workflow_workflow_info.myworkflow: Creation complete after 1s [id=60e200ca696f6e2d30f9a742]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
➜  workflows git:(main) >
```



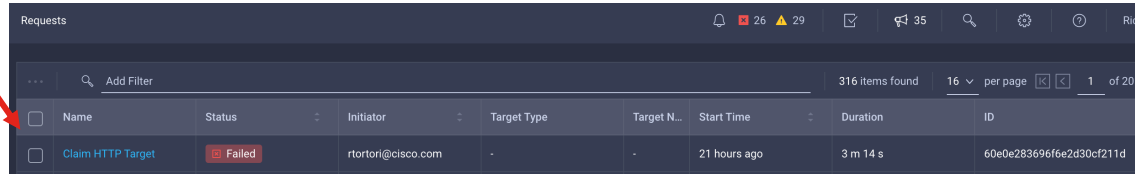


Terraform Destroy




```
➜  workflows git:(main) ✗ > terraform destroy -auto-approve
intersight_workflow_workflow_info.myworkflow: Refreshing state... [id=60e200ca696f6e2d30f9a742]

[...]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # intersight_workflow_workflow_info.myworkflow will be destroyed
  - resource "intersight_workflow_workflow_info" "myworkflow" {

 [...]

Plan: 0 to add, 0 to change, 1 to destroy.
intersight_workflow_workflow_info.myworkflow: Destroying... [id=60e200ca696f6e2d30f9a742]
intersight_workflow_workflow_info.myworkflow: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
➜  workflows git:(main) ✗ >
```

As of July 2021, **`terraform destroy`** will delete the request in Intersight but will **NOT** rollback the workflow execution (in this example the target will be still claimed)



Execute an ICO Workflow with the Python SDK

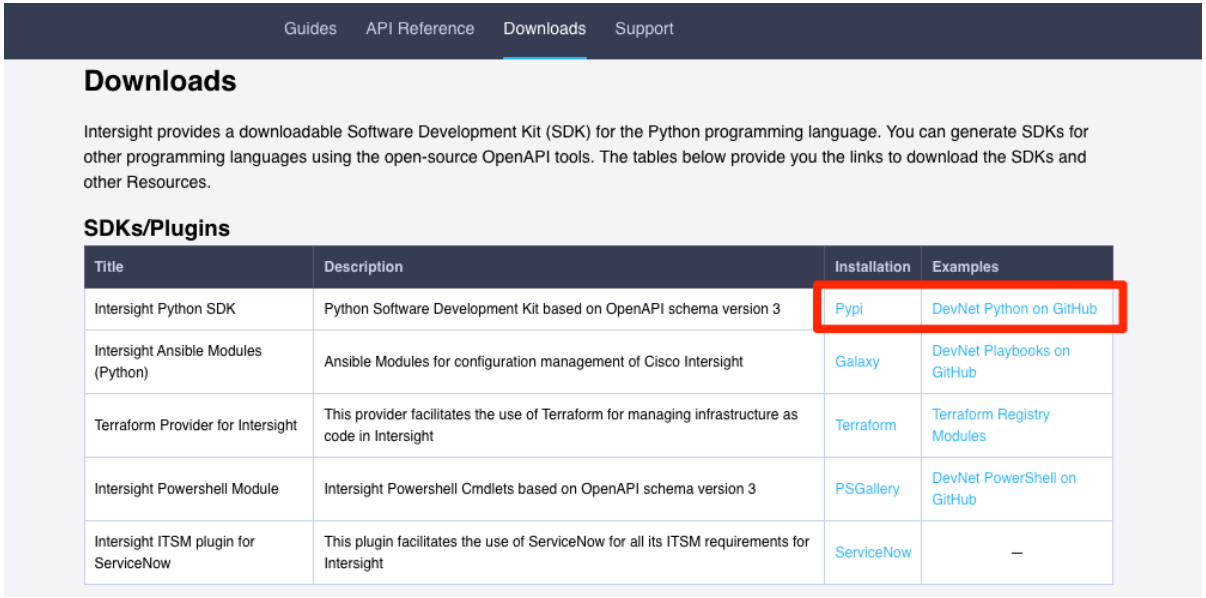
Requirements

- Knowledge of Python and Intersight
- Python Installed
- An Intersight API key and Secret
- Connectivity to intersight.com (or your CVA/PVAPP if you are running on-prem)

Documentation and examples:
https://github.com/CiscoDevNet/intersight-python

Python Examples described in this document:
https://github.com/rtortori/ico-python-examples

Install the Python SDK



Guides API Reference Downloads Support

Downloads

Intersight provides a downloadable Software Development Kit (SDK) for the Python programming language. You can generate SDKs for other programming languages using the open-source OpenAPI tools. The tables below provide you the links to download the SDKs and other Resources.

SDKs/Plugins

Title	Description	Installation	Examples
Intersight Python SDK	Python Software Development Kit based on OpenAPI schema version 3	Pypi	DevNet Python on GitHub
Intersight Ansible Modules (Python)	Ansible Modules for configuration management of Cisco Intersight	Galaxy	DevNet Playbooks on GitHub
Terraform Provider for Intersight	This provider facilitates the use of Terraform for managing infrastructure as code in Intersight	Terraform	Terraform Registry Modules
Intersight Powershell Module	Intersight Powershell Cmdlets based on OpenAPI schema version 3	PSGallery	DevNet PowerShell on GitHub
Intersight ITSM plugin for ServiceNow	This plugin facilitates the use of ServiceNow for all its ITSM requirements for Intersight	ServiceNow	–

https://intersight.com/apidocs/downloads/

Create a Virtual Environment (opt.) and install the Intersight Python Module

