Cisco Crosswork Workflow Manager 2.0 Operator Guide

First Published: 2025-04-25

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

UI overview

This section covers the following topics:

UI overview

The Cisco CWM user interface allows you to access all of the product's core functions. You can access them through four main menu selection paths:

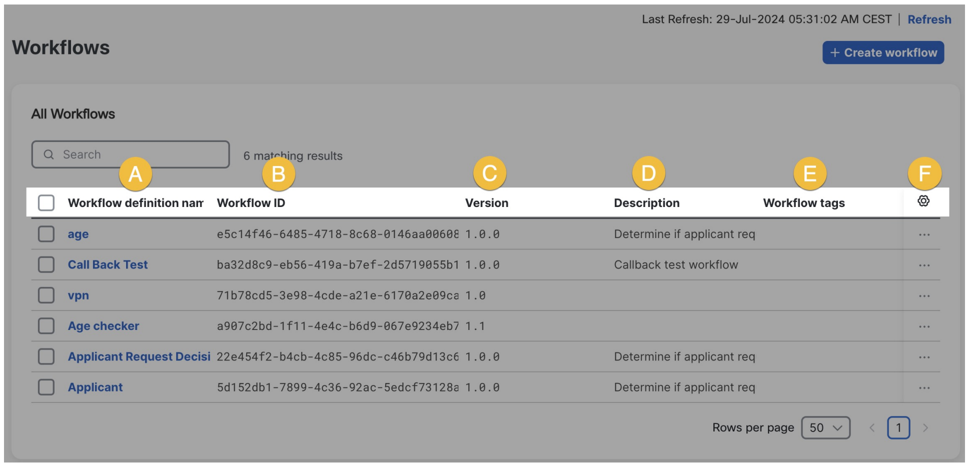
- **Dashboard**: Add customizable dashboards with dashlets that monitor the status of many CNC functions, including CWM workflows, adapters, workers, and active tasks.
- **Design**: Add, manage, and delete workflow definitions, and forms.
- **Operate**: Run new workflows, check the status of particular workflow executions (jobs), view schedules, or manage manual tasks that are triggered by workflow execution callback states.
- **Administration** > **Workflow Administration**: Add and manage workers, adapters, secrets, resources, event types, and system settings.

Workflows

The main **Workflows** view consists of a list of workflows added to CWM with details and possible actions.

Workflows table

Figure 1: All Workflows tab

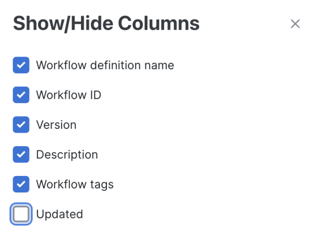


Workflow definition nam	Workflow ID	Version	Description	Workflow tags	
age	e5c14f46-6485-4718-8c68-0146aa00608	1.0.0	Determine if applicant req		···
Call Back Test	ba32d8c9-eb56-419a-b7ef-2d5719055b1	1.0.0	Callback test workflow		···
vpn	71b78cd5-3e98-4cde-a21e-6170a2e09ca	1.0			···
Age checker	a907c2bd-1f11-4e4c-b6d9-067e9234eb7	1.1			···
Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Determine if applicant req		···
Applicant	5d152db1-7899-4c36-92ac-5edcf73128a	1.0.0	Determine if applicant req		···

The **Workflows** table consists of the following columns:

- A) **Workflow definition name**: unique name of the workflow.
- B) **Workflow ID**: unique identifier assigned automatically after workflow creation.
- D) **Version**: indicates the current workflow version number, allows keeping track of the workflow versioning.
- C) **Description**: optional field used for describing your workflow functionality.
- E) **Workflow tags**: shows tags assigned to a given workflow.
- F) **Actions**: shows possible actions for a given workflow:
  - **Run**: starts a single execution of a given workflow (job).
  - **Delete**: removes an added workflow from CWM (this action is irreversible).

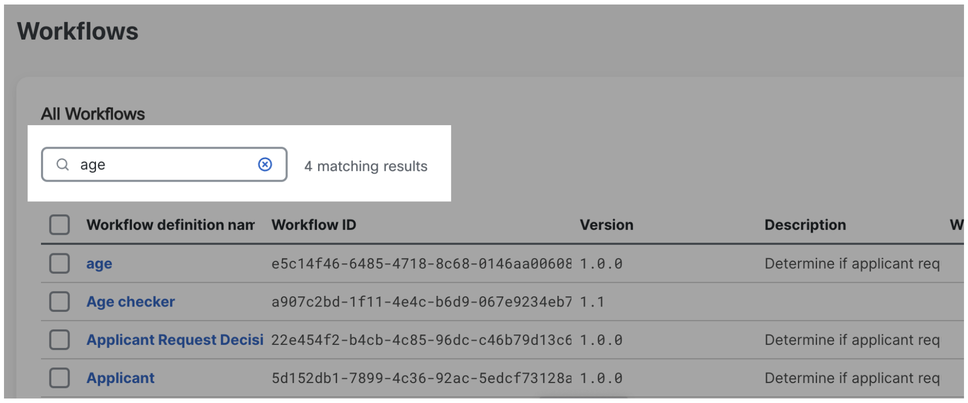
Figure 2: Gear icon



- ✓ Workflow definition name
- ✓ Workflow ID
- ✓ Version
- ✓ Description
- ✓ Workflow tags
- ☐ Updated

To show/hide the columns displayed in the table, use the gear icon on the right side above the table.

Figure 3: Filter option



Workflows

All Workflows

age — 4 matching results

Workflow definition nam	Workflow ID	Version	Description	W
age	e5c14f46-6485-4718-8c68-0146aa00608	1.0.0	Determine if applicant req	
Age checker	a907c2bd-1f11-4e4c-b6d9-067e9234eb7	1.1		
Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Determine if applicant req	
Applicant	5d152db1-7899-4c36-92ac-5edcf73128a	1.0.0	Determine if applicant req	

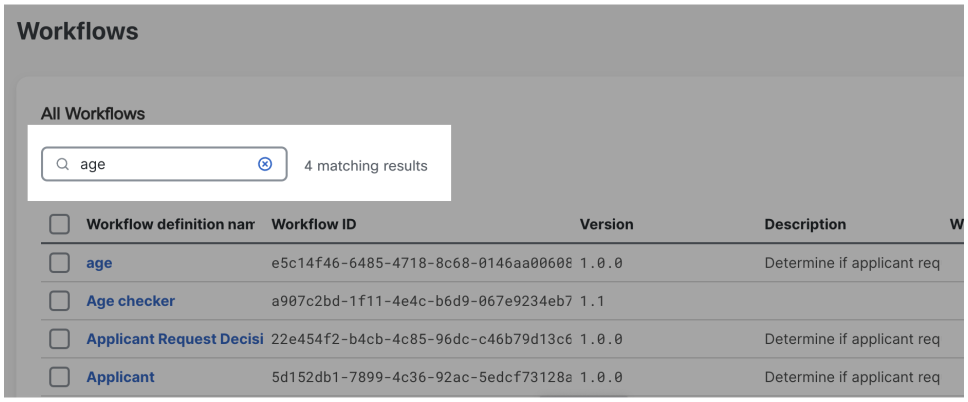
To search the table for the specified value(s), provide your input in the **Search** field above the table. The search is performed in all the columns, whether they are currently displayed in the table .

Bulk actions on workflows

You can perform bulk actions on multiple workflows at once. To select a workflow, tick the checkbox on the right-side of the **All Workflows** table. You have three possible bulk actions:

- **Edit tags**: modifies tags for selected workflows by adding one or more or resetting them.
- **Delete Selected**: removes selected workflows from the list and CWM (this action is irreversible).
- **Export**: exports selected workflow entries to a CSV file.

Figure 4: Bulk actions



Workflows

All Workflows

age — 4 matching results

Workflow definition nam	Workflow ID	Version	Description	W
age	e5c14f46-6485-4718-8c68-0146aa00608	1.0.0	Determine if applicant req	
Age checker	a907c2bd-1f11-4e4c-b6d9-067e9234eb7	1.1		
Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Determine if applicant req	
Applicant	5d152db1-7899-4c36-92ac-5edcf73128a	1.0.0	Determine if applicant req	

Jobs

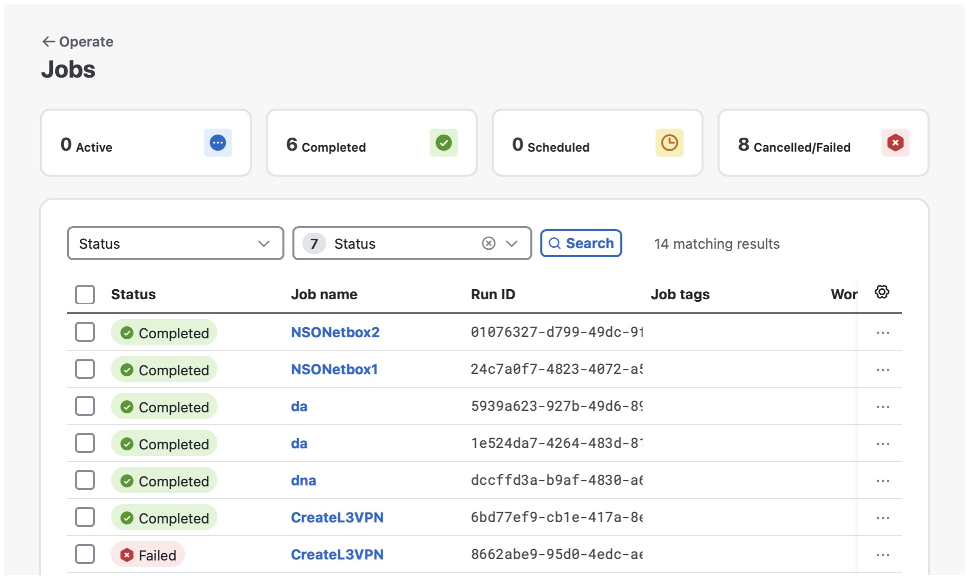
The **Jobs** view lets you monitor job statuses and rerun or cancel jobs (workflow executions).

Job Tabs

At the top of the **Jobs** view, status tabs display an overview of the number of jobs and serve as status filters.

There are four available status tabs:

Figure 5: Jobs status tabs



← Operate

Jobs

0 Active | 6 Completed | 0 Scheduled | 8 Cancelled/Failed

Status | 7 Status | Search | 14 matching results

Status	Job name	Run ID	Job tags	Wor
Completed	NSONetbox2	01076327-d799-49dc-91		
Completed	NSONetbox1	24c7a0f7-4823-4072-a5		
Completed	da	5939a623-927b-49d6-89		
Completed	da	1e524da7-4264-483d-81		
Completed	dna	dccffd3a-b9af-4830-a6		
Completed	CreateL3VPN	6bd77ef9-cb1e-417a-8e		
Failed	CreateL3VPN	8662abe9-95d0-4edc-ae		

1. **Active**: The number of jobs currently running.
2. **Completed**: The number of jobs that have completed successfully.
3. **Scheduled**: The number of jobs currently scheduled to run.
4. **Cancelled/Failed**: The number of jobs that were manually terminated by users or that failed during execution.

To see all the jobs with a particular status, click on the appropriate status tab. For example, to see all the active jobs, click on the **Active** status tab. The Active jobs will then be shown in the table below the tabs.

The **Scheduled** status tab will display only scheduled jobs and their details in the table. You may also find particular jobs on a given schedule under other tabs, according to their completion status (for example, a **Scheduled** job may also be **Active** or **Completed**).

The **Jobs** view shows all jobs (workflow executions) run in the last 24 hours. You can also use the filter table option to search the table for specified values.

While filtering based on tags, you can filter on only one tag at once. You can filter on multiple job tags only while using the API.

Bulk actions on jobs

The Jobs view enables you to perform bulk actions on many jobs at once.

To select one or more jobs, first click one of the four job status tabs, then click the checkbox next to the jobs you want. Then click the more icon (…) on the right side of the table.

The actions available to you will vary depending on the job status tab you have currently selected:

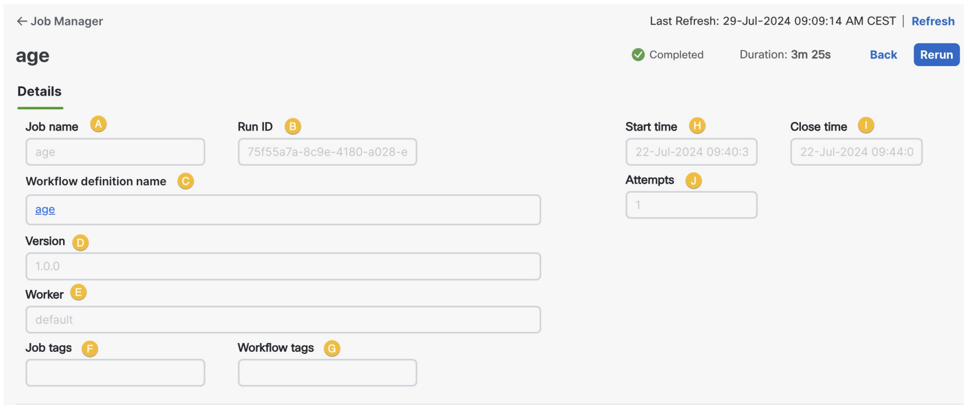
- **Export Selected**: Exports all the selected job entry details to a CSV file. Available for **Active**, **Completed**, **Scheduled**, and **Cancelled/Failed** jobs..

- **Cancel Selected**: Terminates the selected jobs that are currently running. Available only for **Active** jobs.
- **Delete Selected**: Deletes the selected scheduled jobs from the list and from the database. Available only for **Scheduled** jobs; this action is irreversible.

Job details

To check the details of a job, in the Job Manager choose the appropriate tab (**Active**, **Completed**, **Cancelled**). In the table, click on the name of the job entry you want to check.

Figure 6: Details tab



Last Refresh: 29-Jul-2024 09:09:14 AM CEST

age

Duration: 3m 25s

Job name	age
Run ID	75f55a7a-8c9e-4180-a028-e
Workflow definition name	age
Version	1.0.0
Worker	default
Job tags	
Workflow tags	
Start time	22-Jul-2024 09:40:3
Close time	22-Jul-2024 09:44:0
Attempts	1

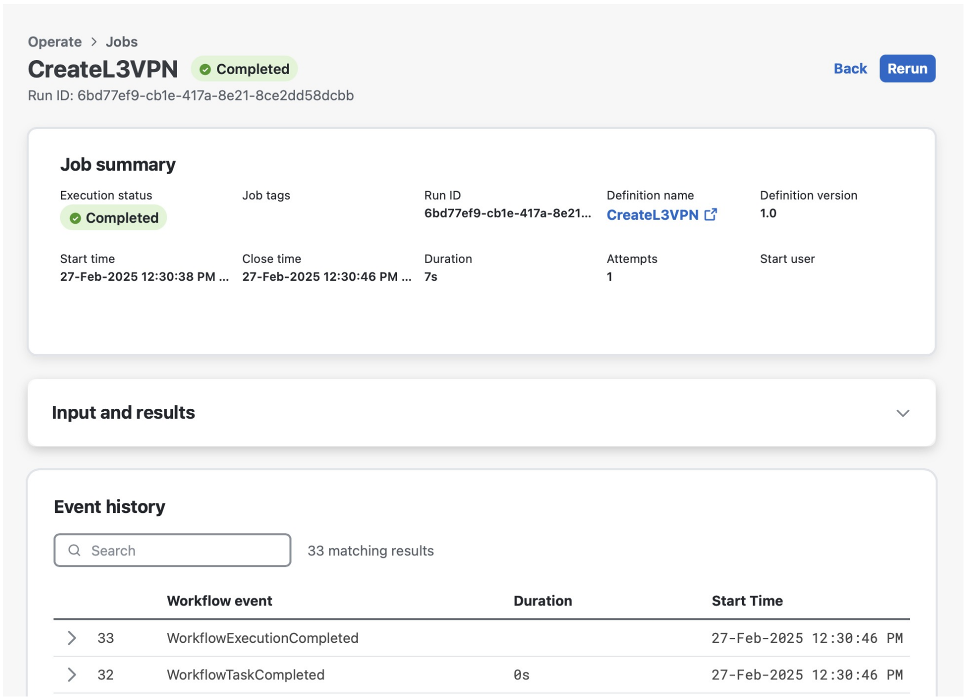
The **Details** tab lets you check the following data for the selected job:

- **Job name**: name of the particular workflow execution; by default, it is the workflow definition name.
- **Run ID**: unique identifier for workflow execution.
- **Workflow definition name**: name of the workflow definition used to run a particular job.
- **Version**: version of the executed workflow definition.
- **Worker**: worker assigned to the workflow definition.
- **Job tags**: keyword(s) assigned to the job that allows filtering/sorting jobs based on them.
- **Workflow tags**: keyword(s) assigned to the workflow definition used to run a particular job.
- **Start time**: date and time when workflow execution started, for example 05-May-2023 02:59:25 PM CEST.
- **Close time**: date and time when workflow execution ended, for example 05-May-2023 03:59:25 PM CEST. If the execution is still running or it was failed/canceled, the appropriate status is displayed instead.
- **Attempts**: indicates the number of tries needed to execute the workflow successfully; the maximum number of attempts can be limited by the workflow definition.
- The **Parent Run ID** field may be displayed if your job was executed as a child workflow.

Job summary, results and event history

The **Job summary** presents the details of the execution of a particular job, along with the workflow input and final result. It also lists the series of events that the workflow engine has performed while executing the workflow definition.

