Cisco Crosswork Workflow Manager 2.0 Get Started Guide

First Published: 2024-12-12

Last Modified: 2025-06-01

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

Install CWM using Docker Installer Tool

This section contains the following topics:

Install CWM using Docker Installer Tool

The CWM 2.0 is installed on the Cisco Crosswork platform by first deploying the Crosswork OVA file using a Docker image on the VMware vCenter 7.0 (or higher) and then installing the CWM CAPP file using the installation script.

Prerequisites

- **VMware vCenter Server 7.0** (U3p or later) and **ESXi 7.0** (U3p or later). Refer to the Crosswork Network Controller 7.0 installation requirements for more details.
- Docker version 19 or higher.
- `sshpass` installed. For Mac, you can use `brew install sshpass`.
- `jq` installed. For Mac, you can use `brew install jq`.

Use script to Deploy Crosswork and CWM

Procedure

Step 1 In your Docker-capable machine, create a directory where you will store all the files you will use during this installation.

Note

If you are using a Mac, ensure that the directory name is in lower case.

Step 2 Download the OVA file containing the Crosswork Network Controller package from cisco.com to the directory you created. It will contain the Crosswork tar.gz CAPP file, the CWM .ova file, the `install.sh` installation script, the `configuration.json` file and Docker installer image tar.gz (along with this instruction).

Step 3 Import the Docker installer image by running the following command. Be sure to adjust the image name as needed:

```
docker image import <docker-image-name>.tar.gz your-image-name:your-tag
```

Step 4 Inside the directory, create a `.txt` file and paste the VMware installation template given below. For this instruction, we'll name the file `deployment.tfvars.txt` for example purposes.

```
Cw_VM_Image = ""    # Line added automatically by installer.
ClusterIPStack        = "IPv4"
DataIPNetmask         = "255.255.255.0"
DataIPGateway         = "192.168.1.1"
DNS                   = "DNS"
DomainName            = "domain_name"
CWPassword            = "your_crosswork_password"
VMSize                = "XLarge"
vm_sizes = {
        "xlarge" = {
                vcpus           = 24
                cpu_reservation = 24000
                //Memory in Mbytes
                memory = 128000
        }
}
NTP                   = "ntp.esl.cisco.com"
Timezone              = "Europe/Paris"
EnableSkipAutoInstallFeature = "True"
ManagementVIP     = "your_mgmt_vip"
ManagementIPNetmask = "255.255.255.0"
ManagementIPGateway = "your_mgmt_gateway"
ThinProvisioned = "true"
DataVIP           = "your_data_vip"
CwVMs = {
"0" = {
    VMName                = "your_VM_name",
    ManagementIPAddress = "your_mgmt_ip",
    DataIPAddress       = "your_data_ip",
    NodeType               = "Hybrid"
}
}
VCenterDC = {
VCenterAddress = "your_vcenter_address",
VCenterUser = "your_username",
VCenterPassword = "your_password",
DCname = "your_datacenter_name",
MgmtNetworkName = "VM Network",
DataNetworkName = "SVM Data Network"
VMs = [{
    HostedCwVMs = ["0"],
    Host = "your_VM_host",
    Datastore = "your_VM_datastore",
    HSDatastore = "your_VM_hsdatastore"
  }
]
}
SchemaVersion = "7.1.0"
```

Note

Note the difference between your VCenter and Datacenter.

Step 5 Edit the parameters to match your deployment.

Note

To learn more about the installation parameters, please refer to the Single VM chapter in the Cisco Crosswork Network Controller 7.0 Installation Guide.