```
rtortori@laptop ICO-Python % python3 -m venv .venv
rtortori@laptop ICO-Python % source .venv/bin/activate
(.venv) rtortori@laptop ICO-Python %
```

```
(.venv) rtortori@laptop ICO-Python % pip install intersight
Collecting intersight
  Downloading intersight-1.0.9.5808.tar.gz (13.4 MB)
     |████████████████████████████████| 13.4 MB 6.2 MB/s
Collecting urllib3>=1.25.3
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 7.7 MB/s
[…]
    Running setup.py install for pycryptodome ... done
    Running setup.py install for intersight ... done
Successfully installed intersight-1.0.9.5808 pem-21.2.0 pycryptodome-3.14.1 python-dateutil-2.8.2 six-1.16.0
urllib3-1.26.9
(.venv) rtortori@laptop ICO-Python %
```

Working with ICO and the Python SDK

Clone the GitHub repo: https://github.com/rtortori/ico-python-examples
This repo includes code that can be used and reworked to address a number of use cases.

This document describes just a few of them, refer to the repository for additional info and instructions.

In order to authenticate to the Intersight API, in the repo is present a file called `credentials.py`
This Python module contains the necessary code to authenticate to Intersight, provided you have generated an API Key and Secret:
https://intersight.com/apidocs/introduction/security/%23generating-api-keys&sa=D&ust=1612024909729000&usg=AOvVaw362rkbFxqhX_Mo8w0xkDJG/

You will need to import that module in order to authenticate to the Intersight API.

The repo also contains a file named `env-linux-mac-example.sh`, which is a sample file you can take as a reference to understand how the `INTERSIGHT_API_KEY_ID` and `INTERSIGHT_API_PRIVATE_KEY` environment variables can be set. The credentials.py Python module will fetch the keys from those variables

Anatomy of an ICO Python Script (Sample)

```python
import credentials
from pprint import pprint
from intersight.api import workflow_api
from intersight.rest import ApiException

# Specify the MOID here
moid='6239aab7696f6e2d3118775d'

def get_tasks_definitions(api_client):
    """ Gets the list of tasks """
    api_instance = workflow_api.WorkflowApi(api_client)

    try:
        api_response = api_instance.get_workflow_task_definition_by_moid(moid=moid,_check_return_type=False)
        pprint(api_response)
    except ApiException as e:
        print("Exception when calling WorkflowApi->get_workflow_task_definition_list: %s\n" % e)