Figure 7: Job Event Log



Run ID: 6bd77ef9-cb1e-417a-8e21-8ce2dd58dcbb

Job summary

Execution status	Job tags	Run ID	Definition name	Definition version
Completed		6bd77ef9-cb1e-417a-8e21...	CreateL3VPN	1.0

Start time	Close time	Duration	Attempts	Start user
27-Feb-2025 12:30:38 PM ...	27-Feb-2025 12:30:46 PM ...	7s	1	

	Workflow event	Duration	Start Time
33	WorkflowExecutionCompleted		27-Feb-2025 12:30:46 PM
32	WorkflowTaskCompleted	0s	27-Feb-2025 12:30:46 PM

The **Job summary** panel displays the following data:

- **Execution status**: event execution status.
- **Start Time**: date and time when job execution started, for example `05-Jan-2025 03:59:25 PM CEST`
- **Job tags**: lists tags attributed to this particular job when running the workflow.
- **Close Time**: date and time when job execution ended.
- **Run ID**: unique ID of this particular job run.
- **Duration**: total time of the job run from start time to close time.
- **Definition Name**: name of the workflow definition used for this job run.
- **Attempts**: number of attempts performed for completion of the task.
- **Definition version**: the version of the workflow definition used for this job run.
- **Start user**: name of the user performing the job run.

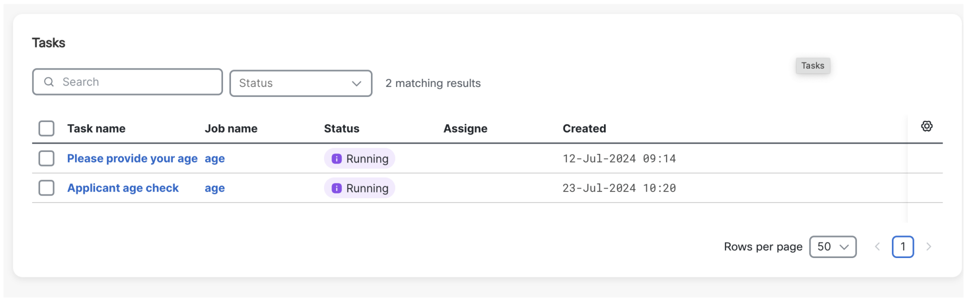
The **Input and results** panel presents the input data (on the left) provided on the workflow definition run and the final output data returned from by the workflow upon completion of the workflow events.

Event history is a comprehensive, ordered log of all Events generated during a workflow execution. These events are produced in response to external occurrences and actions issued by the workflow. The Event history serves as the definitive record of a workflow lifecycle, capturing every state transition and action taken.

Tasks

Tasks allow operators to intervene manually during workflow execution. You configure tasks using the **Forms** view in the **Design** menu, and refer to them as part of the **Callback state** inside the workflow definitions.

The Task List displays tasks that are currently pending for input by the operator.



Task name	Job name	Status	Assigne	Created
Please provide your age	age	Running		12-Jul-2024 09:14
Applicant age check	age	Running		23-Jul-2024 10:20

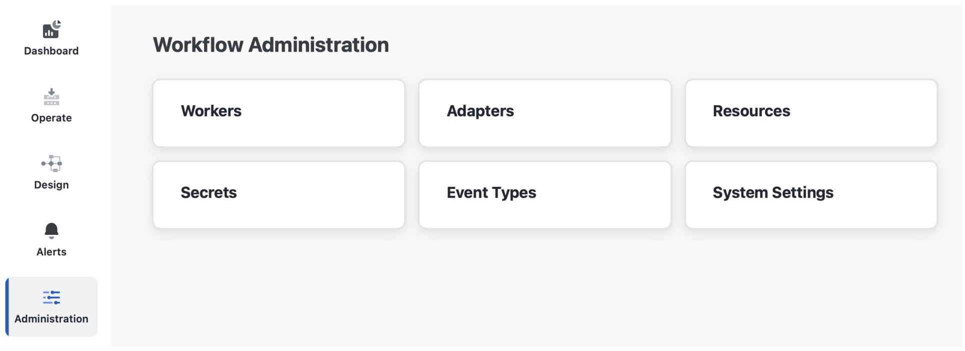
To use the task list:

1. Click on a task name to see the task prompt and provide input.
2. Use **Search** to find a specific task or the **Status** filter to filter out tasks according to their status.
3. Select task(s) to export them to a CSV format.

Workflow Administration

The **Workflow Administration** view allows you to manage adapters, workers, secrets, resources, event types and system settings through dedicated tabs.

Figure 8: Workflow Administration view

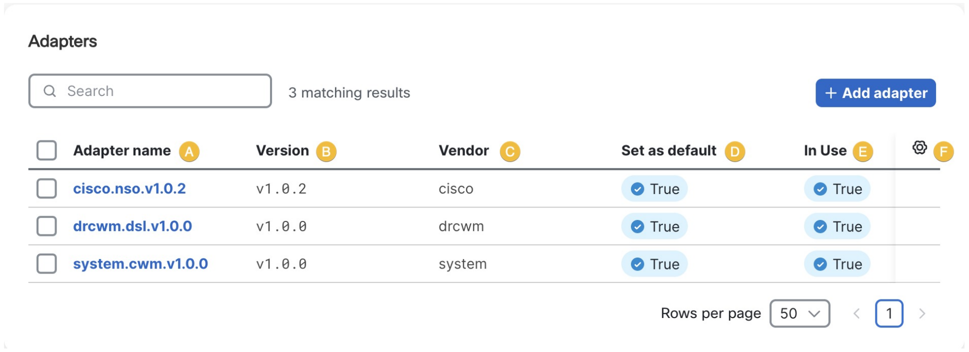


Adapters

The **Adapters** tab displays a table listing all uploaded adapters. Above the table, there is the **Add adapter** action button that allows you to upload adapter to CWM and and install it.

To export all details of selected adapter(s) to a CSV file, check them in the list and click the **Export** action button.

Figure 9: Adapters table



Adapter name A	Version B	Vendor C	Set as default D	In Use E	F
cisco.nso.v1.0.2	v1.0.2	cisco	True	True	
drcwm.dsl.v1.0.0	v1.0.0	drcwm	True	True	
system.cwm.v1.0.0	v1.0.0	system	True	True	

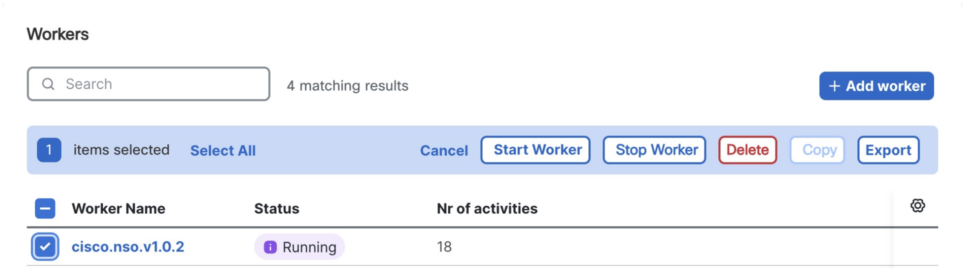
The **Adapters** table has the following columns:

- A) **Adapter name**: unique adapter name.
- B) **Version**: version of the added adapter.
- C) **Vendor**: vendor name, as specified in the adapter properties.
- D) **Set as default**: in this column, the adapter can have one of two possible statuses, depending on whether it is set as a default adapter to be used by workflow.
  - **True**: indicates the adapter is the default one for accociated activities.
  - **False**: indicates the adapter is not set as the default one.
- E) **In Use**: in this column, the adapter can have one of two possible statuses, depending on whether it is currently in use, which determines if it can be deleted.
  - **True**: indicates the adapter is currently in use by one or more workflows and therefore cannot be deleted.
  - **False**: indicates the adapter is not in use by any workflow and can be deleted.
- F) **Actions**:
  - **Delete**: deletes the uploaded adapter from CWM.
  - **Install** installs the already uploaded adapter to CWM.

Workers

The **Workers** tab displays a table listing all existing workers.

Figure 10: Workers action buttons



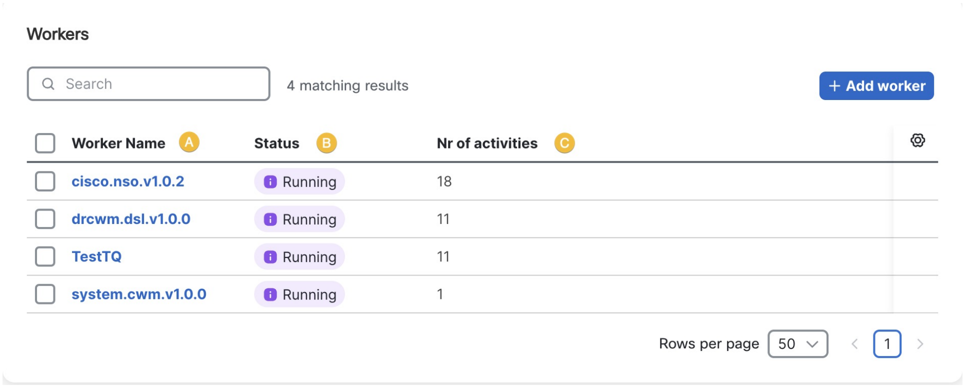
Worker Name	Status	Nr of activities
cisco.nso.v1.0.2	Running	18

When you check worker(s) in the table, various action buttons appear:

- **Add Worker**: allows you to create a new worker and assign to it activities from already installed adapters.
- **Start Worker**: starting the worker allows it to execute assigned tasks and changes the worker's status to running.
- **Stop Worker**: stopping the worker allows it to execute pending actions but blocks its ability to accept new tasks. After clicking on the Stop Worker button, the confirmation modal Stop worker(s) appears. If the worker actively executes an ongoing task, you won't be able to stop it, unless you tick the checkbox Force stop. This action stops the worker regarding whether it performs any task and might cause the failure of ongoing jobs.
- **Delete**: deletes the worker permanently (this action is irreversible). If the worker is running, it's force stopped before deleting.
- **Export selected**: downloads the CSV file with the details of the selected workers, including worker name, status and nr of activities.

The **Workers** table has the following columns:

Figure 11: Workers table



Worker Name	Status	Nr of activities
cisco.nso.v1.0.2	Running	18
drcwm.dsl.v1.0.0	Running	11
TestTQ	Running	11
system.cwm.v1.0.0	Running	1

- A) **Worker Name**: unique worker's name. Once created, it cannot be modified.
- B) **Status**: worker has three possible statuses visible in the CWM UI:
    - **Initialized**: worker status after creation, if you don't choose to start it immediately.
    - **Running**: indicates started worker available to execute tasks.
    - **Stopped**: indicates stopped worker unavailable for task execution.

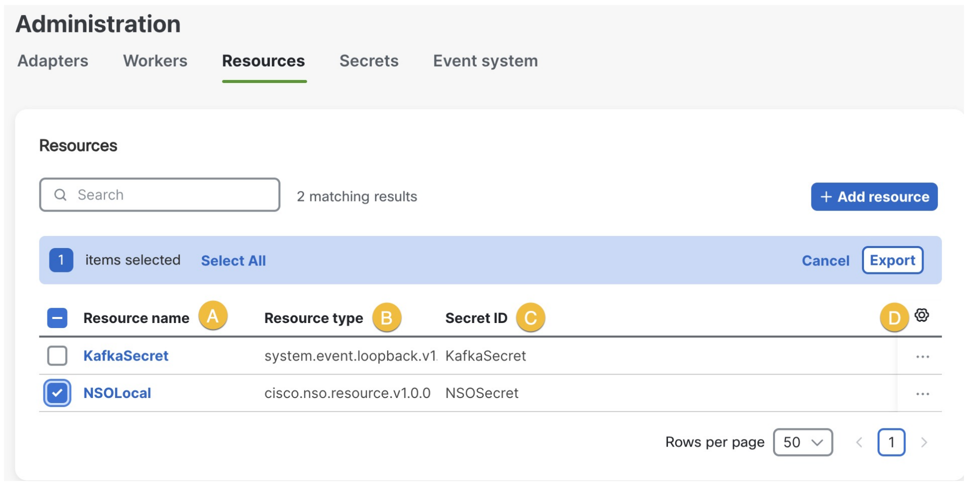
- C) **Nr of activities**: shows how many adapters' activities are assigned to the worker.

Resources

The **Resources** tab displays a table listing all existing resources. Above the table, there are various action buttons:

- **Add Resource**: allows you to add resource to CWM.
- **Export**: downloads the CSV file with all details of selected resources.

Figure 12: Resources table



Resource name	Resource type	Secret ID
KafkaSecret	system.event.loopback.v1	KafkaSecret
NSOLocal	cisco.nso.resource.v1.0.0	NSOSecret

The **Resources** table has the following columns:

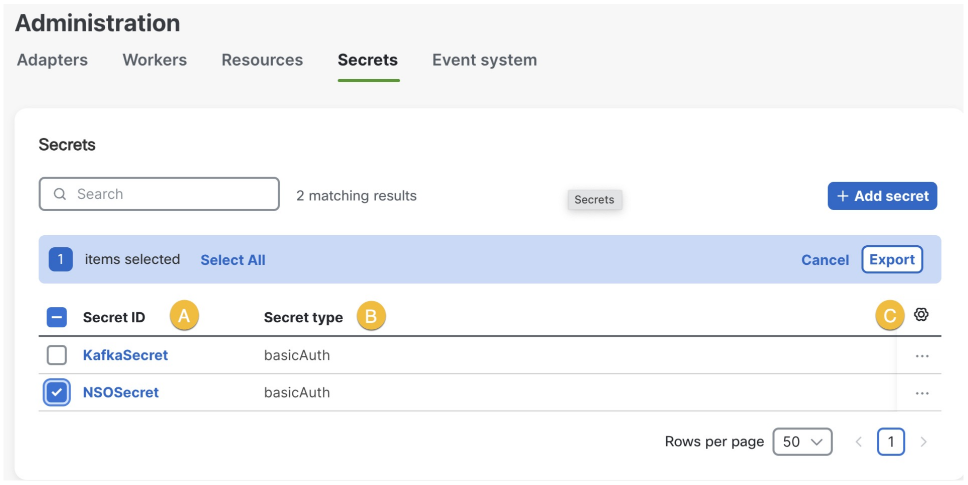
- A) **Resource name**: unique resource name.
- B) **Resource type**: resource type specified in the adapter files.
- C) **Secret ID**: associated secret ID.
- D) **Actions**:
  - **Delete**: removes resource from CWM

Secrets

The **Secrets** tab displays a table listing all existing secrets. Above the table, there are various action buttons:

- **Add Secret**: allows you to add secret to CWM.
- **Export**: downloads the CSV file with all details of selected secrets.

Figure 13: Secrets table



Secret ID	Secret type
KafkaSecret	basicAuth
NSOSecret	basicAuth

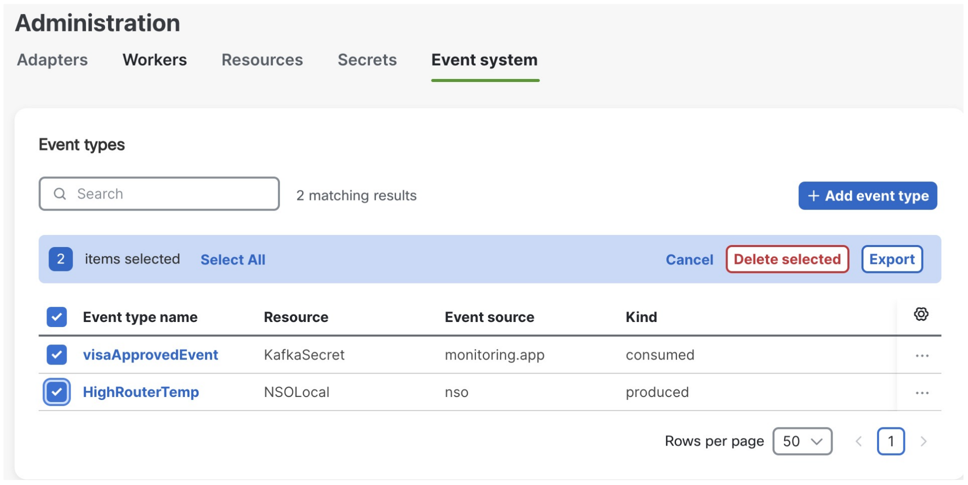
The **Secrets** table has the following columns:

- A) **Secret ID**: unique secret name.
- B) **Secret type**: secret type specified in the adapter files.
- C) **Actions**:
  - **Delete**: removes secret

Event system

In the **Event system** tab, you define Kafka events that your workflows will consume or produce.

Figure 14: Event system



Event type name	Resource	Event source	Kind
visaApprovedEvent	KafkaSecret	monitoring.app	consumed
HighRouterTemp	NSOLocal	nso	produced

- **Add event type**: allows you to define a new Kafka (or other) event in CWM. They can be of either `consume` or `produce` kind.

- **Delete Selected**: deletes selected event types from the list and from CWM (this action is irreversible).
- **Export**: downloads the CSV file with all details of the selected event types.

System Settings

With the **System settings** tab, you can manage job retention, retry policies, and timeouts. It controls how long completed jobs are kept, retry behavior (delay, jitter, max attempts), and various timeout settings for actions, events, states, system activities, and workflows.