# Authenticate
api_client = credentials.config_credentials()

# Get the list of Tasks
get_tasks_definitions(api_client)
```

Import credential module to authenticate

Import workflow_api to work with ICO

Create an instance of the workflow_api, passing the credentials

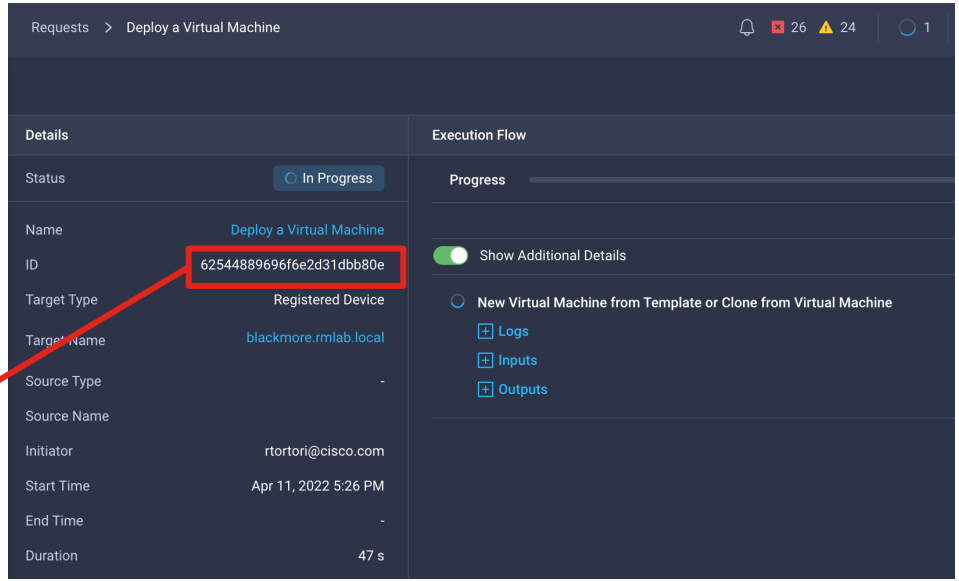
In this case, we use the get_workflow_task_definition_by_moid class as we want to get the task definition for a specific moid. We pass as arguments the moid as well as the optional _check_return_type set to False to get output in JSON format

We set api_client which will be passed to the get_tasks_definitions function in order to authenticate

We call the get_tasks_definitions function which will return the task definition in JSON format

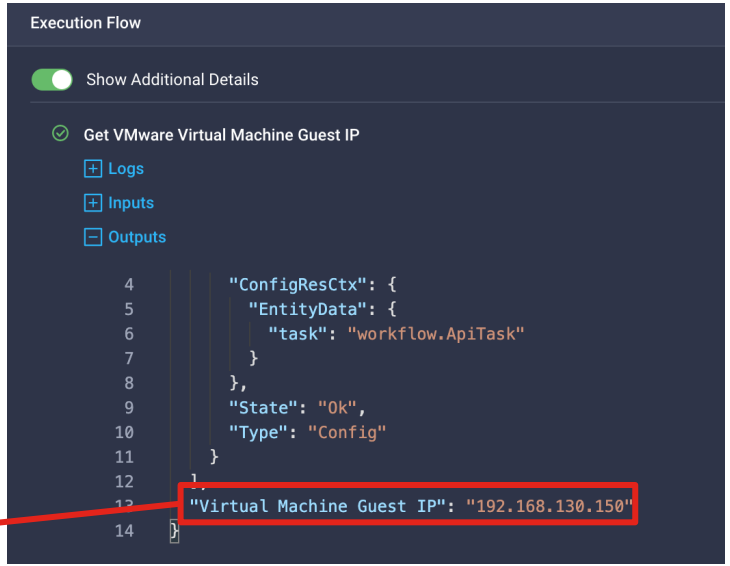
Example 1: Execute a Workflow

```
% source source.sh
% env | grep INTERSIGHT

INTERSIGHT_API_PRIVATE_KEY=/Path/To/secret.txt
INTERSIGHT_API_KEY_ID=xxxxxxxxxxxxxxxxxxxxxxxx

% python ico_wf_execute_by_name.py

{'account': None,
 'account_moid': '5f871f403361396c6a3154ae',
 'action': 'None',
 'ancestors': None,
[…]
 'workflow_task_count': 3,
 'workflow_worker_task_count': 2}