CHAPTER 2

Initial setup

This section covers the following topics:

Initial setup

This section provides instructions to guide you through your first steps in using CWM. This initial setup allows you to prepare your workplace and start executing workflows.

Use these guidelines to accomplish the following:

Procedure

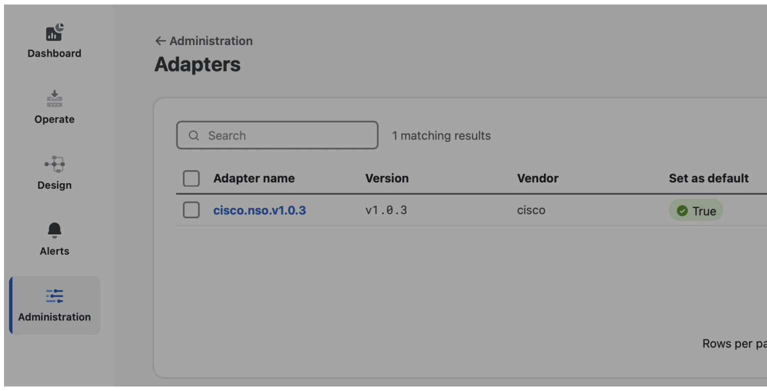
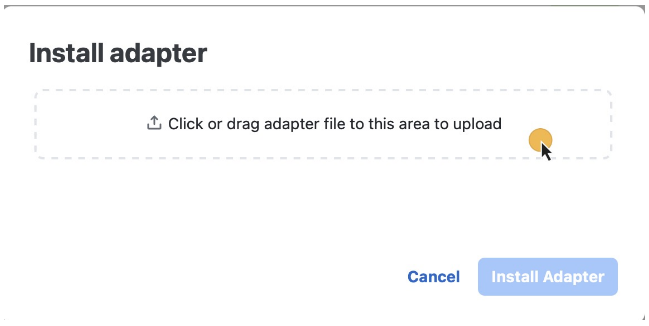
	Command or Action	Purpose
Step 1	**Install adapters and create workers**.	First, you need to add and install adapters that allow workflows to interact with external systems. In this guide, it's assumed that you will use the default adapters shipped with CWM, or that you have custom adapters developed and ready to use. For execution of your adapter activities and code, you need workers. You can create a worker directly during adapter installation.
Step 2	**Configure secrets and resources**.	Adapters mostly communicate with external entities via APIs, and they need to have connection data, like port or host, and authentication data, like username with password. This information is kept in secrets and resources. The structure and schema for each resource and secret type are defined in the adapter files. As an operator, you can add secrets and resources to CWM. Once created, they can be reused by different workflow definitions and passed on when needed.
Step 3	**Add your workflow definition**.	Workflow definitions, often referred to as workflows, are pieces of code intended to automate operations. They are written in JSON or YAML and follow the Serverless

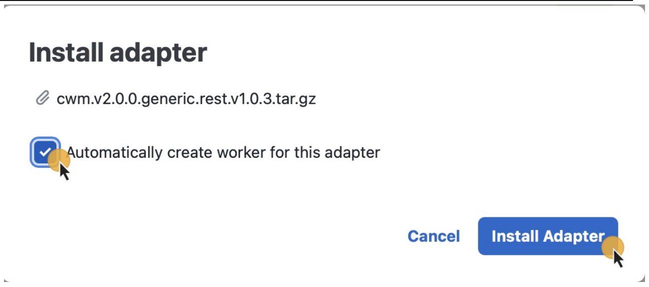
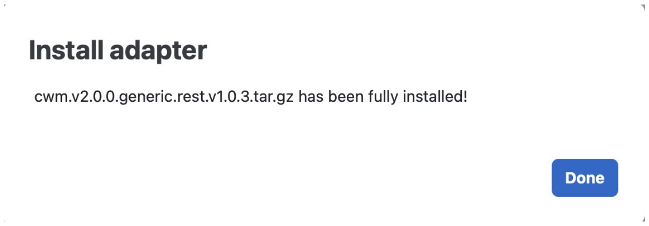
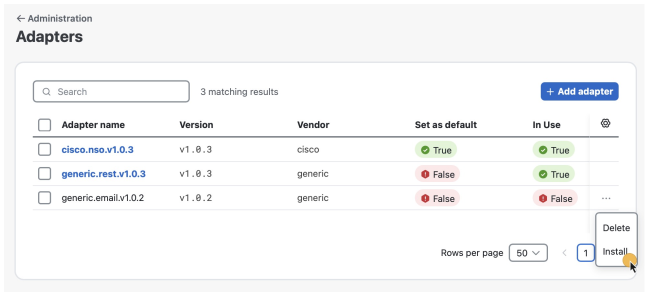
	Command or Action	Purpose
		Workflow Specification. Before workflows can be executed, they must be to be added to CWM. When you complete this setup you are ready to run your first job.

Install an adapter

Follow these steps to upload an adapter file and then install (deploy) it.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Adapters**.	
Step 2	Click the **Add adapter** button.	 Dashboard Operate Design Alerts Administration ← Administration Adapters Search Rows per pa 1 matching results Adapter name / Version / Vendor / Set as default cisco.nso.v1.0.3 / v1.0.3 / cisco / True
Step 3	In the **Install adapter** window, click on or drag an installable adapter file to the file uploader area and then click **Install Adapter**. You can find adapter files in the CWM software package.	**Note** The only supported installable adapter file format is tar.gz.  Install adapter Click or drag adapter file to this area to upload Cancel Install Adapter
Step 4	After the adapter file is uploaded to the CWM database, optionally tick **Automatically create worker for this adapter** checkbox if you want to create one, then click **Install Adapter** to finish the installation process.	If the installation is interrupted or you chose **Cancel**, the adapter will be added to the CWM database and visible in the adapters list, but will not be installed and will be unusable by workflows. If you canceled the installation intentionally and want to remove the adapter from the list, click **Delete** under the Actions column.

	Command or Action	Purpose
		 **Install adapter** cwm.v2.0.0.generic.rest.v1.0.3.tar.gz ☑ Automatically create worker for this adapter
Step 5	If your adapter was installed, you should see the message "*Your_adapter's_name* has been fully installed!". Click **Done**.	 **Install adapter** cwm.v2.0.0.generic.rest.v1.0.3.tar.gz has been fully installed! **Adapter not installed**: If the process failed or was interrupted, you can finalize it later by clicking the **Install** button under the Actions column. 

← Administration

Adapters

Search — 3 matching results

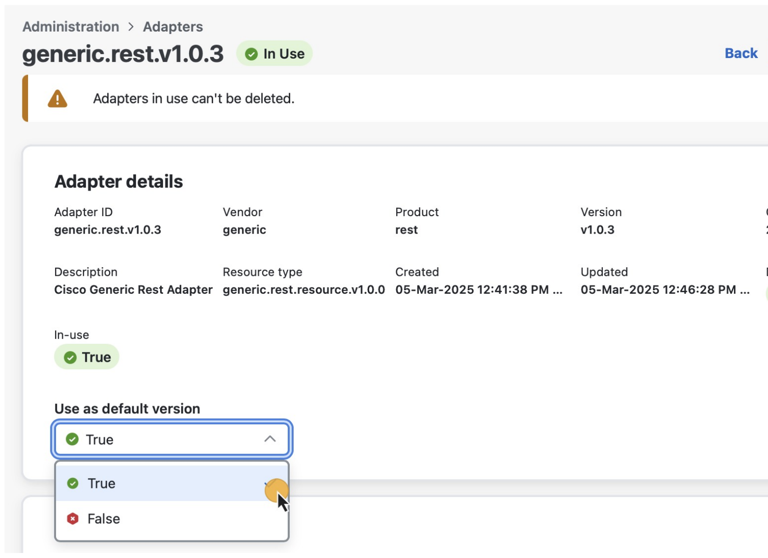
Adapter name	Version	Vendor	Set as default	In Use
cisco.nso.v1.0.3	v1.0.3	cisco	True	True
generic.rest.v1.0.3	v1.0.3	generic	False	True
generic.email.v1.0.2	v1.0.2	generic	False	False

Set adapter as default for associated activities

You can set which adapter should be the default one for associated activities. To do so, follow the steps below.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Adapters**	
Step 2	In the **Adapters** list, click the name of the chosen adapter to open the adapter's details view.	

	Command or Action	Purpose
Step 3	In **Use as default version**, select **True**.	 Administration > Adapters generic.rest.v1.0.3 In Use Back Adapters in use can't be deleted. Adapter details Adapter ID: generic.rest.v1.0.3 Vendor: generic Product: rest Version: v1.0.3 Description: Cisco Generic Rest Adapter Resource type: generic.rest.resource.v1.0.0 Created: 05-Mar-2025 12:41:38 PM ... Updated: 05-Mar-2025 12:46:28 PM ... In-use: True Use as default version: True True False
Step 4	Click **Save Changes**.	After setting the adapter as the default one, you should see a relevant status under **Set as default** column on the adapters list.

Delete adapter

You can delete an adapter if it has the status *Not in use* under the **In Use** column in the adapters list.

You cannot delete an adapter that is in use by one or more running workflows. The adapter will be shown with an *In use* status and the Delete button will be disabled. To delete the adapter, you must first terminate the running workflows and remove the associated workers.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Adapters**.	
Step 2	In the Adapters list, find the adapter you want to remove and click the **Delete** button in the same row under the Actions column.	
Step 3	In the **Delete Adapter** window, confirm by clicking the **Delete Adapter** button.	

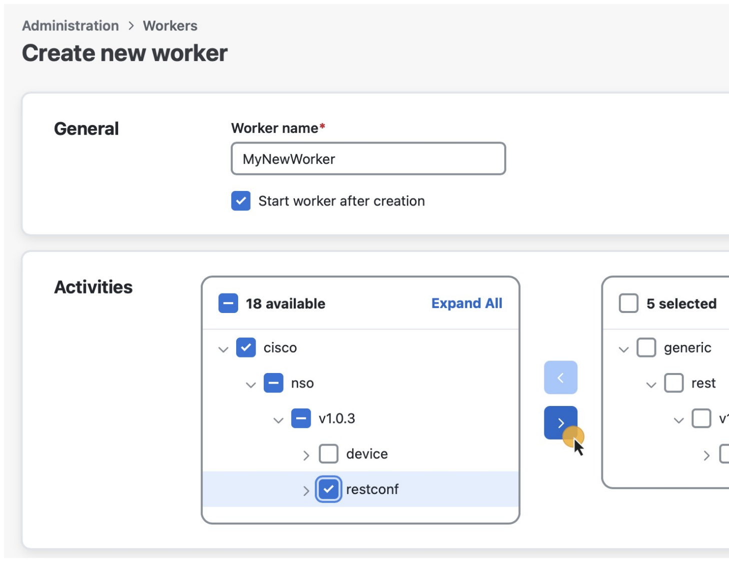
Add worker

This topic explains how to create a worker in CWM.

Before you begin

Be sure to add and deploy one or more adapters.

Procedure

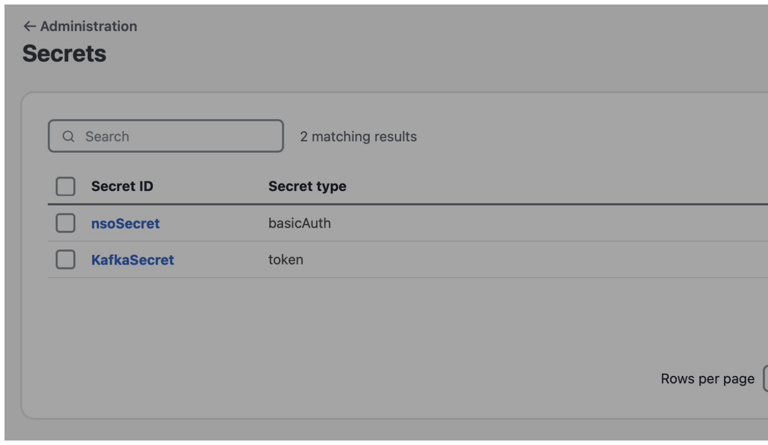
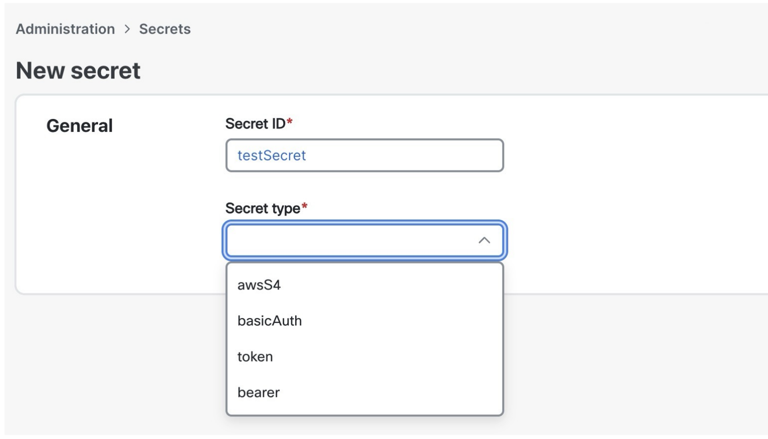
	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Workers**.	
Step 2	In the **Workers** view, click **Add worker**.	
Step 3	In the **Create new worker** window, provide the required input:	 • **Worker name**: Type a name for your worker; for example, `MyNewWorker`. Once created, it cannot be modified. • **Start worker after creation**: Check the checkbox to automatically start the worker as soon as you click **Create worker**. • **Activities**: Check the checkboxes for selected adapter activities you want to assign to your worker, then click the right-pointing arrow to and move them to the **selected** field . In our example, a worker for the Generic Rest adapter's activities is created. If you want to remove any of the previously selected activities from the worker, check the checkbox next to each selected activity you want to remove and then use the left-pointing arrow to remove it.
Step 4	Click **Create worker**.	

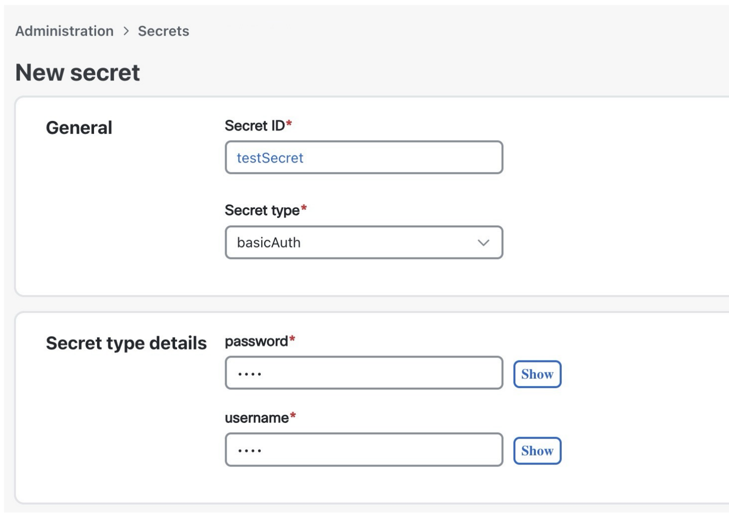
Create secrets and resources

The following topics explain how to add secrets and resources to CWM. Be sure to upload adapters with workers, and add secrets before adding resources, as resources use secrets.

Add a secret

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Secrets**	
Step 2	In the **Secrets** view, click **Add secret**.	 ← Administration Secrets Search 2 matching results Secret ID | Secret type nsoSecret | basicAuth KafkaSecret | token Rows per page
Step 3	In the **New secret** view, provide the required input:	 Administration > Secrets New secret General Secret ID* testSecret Secret type* awsS4 basicAuth token bearer • **Secret ID**: Enter a unique name for your secret. • **Secret type**: Select a **Secret type** from the drop-down menu. Secret types are defined in adapters and are added during adapter installation.

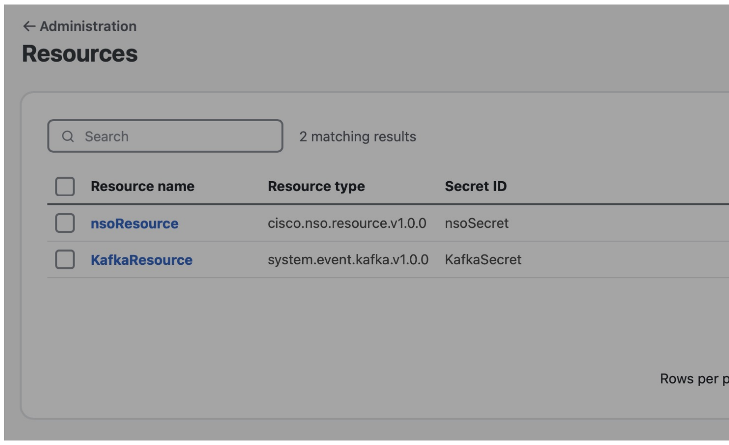
	Command or Action	Purpose
Step 4	After selecting the secret type, CWM displays a set of additional fields (for example, **Username** and **Password**) in the **Secret type details** section. The adapter specifies these detail fields, and they will vary with the adapter and secret type. Fill in the fields with values, as appropriate.	 Administration > Secrets; New secret; General: Secret ID* testSecret; Secret type* basicAuth; Secret type details: password* ····; username* ····
Step 5	Click **Create Secret**.	Once you've added a secret and its details, create a resource in CWM and associate it with the secret, as explained in Add a resource, on page 19.