Execution Moid: 6254166b696f6e2d31db1e93
Executed by: rtortori@cisco.com
```



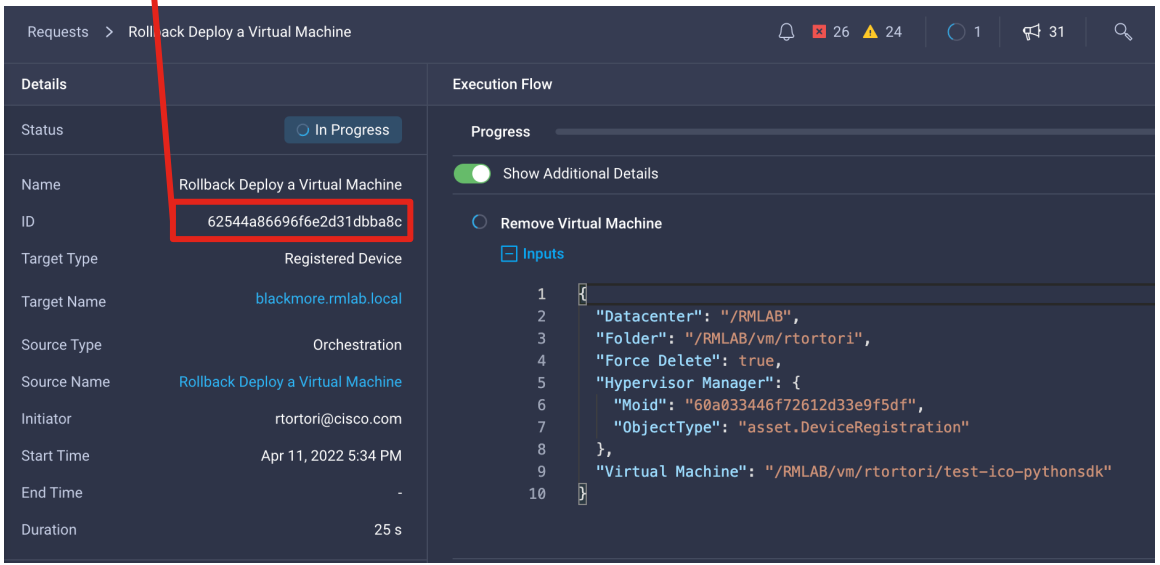
Example 2: Inspect Workflow Execution Outputs

```
% python ico_wf_get_request_tasks_outputs_by_moid.py

### Task New Virtual Machine from Template or Clone from Virtual
Machine Output ###

{'ConfigResults': [{'ConfigResCtx': {'EntityData': {'task':
'workflow.ApiTask'}, 'EntityMoid': '', 'EntityName': '',
'EntityType': ''}, 'Message': "Virtual Machine 'test-ico-pythonsdk'
created successfully.", 'MessageParams': None, 'OwnerId': '',
'State': 'Ok', 'Type': 'Config'}], 'Folder': '/RMLAB/vm/rtortori',
'Name': '/RMLAB/vm/rtortori/test-ico-pythonsdk', 'TaskId': 'task-
318756', 'Template': '/RMLAB/vm/rtortori/ubuntu-template-hx'}

### Task Get VMware Virtual Machine Guest IP Output ###

{'ConfigResults': [{'ConfigResCtx': {'EntityData': {'task':
'workflow.ApiTask'}, 'EntityMoid': '', 'EntityName': '',
'EntityType': ''}, 'Message': '', 'MessageParams': None, 'OwnerId':
'', 'State': 'Ok', 'Type': 'Config'}], 'vmip': '192.168.130.150'}
```



Example 3: Rollback a Workflow

```
% python ico_wf_rollback_by_moid.py
Rolling back Workflow with Moid 62544889696f6e2d31dbb80e
Rollback execution Moid: is 62544a86696f6e2d31dbba8c
Status: Running
```



Example 4: Get ICO Statistics

```
% python ico_statistics_get.py

### Workflow Designer Statistics ###

Total Workflows: 162
System Workflows: 29
User Workflows: 133
Valid/Invalid Workflows: 149/13

### Executions Statistics ###

Total Executions: 695
Success/Failed Executions: 434/261
Last Request: "Rollback Deploy a Virtual Machine", by user
rtortori@cisco.com, with Moid 62544a86696f6e2d31dbba8c

### Task Designer Statistics ###

Total Tasks: 229
System Tasks: 152
Custom Tasks: 77

### Data Types Statistics ###

Total Data Types: 228
System Data Types: 213
Custom Data Types: 15
```

Execute an ICO Workflow with Powershell SDK

Requirements

- Knowledge of Powershell and Intersight
- Powershell 7.1 or later
- Dotnet SDK 3.1 or later
- An Intersight API Key and Secret
- Connectivity to intersight.com (or your CVA/PVAPP if you are running on-prem)

Powershell Examples described in this document:
https://github.com/rtortori/ico-powershell-examples

Install and Configure Intersight Powershell Module

```
rtortori@laptop ~ % pwsh
PowerShell 7.2.2
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS /Users/rtortori> Install-Module -Name Intersight.PowerShell