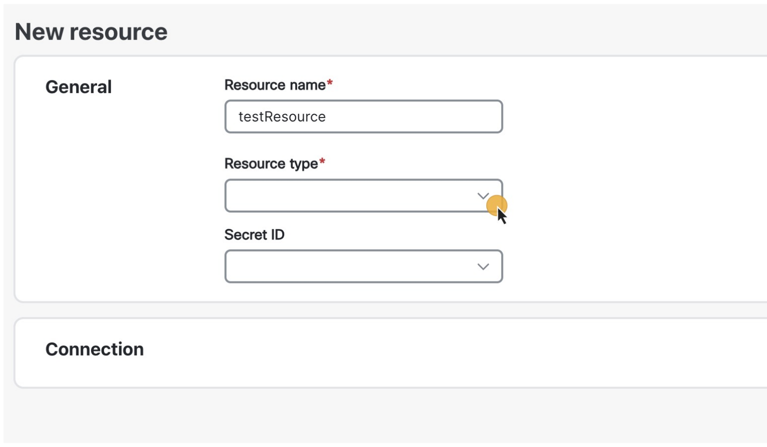
Add a resource

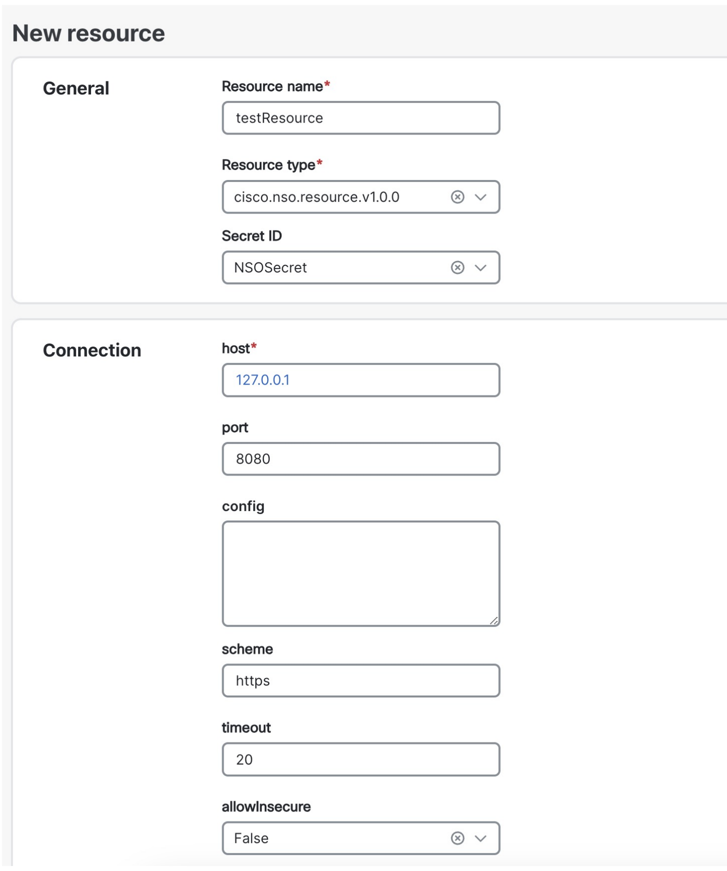
Before you begin

Ensure that you have already created a secret, as explained in Add a secret, on page 18

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Workflow Administration** > **Resources**	
Step 2	In the **Resources** view, click **Add resource**.	 ← Administration; Resources; Search; 2 matching results; Resource name / Resource type / Secret ID; nsoResource / cisco.nso.resource.v1.0.0 / nsoSecret; KafkaResource / system.event.kafka.v1.0.0 / KafkaSecret; Rows per p
Step 3	In the **New resource** view, provide the required input:	• **Resource name**: Enter a unique name for your resource. Once created, it cannot be modified.

	Command or Action	Purpose
		• **Resource type**: Select a **Resource type** from the drop-down menu. Resource types are defined inside adapters and are added to during adapter installation.  New resource — General — Resource name*: testResource — Resource type* — Secret ID — Connection
Step 4	After selecting the **Resource type**, the **Connection** section displays a set of additional fields (for example, **host** and **port**) . The adapter specifies these extra fields, and they will vary with the adapter and resource type. Fill in the fields with appropriate information.	

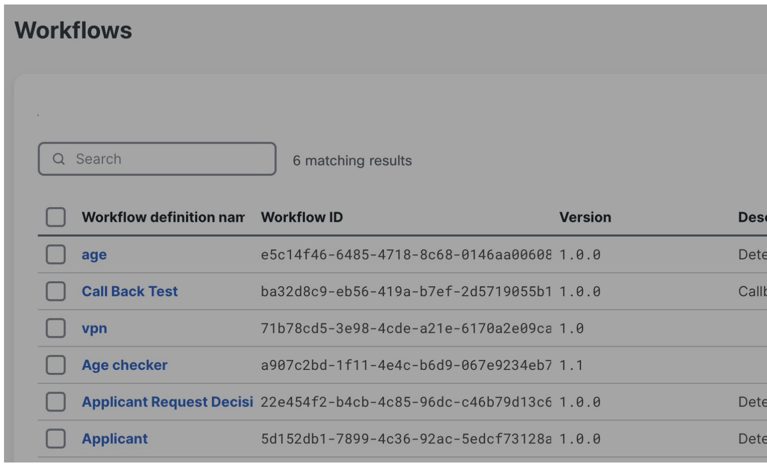
	Command or Action	Purpose
Step 5	Select a **Secret ID** from the drop-down menu to associate a previously created secret with this resource.	 New resource — General: Resource name* testResource; Resource type* cisco.nso.resource.v1.0.0; Secret ID NSOSecret. Connection: host* 127.0.0.1; port 8080; config; scheme https; timeout 20; allowInsecure False
Step 6	Click the **Create resource** button.	You are not required to fill in the **Secret ID** and **Connection** details when creating the resource. You can insert the proper values later, but you must fill them in before attempting to use the resource or it will not work properly. Once created, your resource is displayed on the list in the main **Resources** view.

Add workflow

This topic explains how to add a workflow definition to CWM.

Note These steps create a workflow entry in CWM with dummy code that needs further editing. After you create the workflow, go to the **Designer** tab to insert your workflow definition.

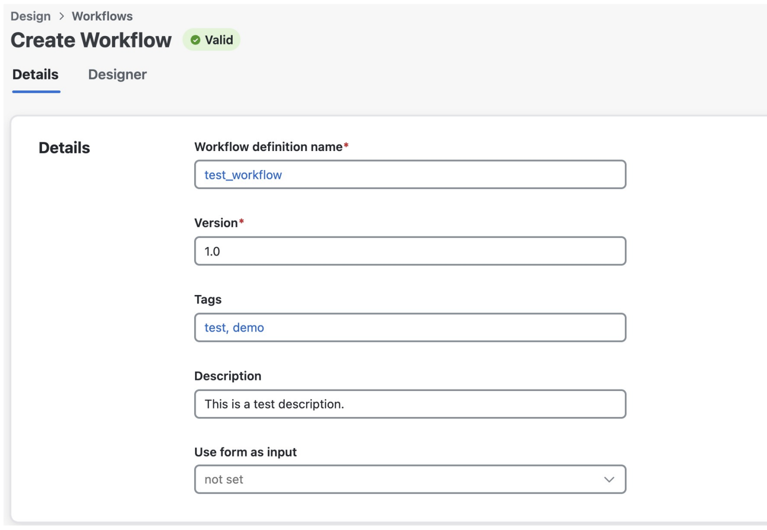
Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Design** > **Workflows**.	
Step 2	Click **Create workflow**.	

Workflows

Search — 6 matching results

Workflow definition nam	Workflow ID	Version	Desc
age	e5c14f46-6485-4718-8c68-0146aa00608	1.0.0	Dete
Call Back Test	ba32d8c9-eb56-419a-b7ef-2d5719055b1	1.0.0	Callb
vpn	71b78cd5-3e98-4cde-a21e-6170a2e09ca	1.0	
Age checker	a907c2bd-1f11-4e4c-b6d9-067e9234eb7	1.1	
Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Dete
Applicant	5d152db1-7899-4c36-92ac-5edcf73128a	1.0.0	Dete

	Command or Action	Purpose
Step 3	In the **Create workflow** page, provide the required input:	• **Workflow definition name**: Enter the name for your workflow definition, for example, `test_workflow`. • **Version**: Enter your workflow definition version, for example, `1.0`.
Step 4	Click **Create**.	

Design > Workflows

Create Workflow — Valid

Details

Workflow definition name*: test_workflow

Version*: 1.0

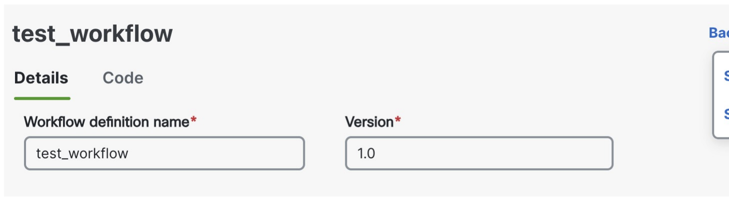
Tags: test, demo

Description: This is a test description.

Use form as input: not set

Workflow editing

Procedure

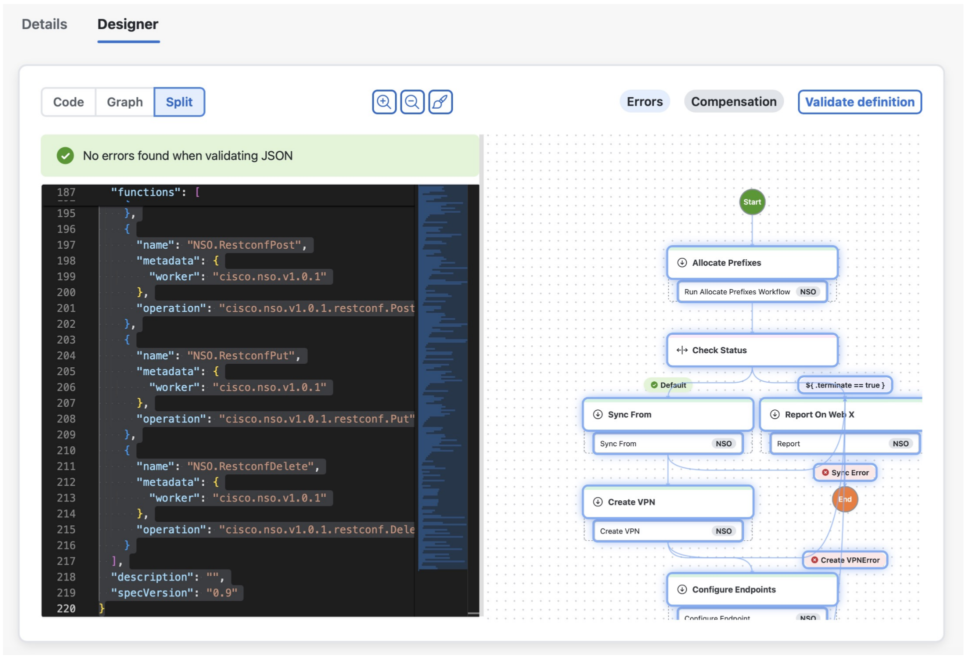
	Command or Action	Purpose
Step 1	To edit an existing workflow, first choose **Design** > **Workflows** to display the **Workflows** list.	
Step 2	In the list, find the workflow definition you want to edit. Then click on that definition's **Workflow name** to display the workflow's **Details** tab. You can then:	• Change the **Workflow definition name**. • Change the workflow's **Version** number. • Change, delete, or add new **Tags** to the workflow. To add more workflow tags, separate them with a comma. Workflow tags are separate from job tags, and are passed to every execution of this workflow definition (job run). You can edit workflow tags from the workflow definition details level, but once you add them to a run, they cannot be changed. • Add or edit the workflow's **Description**. • **Use form as input.** The form you set as input to the workflow must already exist.
Step 3	To edit the workflow definition itself, click the **Designer** tab By default, after creating a new workflow, the **Code** view contains a dummy piece of code that you need to replace with the actual developed workflow definition in JSON format, based on the Serverless Workflow Specification.	
Step 4	After making your changes, you can:	 test_workflow Details Code Workflow definition name* test_workflow Version* 1.0 • **Back**: Reject all the changes and go back to the previous page. • **Delete**: Remove the workflow definition from CWM. • **Save Changes**: Save the changes for the current workflow definition. • **Save as new workflow**: Createsa new workflow definition with inserted changes and keep the previously saved version of the workflow definition (do not overwrite the changes on the same workflow definition, but create a separate, new workflow definition).

	Command or Action	Purpose
		• **Run**: Execute the workflow definition as a single job. **Note** You can update the workflow definition and up-version it without affecting the currently running job.

Workflow designer

With **Designer**, you can view a graphic representation of your workflow code.

To use the Designer, choose **Design** > **Workflows**, click on any workflow to see its details, then click the **Designer** tab.



No errors found when validating JSON

```
  "functions": [
      },
      {
        "name": "NSO.RestconfPost",
        "metadata": {
          "worker": "cisco.nso.v1.0.1"
        },
        "operation": "cisco.nso.v1.0.1.restconf.Post
      },
      {
        "name": "NSO.RestconfPut",
        "metadata": {
          "worker": "cisco.nso.v1.0.1"
        },
        "operation": "cisco.nso.v1.0.1.restconf.Put"
      },
      {
        "name": "NSO.RestconfDelete",
        "metadata": {
          "worker": "cisco.nso.v1.0.1"
        },
        "operation": "cisco.nso.v1.0.1.restconf.Dele
      }
  ],
  "description": "",
  "specVersion": "0.9"
}
```

The Designer provides multiple viewing options for your workflow:

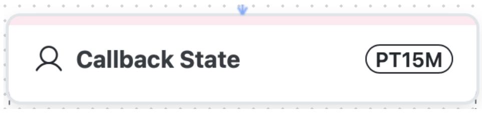
- View modes: Use the toggle buttons at the top to switch between:
  - Code view: displays the workflow's code.
  - Graph view: visualizes the workflow as a graph.
  - Split view: shows both the code and graph side by side.
- Graph controls:
  - Click the magnifying glass icons to zoom in or out.
  - Enable dark mode by clicking the brush icon.
- Top-Right controls:
  - Errors: Show or hide `OnErrors` definitions.

- Compensation: Show or hide compensation actions.
- Validate definition: Check for syntax errors, missing required fields, and other issues in the workflow.

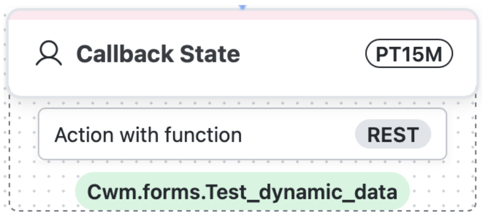
  Successful validation is required in order to save the workflow definition.

Inside a node

Nodes are the visualized steps of a workflow. The main element of a node is the **state**. States are represented inside state headers along with a label denoting the timeout defined for the particular state (if timeout is given):

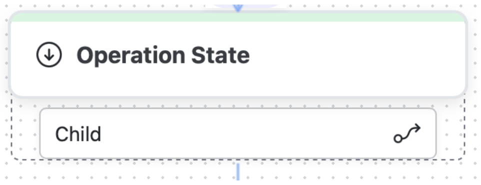


If an **action** is defined inside the state, it is displayed under the state header along with the name of the adapter that is assigned to perform it:

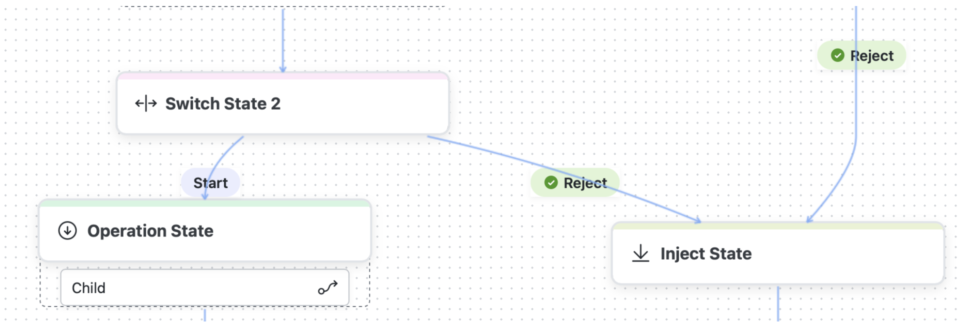


An **event** might be assigned to an action. In this case it will be represented by a green tag under the action element.