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by
running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): Y
PS /Users/rtortori>
```

Working with ICO and the Powershell SDK

Refer to the repo: https://github.com/rtortori/ico-powershell-examples
This repo includes code that can be used and reworked to address a number of use cases.

This document describes just a few of them, refer to the repository for additional info and instructions.

In order to authenticate to the Intersight API, in the repo is present a file called `sample_configuration.ps1`
This powershell script contains the necessary code to authenticate to Intersight, provided you have generated an API Key and Secret:
https://intersight.com/apidocs/introduction/security/%23generating-api-keys&sa=D&ust=1612024909729000&usg=AOvVaw362rkbFxqhX_Mo8w0xkDJG/

You will need to either execute that script in your session or embed it in your powershell scripts in order to authenticate to the Intersight API.

Example 1: Execute and Rollback a Workflow

Invoke Execution

```
ICO > ./ico_wf_execution_by_name.ps1

Name                     Moid                     Email
----                     ----                     -----
Deploy a virtual machine 6254166b696f6e2d31db1e93 user@cisco.com
```

Rollback

```
ICO > ./ico_wf_rollback_by_moid.ps1 6254166b696f6e2d31db1e93

Moid                     Status
----                     ------
625416ec726f6e2d312e72ea Created


Moid                     Status
----                     ------
625416ec726f6e2d312e72ea Running
```

Example 2a: Get all Workflows Definitions

```
ICO > $filter="(Properties.ExternalMeta eq true) and (Name ne 'DeployHyperFlexSDWAN') and (DefaultVersion eq true)"
ICO > $wf = Get-IntersightWorkflowWorkflowDefinition -InlineCount allpages -Top 0 -Filter $filter
ICO > $wf

Count Results
----- -------
  522 {class WorkflowWorkflowDefinition {…

ICO > $wf.Results.Count
522

ICO > $wf.Results


ClassId               : WorkflowWorkflowDefinition
ObjectType            : WorkflowWorkflowDefinition
LicenseEntitlement    : Premier
DefaultVersion        : True
Description           : a test workflow to experiment with custom integrations
InputDefinition       : {class WorkflowBaseDataType {
                            class MoBaseComplexType {
                              ClassId: 0
                              ObjectType: 0
[...]
```

Intersight Powershell SDK cmdlets accept the parameter `-Filter`

This parameter accepts a filter query that can be used to reduce the scope of the search as shown in the example

Example 2b: Get all Workflows Definitions (Table View)

Sort in descending order by CreateTime

Only display Workflow Label, Moid and CreateTime

```
ICO > $wf.Results | Sort-Object CreateTime -Descending | Format-Table -Property Label,Moid,CreateTime

Label                                                Moid                     CreateTime
-----                                                ----                     ----------
testwf12                                             624f59c2696f6e2d316e9464 04/07/2022 21:38:11
testwfe01233                                         624ed76e696f6e2d316bc7b1 04/07/2022 12:22:06
Test Powershell Inspector                            624ec3b8696f6e2d316b95a0 04/07/2022 10:58:00
Powershell - MS Services                             624cb5ef696f6e2d31643927 04/05/2022 21:34:39
Configure on-premises FlexPod storage                624b4bbe696f6e2d3161fe0a 04/04/2022 19:49:18
Create Workspace                                     6244201a696f6e2d314bd343 03/30/2022 09:17:14
Deploy a Baremetal server                            623d2245696f6e2d30384732 03/25/2022 02:00:37
Demo - Detach a workspace from a Policy Set          623a0f8c696f6e2d3119fdc4 03/22/2022 18:03:56
Test - Loop in template                              6239cb92696f6e2d3119624c 03/22/2022 13:13:54
Demo - Create a Policy Set and add a workspace       6239b7c7696f6e2d3118ec22 03/22/2022 11:49:27
Demo - SSH Executor - Custom Tasks                   6239adbe696f6e2d311886b0 03/22/2022 11:06:39
Create a new Policy Set                              6239abf6696f6e2d3118798f 03/22/2022 10:59:02
Variables Demo - Math                                6234a4d3696f6e2d310dabd4 03/18/2022 15:27:15
New Virtual Machine Network on Multiple Hosts        62345857696f6e2d310c47d7 03/18/2022 10:00:55
Workflow Variables Demo - With Vars                  62320c11696f6e2d31cb5611 03/16/2022 16:10:57
HTTP Dump                                            622b4a0d696f6e2d31c0b44d 03/11/2022 13:09:34
```

The bridge to possible