If a child workflow is defined inside a state, it is displayed as an action element under the state header with an icon:



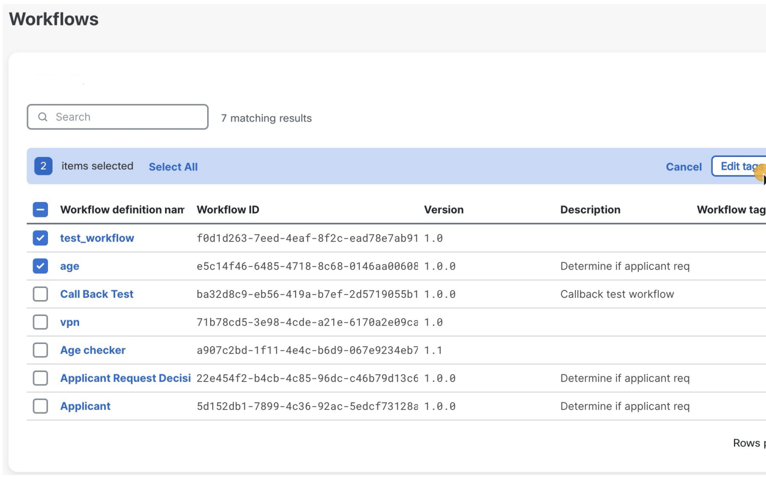
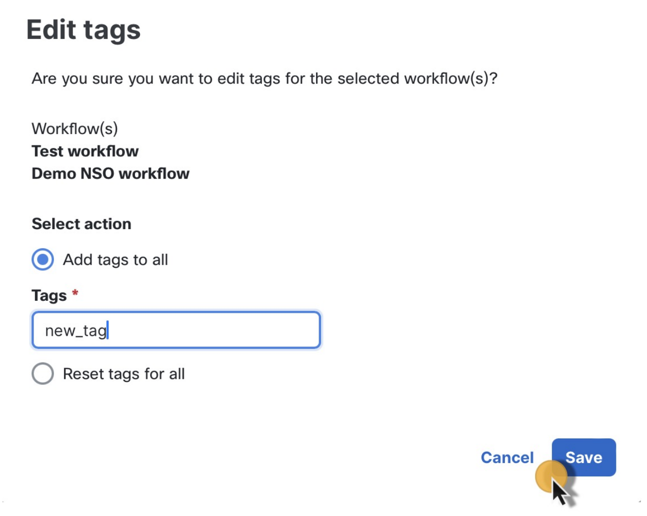
Data conditions are represented as decision points, displayed along the arrows branching down from one node to another within the tree. If a condition is marked as `default`, it will be shown inside a green label with a green checkmark icon.



Bulk tags editing

You can edit Tags for many workflow definitions at once. You can also add one or more Tags to all selected workflows or remove all Tags by resetting them.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Design** > **Workflows**.	
Step 2	Select chosen workflows by checking the checkbox on the left side of the **Workflows** table, then click on the **Edit tags** button.	
Step 3	In the **Edit tags** window, under the **Select action** section, click on the chosen radio button, depending on whether you plan to add tags or reset them:	• **Add tags to all**: Assign one or more tags (separated by a comma) to all selected workflows. • **Reset tags for all**: Remove all tags for selected workflows. 
Step 4	Click **Save**.	You can see the result under the **Workflow tags** column.

Workflows

Search 7 matching results

2 items selected Select All Cancel Edit tags

	Workflow definition nam	Workflow ID	Version	Description	Workflow tag
✓	test_workflow	f0d1d263-7eed-4eaf-8f2c-ead78e7ab91	1.0		
✓	age	e5c14f46-6485-4718-8c68-0146aa00608	1.0.0	Determine if applicant req	
☐	Call Back Test	ba32d8c9-eb56-419a-b7ef-2d5719055b1	1.0.0	Callback test workflow	
☐	vpn	71b78cd5-3e98-4cde-a21e-6170a2e09ca	1.0		
☐	Age checker	a907c2bd-1f11-4e4c-b6d9-067e9234eb7	1.1		
☐	Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Determine if applicant req	
☐	Applicant	5d152db1-7899-4c36-92ac-5edcf73128a	1.0.0	Determine if applicant req	

Rows

Edit tags

Are you sure you want to edit tags for the selected workflow(s)?

Workflow(s)
Test workflow
Demo NSO workflow

Select action

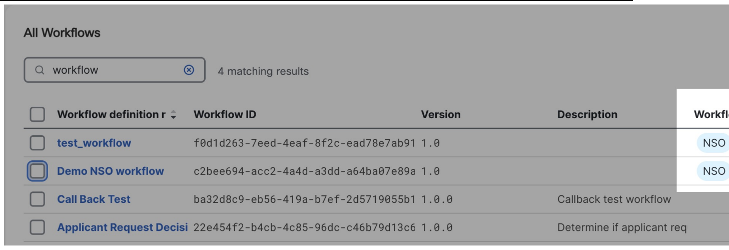
[✓ Add tags to all]

Tags *

new_tag

Reset tags for all

Cancel Save

	Command or Action	Purpose
		

All Workflows

workflow 4 matching results

Workflow definition r	Workflow ID	Version	Description	Workfl
test_workflow	f0d1d263-7eed-4eaf-8f2c-ead78e7ab91	1.0		NSO
Demo NSO workflow	c2bee694-acc2-4a4d-a3dd-a64ba07e89a	1.0		NSO
Call Back Test	ba32d8c9-eb56-419a-b7ef-2d5719055b1	1.0.0	Callback test workflow	
Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Determine if applicant req	

CHAPTER 3

Jobs

This section covers the following topics:

Jobs

After completing the instructions described in Initial setup, on page 13, you are ready to start executing workflows.

In CNC Workflow Automation, a particular execution of a workflow definition is called a job. By following the guidelines in this chapter, you can:

- Run a Job, on page 29 so that it starts immediately, or
- Schedule a job, on page 35 to run at one or more points in the future.

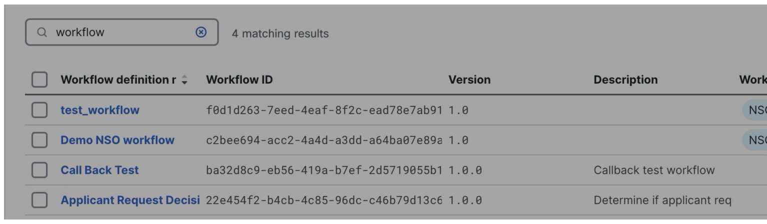
Run a Job

This topic explains how to run a job (workflow execution).

Before you begin

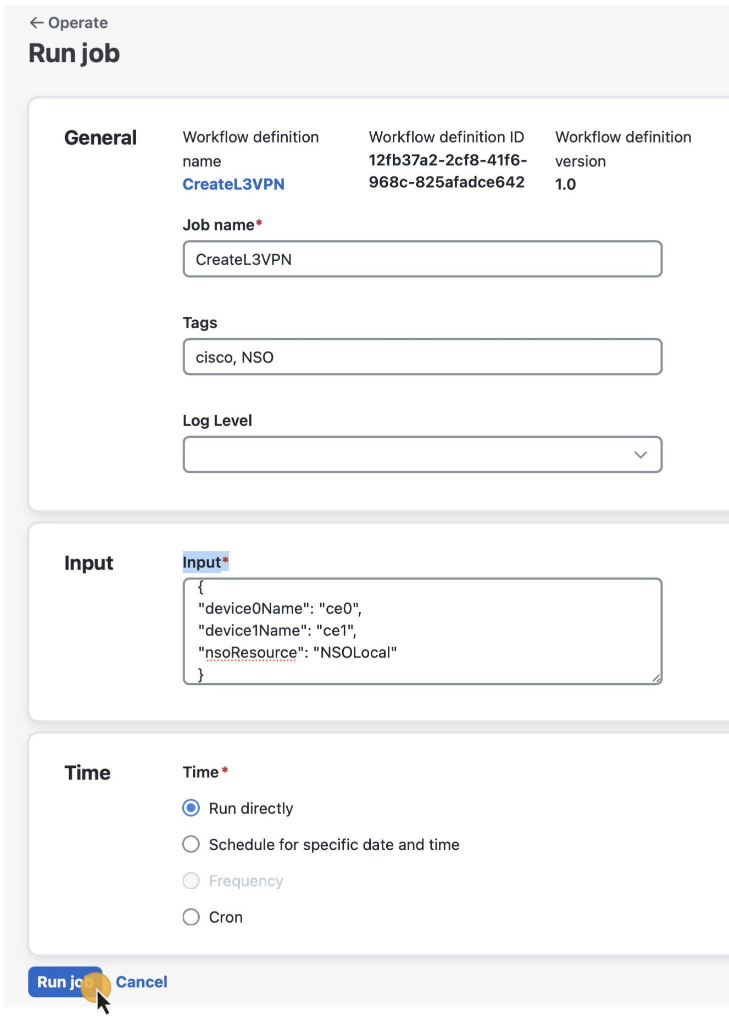
Ensure you have created a workflow definition and have uploaded one or more adapters with workers.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Design** > **Workflows** > Find the workflow you want to execute and click **Run** under the Actions column.	
Step 2	In the **Run job** window, insert the initial workflow data input in the **Input** field.	
Step 3	By default, your job's name is the same as your workflow definition, but you can change it in the **Job name** field.	
Step 4	Optionally, you can assign tags to your job. To insert more than one tag, separate them with a comma.	**Note** Tags are case-sensitive. Once added to the job, you cannot edit them.
Step 5	Leave **Run directly** selected to run your job immediately. If you want to create a schedule for a future run, follow the steps in Schedule a job, on page 35.	

workflow 4 matching results

	Workflow definition r	Workflow ID	Version	Description	Work
☐	test_workflow	f0d1d263-7eed-4eaf-8f2c-ead78e7ab91	1.0		NS
☐	Demo NSO workflow	c2bee694-acc2-4a4d-a3dd-a64ba07e89a	1.0		NS
☐	Call Back Test	ba32d8c9-eb56-419a-b7ef-2d5719055b1	1.0.0	Callback test workflow	
☐	Applicant Request Decisi	22e454f2-b4cb-4c85-96dc-c46b79d13c6	1.0.0	Determine if applicant req	

	Command or Action	Purpose
Step 6	Click **Run job** to start the workflow execution.	 ← Operate Run job General Workflow definition name: CreateL3VPN Workflow definition ID: 12fb37a2-2cf8-41f6-968c-825afadce642 Workflow definition version: 1.0 Job name*: CreateL3VPN Tags: cisco, NSO Log Level: Input Input*: { "device0Name": "ce0", "device1Name": "ce1", "nsoResource": "NSOLocal" } Time Time*: [✓ Run directly] / Schedule for specific date and time / Frequency / Cron Run job Cancel

Check job status

Procedure

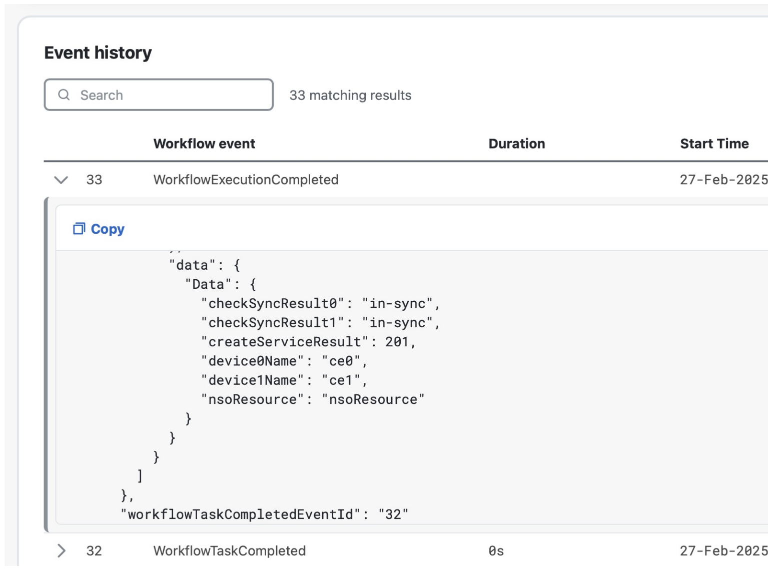
	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**.	
Step 2	In the **Jobs** table, find your job and check the status of the workflow execution in the **Status** column.	• If the workflow was executed correctly, a green icon and the status **Completed** is displayed. • If the workflow execution is still in progress or the workflow engine is retrying an action, a blue icon with three dots and the status **Running** is displayed.

	Command or Action	Purpose
		• If the workflow execution failed, a red icon and the status **Failed** is displayed.

Check job result

You can check results for any failed or completed job.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**	
Step 2	In the **Jobs** table, find the job you want and click the job name to view its details.	
Step 3	To see basic information about the job, review the **Job summary** tab at the top of the table.	
Step 4	To check the input and result of the execution, click the **Input and results** tab in the middle of the table.	Click **Copy** to copy the contents of the Input or Results panel to the clipboard.
Step 5	In **Event history** tab, you can expand each **Workflow event** entry by clicking on the arrow icon on the left-hand side.	Click **Copy** to copy the contents of any Workflow event panel to the clipboard. Or click **Download** to download the entire event history as a JSON file. 

Event history

Search | 33 matching results

		Workflow event	Duration	Start Time
∨	33	WorkflowExecutionCompleted		27-Feb-2025

Copy

```
            "data": {
              "Data": {
                "checkSyncResult0": "in-sync",
                "checkSyncResult1": "in-sync",
                "createServiceResult": 201,
                "device0Name": "ce0",
                "device1Name": "ce1",
                "nsoResource": "nsoResource"
              }
            }
          }
        ]
      },
      "workflowTaskCompletedEventId": "32"
```

		Workflow event	Duration	Start Time
>	32	WorkflowTaskCompleted	0s	27-Feb-2025

Find child workflow runs in Event History

If a given job run triggers child workflow executions, you can check them under the **Event history** log of the parent run. Child workflow names are created using the following schema:

parent_job_run_name.child_workflow_def_name.child_workflow def_version.

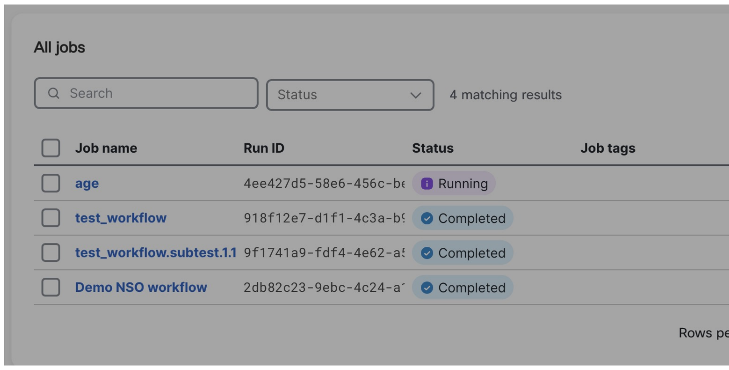
For example, if a parent job-run name happens to be `Parent`, the child workflow definition name is `Child`, and the child workflow definition version is `1.0`, the job run executed as a child workflow will be named `Parent.Child.1.0..`

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**	
Step 2	In the **Jobs** table, find the parent job you want and click the job name to view its details.	
Step 3	In **Event history**, under the **Workflow events** column, search for a `ChildWorkflowExecutionStarted` or `ChildWorkflowExecutionCompleted` entry. Click on its event name; you will be redirected to the job details page of this child workflow execution.	
Step 4	On the child job's **Details** page, in addition to the standard fields, you will find the **Parent Run ID** that redirects you to the details of the parent job run.	**Note** Child workflow runs do not inherit tags applied to their parent job runs.

Rerun job

Procedure

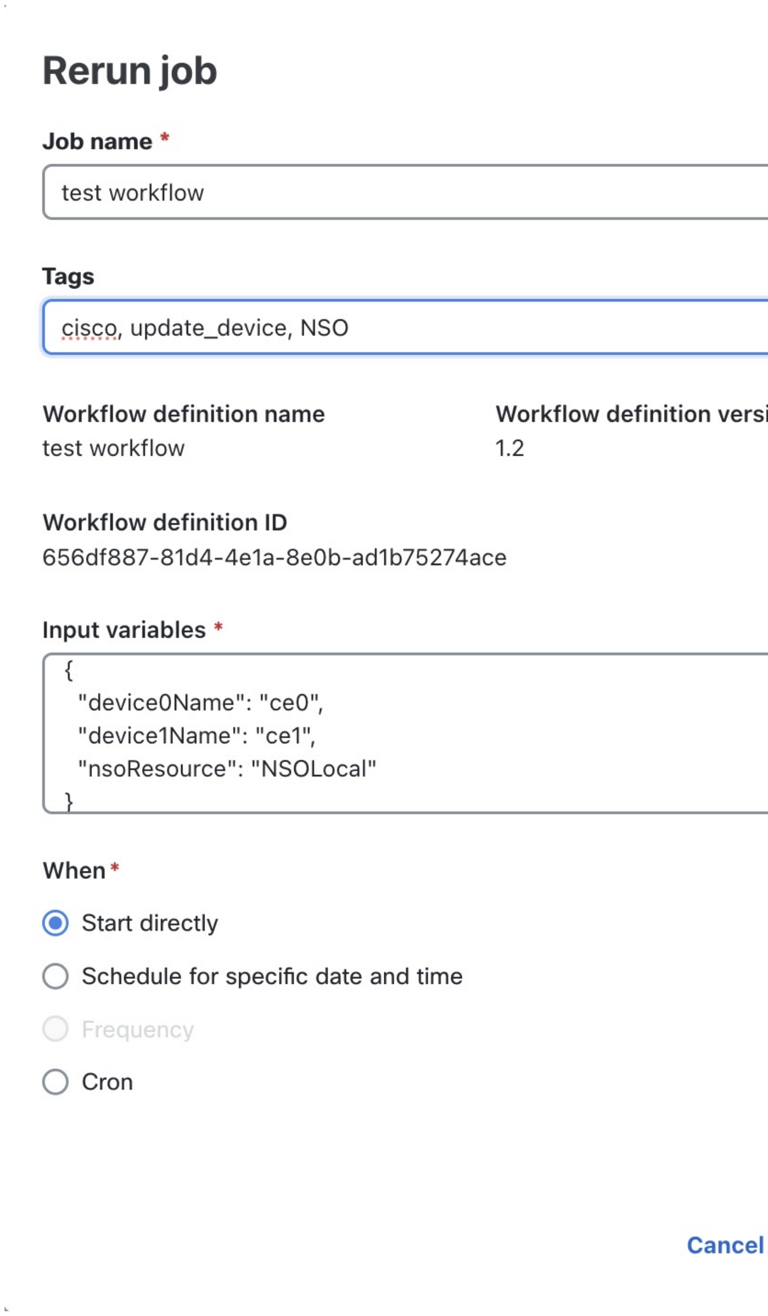
	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**.	
Step 2	In the **Jobs** table, find the job that you want to rerun.	
Step 3	Click the **Rerun** button in the Actions column in the same row as the job you want to rerun.	

All jobs

Search | Status | 4 matching results

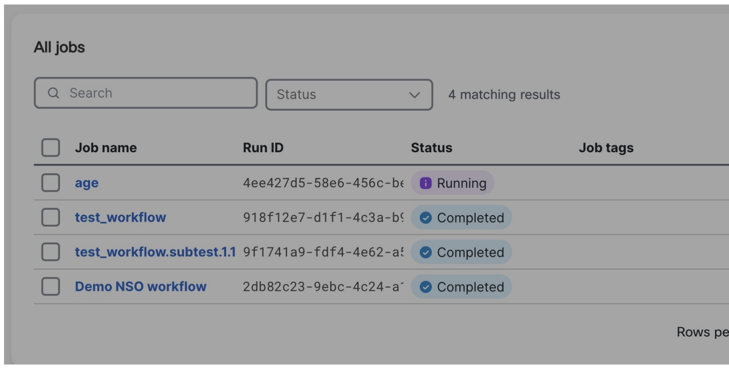
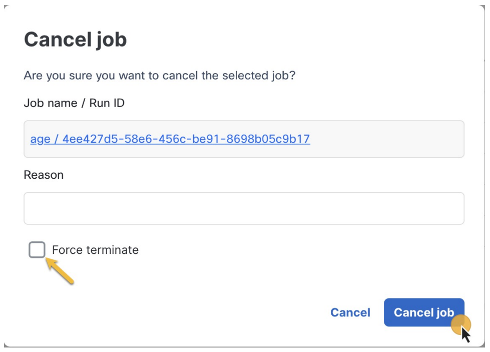
Job name	Run ID	Status	Job tags
age	4ee427d5-58e6-456c-be	Running	
test_workflow	918f12e7-d1f1-4c3a-b9	Completed	
test_workflow.subtest.1.1	9f1741a9-fdf4-4e62-a5	Completed	
Demo NSO workflow	2db82c23-9ebc-4c24-a1	Completed	

Rows pe

	Command or Action	Purpose
Step 4	In the **Rerun job** window, you can edit the **Job name**, tags and **Input** (data input) or use the same values as the previous run.	 Rerun job Job name *: test workflow Tags: cisco, update_device, NSO Workflow definition name: test workflow Workflow definition versi: 1.2 Workflow definition ID: 656df887-81d4-4e1a-8e0b-ad1b75274ace Input variables *: { "device0Name": "ce0", "device1Name": "ce1", "nsoResource": "NSOLocal" } When *: [✓ Start directly] Schedule for specific date and time Frequency Cron
Step 5	Click **Run job**.	You can also rerun the job from the job's **Details** tab.

Cancel job

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**.	
Step 2	Select the **Active** filter at the top of the table, find a job that you want to cancel and click the **Cancel** button in the same row under the Actions column.	 All jobs Search Status 4 matching results Rows pe Job name | Run ID | Status | Job tags age | 4ee427d5-58e6-456c-be | Running test_workflow | 918f12e7-d1f1-4c3a-b9 | Completed test_workflow.subtest.1.1 | 9f1741a9-fdf4-4e62-a5 | Completed Demo NSO workflow | 2db82c23-9ebc-4c24-a7 | Completed
Step 3	In the **Cancel job** window, you can enter an optional Reason for cancelling.	
Step 4	Check the **Force terminate** checkbox if you want to cancel the running job immediately. If you leave this checkbox unticked, the workflow worker will complete the ongoing task execution from the workflow definition and then cancel the job.	Cancel job Are you sure you want to cancel the selected job? Job name / Run ID age / 4ee427d5-58e6-456c-be91-8698b05c9b17 Reason Force terminate
Step 5	Click **Cancel Job**.	You can also cancel the job from the job's **Details** tab.

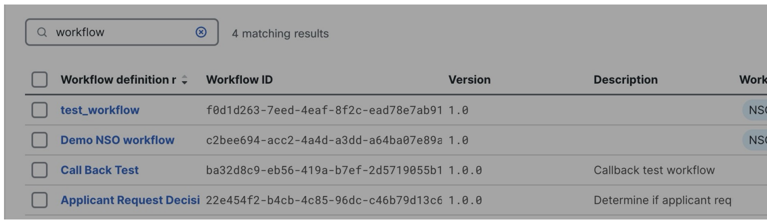
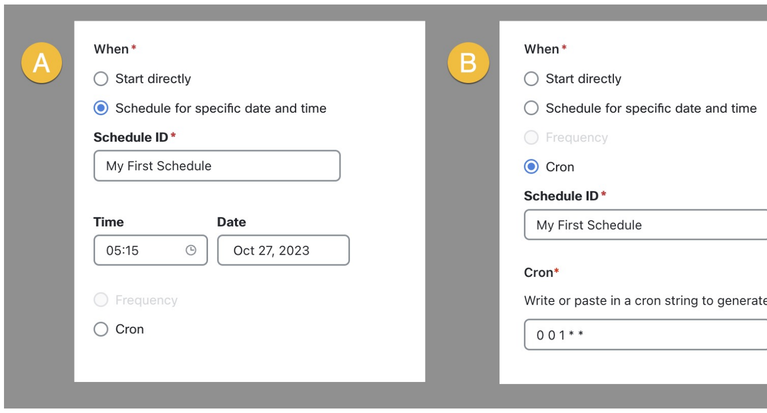
Schedule a job

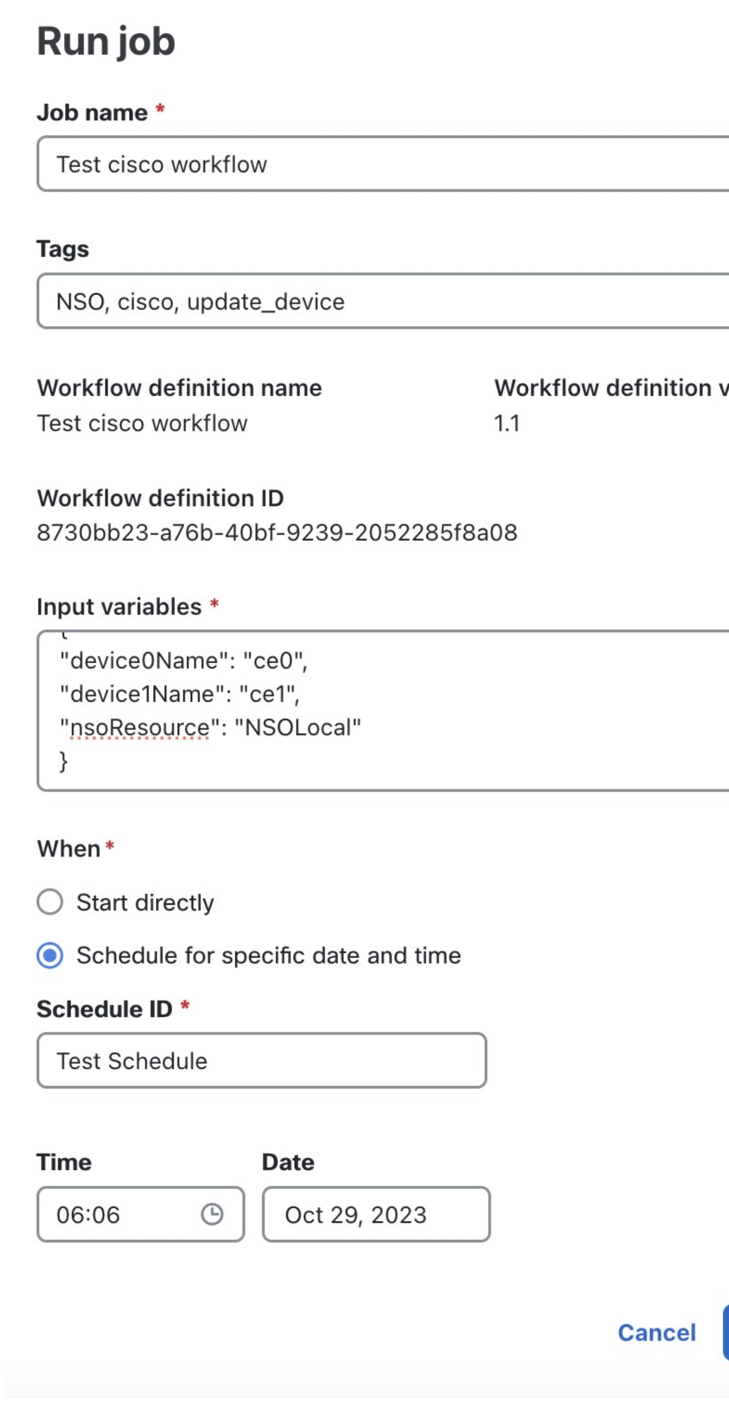
This topic explains how to schedule a single job run (workflow execution) or recurring runs that are invoked in the future.

Before you begin

Ensure that you have added a workflow definition and listed it in the **Workflows** view, and uploaded one or more adapters with workers.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Design** > **Workflows**.	
Step 2	Select the workflow you want to schedule and click **Run** under the Actions column.	 workflow 4 matching results Workflow definition r | Workflow ID | Version | Description | Work test_workflow | f0d1d263-7eed-4eaf-8f2c-ead78e7ab91 | 1.0 | | NS Demo NSO workflow | c2bee694-acc2-4a4d-a3dd-a64ba07e89a | 1.0 | | NS Call Back Test | ba32d8c9-eb56-419a-b7ef-2d5719055b1 | 1.0.0 | Callback test workflow | Applicant Request Decisi | 22e454f2-b4cb-4c85-96dc-c46b79d13c6 | 1.0.0 | Determine if applicant req |
Step 3	In the **Run job** window, insert the initial workflow data input in the **Input** field. Each scheduled run of a workflow receives identical input variables.	
Step 4	By default, your job name is the same as your workflow definition name, but you can change it in the **Job name** field.	
Step 5	Optionally, you can assign tags to your scheduled job(s). To insert more than one tag, separate them with a comma. All jobs in the schedule will have the same tags set.	
Step 6	Choose one scheduling option depending on whether you planned a single run in the future or a recurring series of runs:	• **A) Schedule for specific date and time**: Pick a specific time and date for a single run. • **B) Cron**: set recurring runs based on cron expressions  A: When* Start directly [✓ Schedule for specific date and time] Schedule ID*: My First Schedule Time: 05:15 Date: Oct 27, 2023 Frequency Cron B: When* Start directly Schedule for specific date and time Frequency [✓ Cron] Schedule ID*: My First Schedule Cron* Write or paste in a cron string to generate 0 0 1 * *
Step 7	Type a unique name for your schedule in the Schedule ID field.	

	Command or Action	Purpose
Step 8	Click **Schedule Job** to schedule the workflow execution(s).	 Run job Job name * Test cisco workflow Tags NSO, cisco, update_device Workflow definition name: Test cisco workflow Workflow definition v: 1.1 Workflow definition ID: 8730bb23-a76b-40bf-9239-2052285f8a08 Input variables * "device0Name": "ce0", "device1Name": "ce1", "nsoResource": "NSOLocal" } When * Start directly [✓ Schedule for specific date and time] Schedule ID * Test Schedule Time: 06:06 Date: Oct 29, 2023 Cancel **Note** **Overlap policy**: By default, the schedule overlap policy prevents overlapping runs in one schedule. If the previous job from a given schedule is still being executed while it's

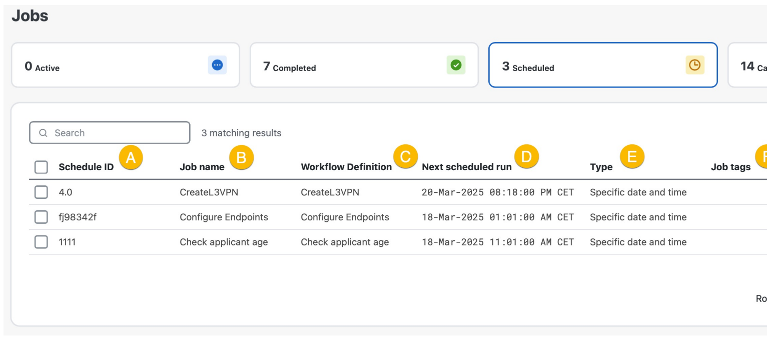
	Command or Action	Purpose
		time to start the next scheduled job, the next run will be skipped. If you want to allow overlapping runs, you can change the overlap policy via the API.

Check scheduled jobs

All schedule jobs are displayed in the **Operate** > **Jobs** window, under the **Scheduled jobs** tab. When a scheduled job is executed, it will be shown in the **Active jobs** tab. When the job is completed, it will display in the **Completed jobs** tab.

A schedule job does not have an expiration date. This means that you can update the scheduled job anytime and retrieve some of its details. A scheduled job is displayed on the **Scheduled job** table even if it only one job is scheduled for a specific date and time, and that job has already been executed. If you don't want the scheduled job to be displayed in the list of scheduled jobs, you must delete it.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**.	
Step 2	Click the **Scheduled jobs** tab at the top of the window.	
Step 3	In the **Scheduled jobs** table, you can find all the schedule jobs, with details:	 • **A) Schedule ID**: unique identifier for every schedule • **B) Job Name**: name of the scheduled workflow execution; shared between all runs in a schedule • **C) Workflow Definition**: name of the workflow definition used to run a scheduled job(s) • **D) Next scheduled run**: the date and time of the next upcoming run in the schedule, for example `05-May-2023 03:59:25 PM CEST` • **E) Type**: indicates whether the scheduled run is planned once in the future (*Specific date and time*), whether there are recurring runs (*Cron*) or recurring runs based on frequency (*Frequency*).

Jobs

0 Active | 7 Completed | 3 Scheduled | 14 Ca

Search | 3 matching results

Schedule ID	Job name	Workflow Definition	Next scheduled run	Type	Job tags
4.0	CreateL3VPN	CreateL3VPN	20-Mar-2025 08:18:00 PM CET	Specific date and time	
fj98342f	Configure Endpoints	Configure Endpoints	18-Mar-2025 01:01:00 AM CET	Specific date and time	
1111	Check applicant age	Check applicant age	18-Mar-2025 11:01:00 AM CET	Specific date and time	

	Command or Action	Purpose
		• **F) Job tags**: keyword(s) assigned to the scheduled job run • **G) Workflow tags**: keyword(s) assigned to the workflow definition used for that planned run plus its version • **H) Actions**: displays possible actions for the schedule.

Delete scheduled jobs

Deleting a scheduled job will delete all future planned jobs in that schedule, but will not affect a job that is already in execution. You can cancel a scheduled job that is already being executed using the Cancel job, on page 35 action.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Operate** > **Jobs**.	
Step 2	Click the **Scheduled jobs** tab at the top of the window.	
Step 3	In the **Scheduled Jobs** table, find a scheduled job that you want to delete and click the **Delete** button under the Actions column in the same row as the job you want to delete.	

CHAPTER 4

Adapters

This section covers the following topics:

Use the generic-email adapter

The **generic email adapter** (`generic-email`) enables you to add email reporting into your workflows. The adapter provides the ability to send emails using an SMTP server. For the latest adapter version, you can use the `Send` activity to send an email with a message defined inside the workflow definition.

You can get the generic email adapter by downloading the CWM package `cwm.v2.0.generic.email.v1.0.3.tar.gz` from the Cisco Software Download page.

To install the adapter, follow the instructions in Install an adapter, on page 14.

Add an SMTP secret

Let's start by defining an SMTP secret.

Before you begin

Before going into the details of defining your email message inside a workflow, you need to add an SMTP secret and resource to CWM. You will later need to reference them inside your workflow.

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Secrets**.	
Step 2	In the **Secrets** view, click **Add secret**.	

	Command or Action	Purpose
Step 3	In the **New secret** view, provide the required input:	• **Secret ID**: Name your secret. You'll need to reference this secret ID later in the resource and inside the workflow. E.g. `emailSecret`. • **Secret type**: Select `basicAuth`.
Step 4	After selecting the secret type, a set of additional fields is displayed under the **Secret type details** section. Fill in the fields with the following:	• **Password**: Enter the password for the sender's email address. • **Username**: Enter the sender's email address, in the following format: `sender@address.com`.
Step 5	Click the **Create Secret** button.	You can edit the secret type and its details later. Once you've added a secret and its details, create a resource in and associate it with the secret, as explained in Add an SMTP resource, on page 42.

Add an SMTP resource

Now we can create the resource.

Before you begin

Ensure that you have already created an email secret, as explained in Add an SMTP secret, on page 41

Procedure

	Command or Action	Purpose
Step 1	In CWM, choose **Administration** > **Workflow Administration** > **Resources**.	
Step 2	In the **Resources** view, click **Add resource**.	
Step 3	In the **New resource** view, provide the required input:	• **Resource name**: Name your resource. You'll need to provide the resource ID later inside the workflow as a reference. For example: `emailResource`. • **Resource type**: Select `generic.email.resource.v1.0.3`. • **Connection**: • **Host**: Enter the URL or IP address of the SMTP email server to be used. • **Port**: Enter the outgoing SMTP port on the server. The default outgoing SMTP port for encrypted email transmission is 587. Port 465 is still supported by some providers, 25 is supported for

	Command or Action	Purpose
		SMTP relay, and 2525 is a popular alternative when 587 is blocked. • **Scheme**: Leave blank (it's not required for this workflow). • **Timeout**: Leave blank (it's not required for this workflow). • **Allow insecure**: Select `true`.
Step 4	Select a **Secret ID** from the drop-down menu to associate a previously created secret with this resource.	
Step 5	Click the **Create resource** button.	Once created, your resource is displayed on the list in the main **Resources** view.

Define the Send activity in a workflow

Once you've created the adapter, secret and resource, you can use it in a workflow.

Set activity reference

In CWM, the adapter `Send` activity is referred to as `generic.email.smtp.Send`. When defining a workflow, you need to specify it as the value of the `operations` parameter under `functions`:

```
"functions": [
  {
    "name": "smtp.send",
    "operation": "generic.email.v1.0.2.smtp.Send"
  }
]
```

Inside the `name` parameter, provide an activity name that you will later refer to in the `refName` parameter while defining the action.

Define email message in actions

Now you can define an action in which an email will be sent as part of a workflow state.

The available input parameters for the action are:

Field	Type	Label	Description
from	string		Sender email address
to	string	repeated	List of recipient email addresses
cc	string	repeated	List of recipient cc email addresses
bcc	string	repeated	List of recipient bcc email addresses

Field	Type	Label	Description
subject	string		Email title
text	string		Email body as text
html	string		Email body as html

Use the available fields as the `input` key/value pairs within the `arguments` that define the `SendEmail` example action, as shown below:

```
"states":[
  {
    "name":"EmailState",
    "type":"operation",
    "end": true,
    "actions": [
      {
        "name": "SendEmail",
        "functionRef": {
          "refName": "smtp.send",
          "arguments": {
            "input": {
              "to": ["recipient1@address.com", "recipient2@address.com"],
              "from": "sender@address.com",
              "text": "Hello, this is some placeholder email text.",
              "subject": "A test email from CWM"
            },
            "config": {
              "resourceId": "emailResource"
            }
          }
        }
      }
    ]
  }
]
```

CHAPTER 5

Guided tasks

This section covers the following topics:

Forms and guided tasks

Forms are customizable templates that are used to create what we can call guided tasks. These forms serve to introduce an element of human intervention into an automated workflow. They are configurable using the drag-and-drop **Forms Designer** in the CWM user interface. Forms can contain a number of different optional input components (such as text, dropdowns, radio buttons, and checkboxes) and layout components (such as tabs, columns, cards, HTML elements, and more).

With forms, you define what kind of input you want to include in your manual task and how it will be laid out in the task prompt itself. Forms can be set to contain static form elements (set values) or dynamic data elements (where values can be set using jq expressions).

Forms list

Choose **Design** > **Forms** to access Forms view. Here, you can view, edit, export, and delete all the existing forms, or access the **Forms Designer** by clicking **Create form**.

To export forms to a CSV file, check one or more of the check boxes shown next to the Form names in the list and then click **Export**.

Forms Designer

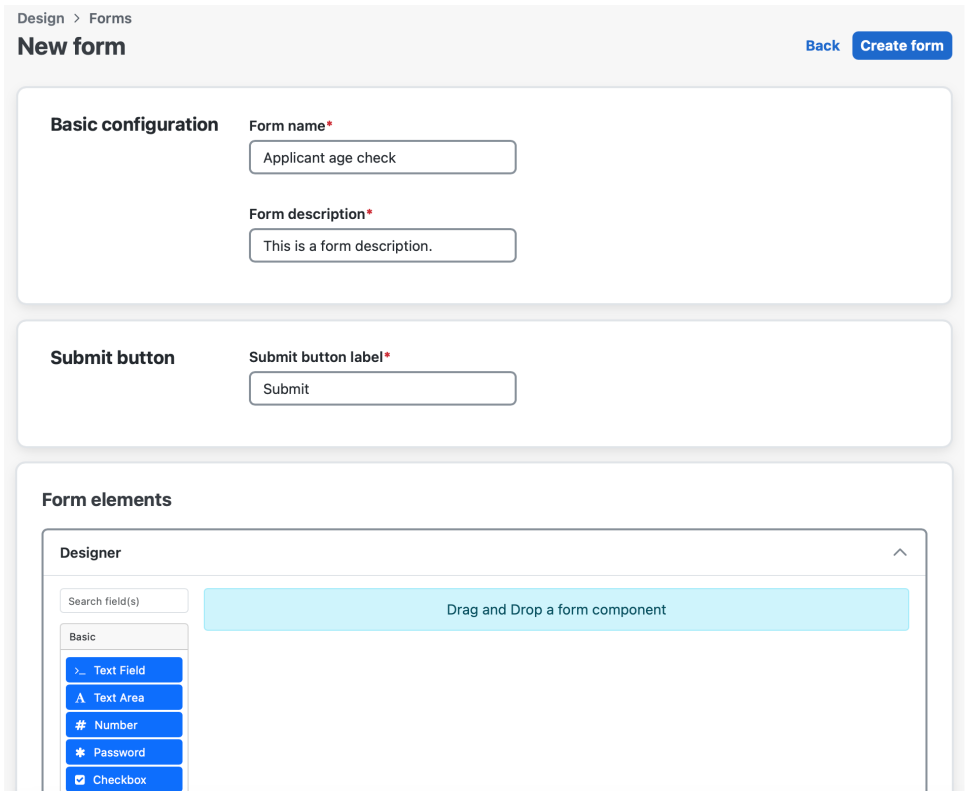
The **Forms Designer** is a user-friendly drag-and-drop editor accessible via the **Forms** in the CWM UI. To get started, simply click the **Create form** button. The designer consists of three main panels:

- **Basic configuration**: Here, you provide a name and a description for your form. Those are required fields.

Note Important: The name you provide for your form determines the **Form ID** that you will need to provide inside your workflow. To retrieve the **Form ID**, click on the name of the newly added form in the forms list.

- **Submit button**: You can customize the **Submit** button label here. The button is automatically added to every form you create.
- **Form elements**: This is where you design and preview your form.

Within the **Form elements** panel, the **Designer** field is where you craft the input, structure, and visual layout of your form.



Form name*: Applicant age check

Form description*: This is a form description.

Submit button label*: Submit

The Forms Designer's components are grouped into four categories:

- Basic

- Advanced
- Layout
- Data

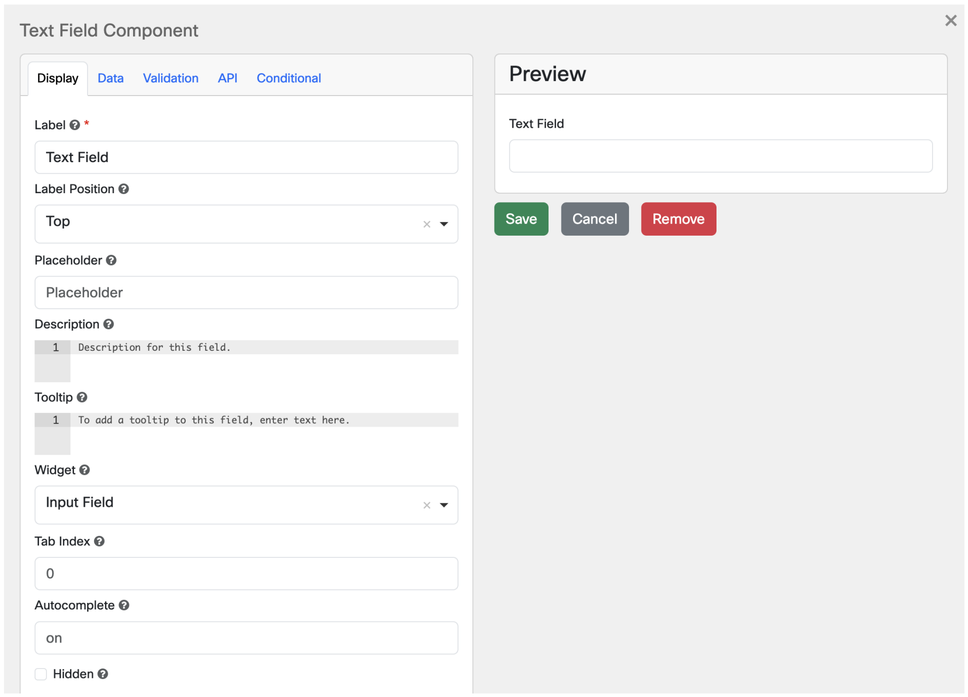
Input components

The **Basic** and **Advanced** categories contain customizable input components. Using those components you can define how you would like to structure your task and what type of input(s) would be required to proceed. It can be:

- text fields or areas,
- numbers,
- passwords,
- checkboxes or select boxes,
- drop-down lists (called *selects* in the UI),
- radio buttons,
- email fields,
- date and time fields, and
- signatures.

Inside a component

To add and customize a component, drag and drop it onto the blue rectangle inside the **Designer** field. A panel will appear containing a few tabs and a **Preview** field.



Label *
Text Field
Label Position
Top
Placeholder
Placeholder
Description
Description for this field.
Tooltip
To add a tooltip to this field, enter text here.
Widget
Input Field
Tab Index
0
Autocomplete
on
Hidden

Text Field

- The **Display** tab is where you customize your input field. It features options like placeholder, description, tooltip and others. Those options may differ depending on the type of component that you selected, but there is one that is required:
  - **Label**: provide a name for the field that will appear next to it in the task prompt. The name you provide in the the **Label** field will be used for the **Property Name** field in the API tab of the component.
- The **Data** tab allows you to add default values in the input and modify the way those values are handled. This tab is available for selected components.
- The **Validation** tab features a number of options that you can use to enforce input rules for your component:
  - **Required**: ensures that input field will not be left empty before submission. The form will not submit unless a value is provided.
  - **Minimum Length**: Sets the minimum number of characters required in the input.
  - **Maximum Length**: Limits the number of characters allowed in the input.
  - **Regular Expression Pattern** – Uses a regex pattern to enforce specific formatting or constraints.
  - **Error Label** – Specifies the label for this field when an error occurs.
- The **API** tab features the **Property Name** field which you need to provide as the component name inside your workflow.
- The **Conditional** tab lets you configure the behavior of the component in relation to the input provided in any other component inside yur task prompt.

To edit an added component, hover over the component and click the **cogwheel** icon.

Layout components

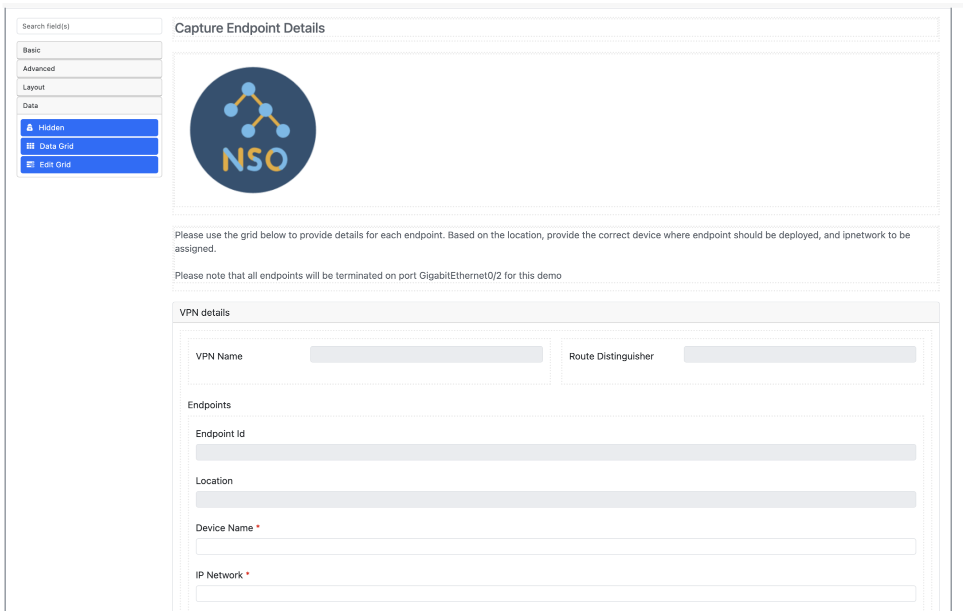
Use a **Layout** component to change the general layout and position of fields on a form. It allows you to add logos, headers, or static contextual language to your form.

Data components

Data components are special components that are used to not only visually show the data being collected in a data structured way, but will also change the data structure of the submissions being produced by the rendered form.

Sample form

Let's create an advanced form with headlines, graphics, and data inputs that would allow the workflow operator to validate or update specific VPN endpoint attributes in Cisco NSO.



Capture Endpoint Details

Please use the grid below to provide details for each endpoint. Based on the location, provide the correct device where endpoint should be deployed, and ipnetwork to be assigned.

Please note that all endpoints will be terminated on port GigabitEthernet0/2 for this demo

VPN details

VPN Name

Route Distinguisher

Endpoints

Endpoint Id

Location

Device Name *

IP Network *

Go through the elements of the form one by one to see how you can set it up:

- **Header**: use the HTML element (**Layout** tab). Type your title and provide the relevant HTML tag (`h3` in the example).
- **Image**: use the HTML element (**Layout** tab). Type `img` in the **HTML tag** field and use the **Attributes** for defining image source, description, and other properties. Alternatively, provide your image address inside the `img` HTML tag in the **Content** field (for example: `<img src="https://your-image-url.com/image.jpg" alt="Description" width="300" >`).
- **Description**: use the HTML element (**Layout** tab). Provide your description. You may want to use HTML formatting like `</br></br>` for better text layout.
- **VPN details panel**: use the Panel element (**Layout** tab). It will serve as a container for other elements.
  - **Read-only VPN name and RD fields**: use the Columns element (**Layout** tab). Note that these fields will be populated with workflow data and are therefore read-only.

    - **VPN Name field**: use the Text field component. Label position is **Left-aligned** in the example. Check the **Disabled** box to make the field read-only. In the **Data** tab, click on **Custom Default Value** and provide dynamic value for the field so that it can be populated by the workflow input (`value = form.dynamicData.input.vpnName`).
    - **Route Distinguisher field**: use the Number component. Label position is **Left-aligned** in the example. Check the **Disabled** box to make the field read-only. In the Data tab, provide dynamic value for the field so that it can be populated by the workflow input (`value = form.dynamicData.input.rd`).

- **Endpoint information grid**: Use the edit grid component (**Data** tab). Edit grid is for capturing endpoint details, where you can view and edit information about endpoints. Note that the read-only fields will be populated with workflow data, so be sure to provide dynamic value in the **Custom Default Value** field of the **Data** tab inside the component (`value = form.dynamicData.input.endpoint`).
    - **Endpoint Id, Location, Bandwidth fields**: use the Text field component. Label position is **Top** in the example. Check the **Disabled** box to make the field read-only.
    - **Device name and IP Network**: use the Text field component. Label position is **Top** in the example. Keep the **Disabled** box unchecked so that the operator can pass the data once the task is triggered. In the **Validation** tab, check the **Required** box.

To learn how you can use dynamic data in forms, see:

- Dynamic data: Create a form with custom component values, on page 50
- Dynamic data example: Create a dry-run approval form, on page 51

Dynamic data: Create a form with custom component values

You can use jq expressions to display data accessed by the workflow within a guided task, then filter or modify the data before passing it to the task prompt.

One example might be populating the Select component options for a workflow with custom values by means of `form.dynamicData`. Here is how you configure the use of dynamic data for this case when creating a form.

Procedure

	Command or Action	Purpose
Step 1	In **Forms Designer**, click **Basic** to expand it. From there, drag and drop the **Select** component.	
Step 2	Go to the **Data** tab and in **Data Source Type**, select **Custom**.	
Step 3	In the **Custom values** field, provide the following: `values = form.dynamicData.options;`	
Step 4	Click **Save**.	

	Command or Action	Purpose
Step 5	Now let's configure the workflow side of the dynamic data: In your workflow definition, find the `callback` state.	
Step 6	Inside the `callback` state, add the `metadata` key. Then, inside it, add `formData` and provide a value in the following format: `"${ { options : .options } }"`.	
Step 7	A sample workflow with the dynamic data part configured may look like this:	`{ "name": "decision", "type": "callback", "action": { "name": "decision_point", .... "actionDataFilter": { "toStateData": "${ .result }" } }, "eventRef": "cwm.forms.Options", "metadata": { "formData": "${ { options : .options } }", "taskName": "Select option" }, }`
Step 8	To provide the option labels and values for the Select component, use the **Input** when running the workflow:	`{ "options": [ { "label": "First option", "value": "Value1" }, { "label": "Second option", "value": "Value2" } ] }`

Step 7 purpose:

```
        {
            "name": "decision",
            "type": "callback",
            "action": {
                "name": "decision_point",
                 ....
                "actionDataFilter": {
                    "toStateData": "${ .result }"
                }
            },
            "eventRef": "cwm.forms.Options",
            "metadata": {
               "formData": "${ { options : .options
 } }",
                "taskName": "Select option"
            },
        }
```

Step 8 purpose:

```
      {
      "options": [
          {
          "label": "First option",
          "value": "Value1"
          },
          {
          "label": "Second option",
          "value": "Value2"
          }
      ]
      }
```

Dynamic data example: Create a dry-run approval form

You can use jq expressions to display data accessed by the workflow within a guided task, then filter or modify the data before passing it to the task prompt.

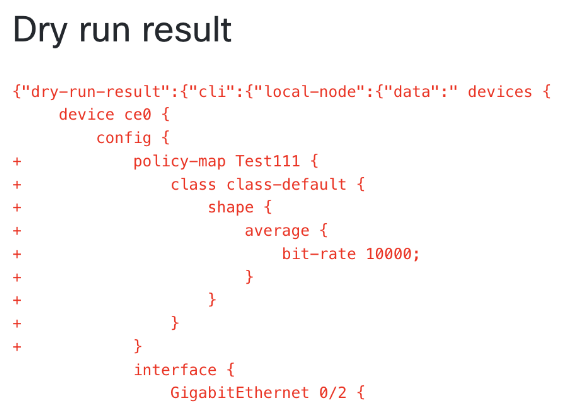
Here's an example: Let's say we want to retrieve the dry run of a network configuration change in Cisco NSO and present its payload for approval or rejection by a network operator. The following steps show how to use dynamic data when creating a form to use in this case.

Procedure

	Command or Action	Purpose
Step 1	In **Forms Designer**, click **Layout** to expand it. From there, drag and drop the **HTML element** component.	

	Command or Action	Purpose
Step 2	In the Content field, provide the *jq* expression inside the `code` brackets, for example: `<code>${ .dryrunPre }</code>`.	**Note** The expression begins with a `$` and is enclosed in curly brackets. The text inside the brackets can be custom, but it needs to be a *jq* expression. You will later reference it in the workflow definition and the input from the workflow will be presented inside the HTML element once the task is prompted. For more about how to build jq expressions, refer to the jq manual.
Step 3	Click **Save**.	
Step 4	Now let's configure the workflow side of the dynamic data: In your workflow definition, find the `callback` state.	
Step 5	Inside the `callback` state, add the `metadata` key. Then, inside it, add `formData` and provide a value in the following format: `"${ { dryrunPre : .result.data } }"`.	**Note** Note that `dryrunPre` refers to the expression defined within the HTML element inside the form, whereas the other part refers to the value of the `toStateData` key inside `actionDataFilter`. In this case, `actionDataFilter` handles and filters the payload provided by NSO during the execution of a dry run for the **Create L3VPN** operation. Be sure to customize the provided example according to your specific use case.
Step 6	A sample workflow with the dynamic data part configured may look like this:	`{ "name": "decision", "type": "callback", "action": { "name": "show-dry-run", .... "actionDataFilter": { "toStateData": "${ .result }" } }, "eventRef": "cwm.forms.NSO_dry_run", "metadata": { "formData": "${ { dryrunPre : .result.data } }", "taskName": "Create L3VPN with dry run" }, }`

```
{
  "name": "decision",
  "type": "callback",
  "action": {
      "name": "show-dry-run",
       ....
      "actionDataFilter": {
          "toStateData": "${ .result }"
      }
  },
  "eventRef": "cwm.forms.NSO_dry_run",
  "metadata": {
      "formData": "${ { dryrunPre : .result.data
} }",
      "taskName": "Create L3VPN with dry run"
  },
}
```

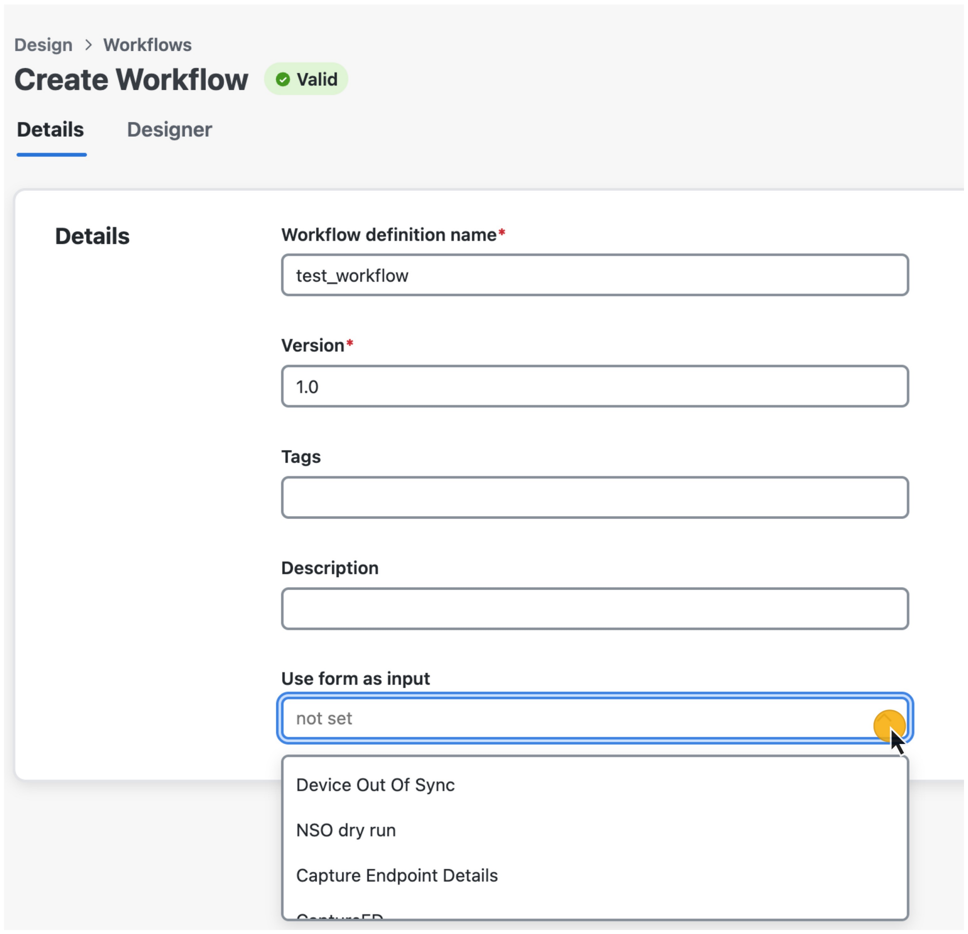
	Command or Action	Purpose
Step 7	Once you start the workflow and the task is invoked, you will see the payload presented inside the prompt.	

Dry run result

```
{"dry-run-result":{"cli":{"local-node":{"data":" devices {
     device ce0 {
         config {
+            policy-map Test111 {
+                class class-default {
+                    shape {
+                        average {
+                            bit-rate 10000;
+                        }
+                    }
+                }
+            }
             interface {
                 GigabitEthernet 0/2 {
```

Use a form as workflow input

Use form as input is an option that allows you to provide initial input for a workflow by selecting a prebuilt form instead of manually entering JSON data. It helps to standardize input collection by providing a structured interface for initial workflow data.


Design > Workflows

Create Workflow ✓ Valid

Details | Designer

Details

Workflow definition name*: test_workflow

Version*: 1.0

Tags:

Description:

Use form as input: not set

To use this option, select the **Use form as input** option when creating a workflow and choose an existing form. When running the workflow, you will be able to submit data so that it is automatically mapped to the workflow's input, ensuring consistency and reducing manual errors.

Guided task: Check an applicant's age

About this example

In this example, we create a simple employment application form that check's an applicant's age.

Click here to download and extract the Check Age Workflow file, including the JSON form file.

Step 1: Create a form

There are two ways you can create a form in CWM. To create the form without **Forms Designer**, use the `POST/form` API endpoint to send a JSON object that defines the form schema. The other option is to use the CWM user interface and create the form in a step-by-step process. Follow the steps below to create the form using the user interface.

1. In CWM, select **Design** > **Forms**, then click **Create form**.

2. In **Basic configuration**, enter:
    - **Form name**: type `Check applicant age`.
    - **Submit button**: A label for the **Submit** button.
    - **Description**: A short description, such as "Applicant form".

Step 2: Add Number component

The number component will add a field in the guided task that the operator will use to type the applicant age.

1. In **Form** elements, click the **Designer** to expand it.
2. In **Basic** components, locate the **Number** component and drag and drop it onto the blue rectangle in the center space. This will open a new dialog.
3. n the **Label** field, change the default name to `Provide applicant age:`.

Note In the API tab of the component, you'll find the **Property name** which will be needed later when defining the task on the workflow side.

4. Click **Save** to create the component.

Step 3: Define a task in the workflow

After you create a form, you can proceed to define how it will be employed by the workflow. The workflow that will serve as an example consists in a simple action: checking applicant age using the guided task feature and then rejecting or accepting an application based on the provided input. In the workflow definition, let's start by setting up the events parameter with the data of the form we created.

```
"events": [
    {
      "kind": "consumed",
      "name": "cwm.forms.Check_applicant_age",
      "type": "cwm.forms.Check_applicant_age",
      "source": "system.internal.cwm"
    }
  ],
```

The key values here are:

- `kind`: The event is `consumed`, since the input provided in the task is treated as external and will be consumed by the workflow.
- `name`: It's the **Form ID** which you can find in the form details when selecting it from the **Forms** list.
- `type`: It takes the same value as the `name` key.
- `source`: It needs to be `system.internal.cwm` as the source

Next, we want to enter a `callback` state into the workflow and refer to the *event* (the form ID) that it will use to invoke the task.

```
"states": [
    {
        "name": "start",
        "type": "callback",
        "action": {
          "name": "no operation",
          "functionRef": {
            "refName": "noop",
            "arguments": {}
          }
        },
        "eventRef": "cwm.forms.Check_applicant_age",
        "metadata": {
          "formData": "${ {user : .user} }",
          "taskName": "Provide applicant age"
        },
        "timeouts": {
          "eventTimeout": "PT15S",
          "stateExecTimeout": "PT15M",
          "actionExecTimeout": "PT15S"
        },
        "transition": "evaluate"
      },
      ...
 ]
```

The callback state requires the `action` parameter to be stated. The `name` and `functionRef` keys need to have values provided even if no action is expected to happen during this state. Therefore, a mock action needs to be provided. For this, you can use a utility function, like `noOp`, or any activity of your custom adapter, but keep in mind that the action needs to complete successfully for the workflow to pass to another step.

The key values here are:

- `eventRef`: provide the ID of the form.
- `taskName`: this can be an arbitrary name that refers to the task.

The next state in the example workflow is a **switch** state. It is where you provide conditions for the task input. If the age of the applicant is stated to be under 18, the application is rejected. If it is 18 or more, the application is approved.

```
         {
        "name": "evaluate",
        "type": "switch",
        "dataConditions": [
          {
            "name": "adult",
            "condition": "${ .provideApplicantAge = 18 }",
            "transition": "approve"
          },
          {
            "name": "underage",
            "condition": "${ .provideApplicantAge  18 }",
            "transition": "reject"
          }
        ],
        "defaultCondition": {
          "transition": "reject"
        }
      },
```

The key value here is `condition`. Here is where you refer to the **Property name** of the component that you can find under the **API** tab inside the component. The component property name needs to be referred to inside a jq expression that starts with `$`. The condition is stated in form of an inequality: `.provideApplicantAge = 18`.

The last two states would be the inject states where the workflow either accepts or rejects the application and prints the result in the job execution logs.

```
{
  "end": true,
  "data": {
    "application": "approved"
  },
  "name": "approve",
  "type": "inject"
},
{
  "end": true,
  "data": {
    "application": "rejected"
  },
  "name": "reject",
  "type": "inject"
}
```

Step 3: Run the workflow and submit task

If you have created a workflow definition based on the example, you may want to test it against the system. To do that, follow the steps below.

1. In CWM, choose **Design** > **Workflows**.

   Click **Create workflow**, provide a **Workflow definition name** and **Version** for the new workflow, and click **Create**.

   Click the **Designer** tab and, in the **Code** tab, paste your workflow definition. Click **Save**.

2. Find your workflow in the list of workflow definitions, enter it and click **Run**. In the dialog, click **Run job**.

3. Choose **Workflow Automation** > **Operate** > **Tasks** and look for the **Applicant age check** task (or whatever custom name you assigned to this task).

4. Click the task, provide input (in this example, it will be a number) in the field and click **Submit**.

5. The workflow will process the input given in the task and finish execution.

   You can see the result by selecting to **Workflow Automation** > **Operate** > **Jobs** and looking up the logs of your job.

Guided task: Create an L3 VPN

About this example

To complete and test the following example, you will need to have a running instance of Cisco Network Service Orchestrator (NSO).

This example creates an L3 VPN with the given parameters in Cisco NSO. Before executing the operation, CWM performs a dry run to simulate the configuration. During execution, the dry-run payload is displayed in the task prompt, allowing the user to either accept or reject the operation based on the previewed results.

Click here to download and extract the Create NSO L3VPN file, including the JSON form file.

Step 1: Create a form

For the purpose of this example, we will focus on creating the example form using the `POST/form` API endpoint.

1. Go to the CWM API in Postman and use the `POST/form` endpoint.
2. Copy the content of the JSON form file from the download and paste it into the request field.
3. Click **Send**. The form has been created in CWM.

Step 2: Define task in the workflow

Let's see how the task that will employ the created form is defined inside the workflow definition. In the workflow definition, let's start by setting up the events parameter with the data of the form we created.

```
"events": [
  {
    "kind": "consumed",
    "name": "cwm.forms.NSO_dry_run",
    "type": "cwm.forms.NSO_dry_run",
    "source": "system.internal.cwm"
  }
],
```

The key values here are:

- `kind`: The *event* is `consumed`, since the input provided in the task is treated as external and will be consumed by the workflow.
- `name`: It's the **Form ID** which you can find in the form details when selecting it from the **Forms** list.
- `type`: It takes the same value as the `name` key.
- `source`: It needs to be `system.internal.cwm` as the source.

Next, we want to include the `callback` state into the workflow and refer to the event (the form ID) that it will use to invoke the task. For this, we need to add the `eventRef` key and its accompanying metadata: `formData`, where you refer to the variable used to retrieve the result of the dry run operation.

Note that inside the callback state, we need to state the `action` parameter. Inside the `functionRef` key we state `refName` and give the name of the NSO adapter activity there: `NSO.restconfPost`.

```
"states": [
        {
            "name": "decision",
            "type": "callback",
            "action": {
                "name": "show-dry-run",
                "functionRef": {
                    "refName": "NSO.restconfPost",
                    "arguments": {
                        "input": {
                            "data": {
                                "l3vpn": [
                                    {
                                        "name": "${ .vpnName }",
                                        "endpoint": "${  [range( .endpoint | length) as $i
 | {id: .endpoint[$i].name,
\"ce-device\":.endpoint[$i].device, \"ce-interface\":\"GigabitEthernet0/2\" ,
\"ip-network\":.endpoint[$i].ipNetwork,
bandwidth: .endpoint[$i].bandwidth, \"as-number\":65002 }] }",
                                        "route-distinguisher": "${ .rd }"
                                    }
                                ]
                            },
                            "path": "restconf/data/l3vpn:vpn",
                            "queries": {
                                "additional": {
                                    "dry-run": "cli"
                                }
                            }
                        },
                        "config": {
                            "resourceId": "${ .nsoResource }"
                        }
                    }
                },
                "actionDataFilter": {
                    "toStateData": "${ .result }"
                }
            },
            "eventRef": "cwm.forms.NSO_dry_run",
            "metadata": {
                "formData": "${ { dryrunPre : .result.data } }",
                "taskName": "Approve/reject dry run operation"
            },
        },
    ]
```

The key values here are:

- `eventRef`: provide the ID of the form.
- `metadata`: use this key to decide what data will be presented in the task:
  - `formData`: build a jq expression to make the dry run result appear inside the task prompt. Note that the first element of the expression, `dryrunPre`, is what is provided in the HTML component in the form so that whatever the dry run result is, it will be displayed in the task. The other element is the part of the expression that specifies what data exactly is to be shown out of the whole payload of the dry run operation collected by the `toStateData` key inside `actionDataFilter`.
  - `taskName`: this can be an arbitrary name that refers to the task.

The next state in the example workflow is a **switch** state. It is where you provide conditions for the task input. If the configuration shown by the dry run result is correct, the operator approves the operation. If it is not correct, the operator rejects the change.

```
"name": "evaluate-decision",
"type": "switch",
"dataConditions": [
        {
            "name": "start-application",
            "condition": "${ .approve == true }",
            "transition": "commit-service"
         },
        {
            "name": "reject-application",
            "condition": "${ .approve == false}",
            "transition": "abandon-service"
        }
    ],
"defaultCondition": {
        "transition": "abandon-service"
    }
```

The key value here is `condition`. Here is where you refer to the **Property name** of the component that you can find under the **API** tab inside the component. The component property name needs to be referred to inside a jq expression that starts with `$`. The condition is stated in form of an equality: `.approve == true`.

The last two states would be the operation states where the workflow either creates the VPN or rejects the operation and prints the result in the job execution logs.

Step 3: Run workflow and submit task

If you have created a workflow definition based on the example, you will want to test it against the system.

1. In CWM, choose **Design** > **Workflows**.

   Click **Create workflow**, provide a **Workflow definition name** and **Version** for the new workflow, and click **Create**.

   Click the **Designer** tab and, in the **Code** tab, paste your workflow definition. Click **Save**.

2. Find your workflow in the list of workflow definitions, enter it and click **Run**. In the dialog, click **Run job**.

3. Choose **Operate** > **Tasks** and look for the **Approve/reject dry run operation** task (or whatever custom name you assigned to this task).

4. Click the task, provide your signature and click **Approve**.

   The workflow will process the input given in the task and finish execution. To see the result, choose **Workflow Automation** > **Operate** > **Jobs** and look up the logs of your job.