Step 6 Inside the directory, create another file named `product.json` file and paste the data below.

```
{
"product_id": "CWM",
"attribute": {
    "key1": "value1",
    "key2": "value2"
}
}
```

Step 7 Open the `configuration.json` file and provide the following parameters to match your deployment:

```
{
    "SVM_NAME": "your_VM_name",
    "host": {
        "remote_user": "your_username",
        "remote_password": "your_password",
        "remote_host": "your_scp_host",
        "remote_port": "22",
        "capp_file": "/path/to/capp_file.tar.gz"
    },

    "cwm_login": {
        "ip": "your_mgmt_ip",
        "cwm_user": "admin",
        "cwm_old_password": "admin",
        "cwm_password": "your_new_password"
    },
    "deployment": {
        "tfvars_path": "/path/to/deployment.tfvars.txt",
        "ova_file": "/path/to/cwm.ova",
        "product_json": "/path/to/product.json"
    }
}
```

a) for `host`, provide the details of the SCP server where your Crosswork CAPP file is located like host address and port, your username and password, and the path to the file.
b) for `cwm_login`, provide your management IP and the default Crosswork username and password. In `cwm_password`, provide the new password to replace the default one upon installation completion.
c) for `deployment`, provide the local paths to the `deployment.tfvars.txt` created in a previous step, to the CWM OVA file and to the `product.json` file.

Step 8 From the directory, run the installer script:

```
bash install.sh
```

This will start the installation process for the Crosswork platform and then for CWM once the platform is deployed.

Step 9 To follow the installation inside the Docker container, run the following command:

```
sudo docker ps -a
```

Copy the ID of the container in which the installation started. Usually its name contains the OVA filename, such as: `cw-na-cwm-7.1.0-20-releasecnc710-250512-cwm-59-50`

To see the logs, run:

```
sudo docker logs your_container_id -f
```

Step 10 Once the installation script is done and the deployment status reaches 100%, go to http://your_mgmt_vip_address:30603 and log in with the default `admin` user and the password you provided in `configuration.json`.

CHAPTER 2

What is Crosswork Workflow Manager

This section covers the following topics:

What is Cisco Crosswork Workflow Manager?

Cisco Crosswork Workflow Manager (CWM) is a tool that simplifies and automates complex network operations and other business processes. It provides a centralized platform for creating, managing, and executing workflows, allowing for manual operator intervention during workflow execution while ensuring repeatability and fault-tolerance.

Workflows are defined using a standardized Domain Specific Language based on the Serverless Workflow specification, enabling workflow designers to express complex business processes, dependencies, and decision logic in a unified and readable format.

CHAPTER 3

Core Concepts

This section covers the following topics:

Core Concepts

The following topics explain the main concepts and components used in CWM. Understanding these concepts will help you understand how the platform works and how to use it.

Activity

An activity is a CWM function that executes a single, specific action on an external system, such as another application or solution. You define activities in adapters, which allow communication with the outside system.

Adapter

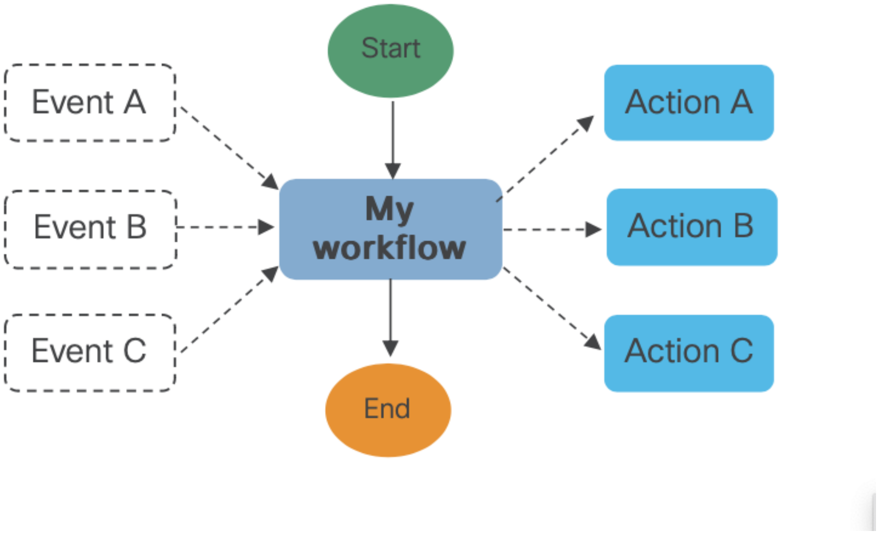
Adapters are responsible for communication with external services, such as other applications, systems, or environments. The adapters define and expose activities that are consumed by workflow definitions. Each adapter can be associated with the worker that executes the adapter activities.

Adapter SDK

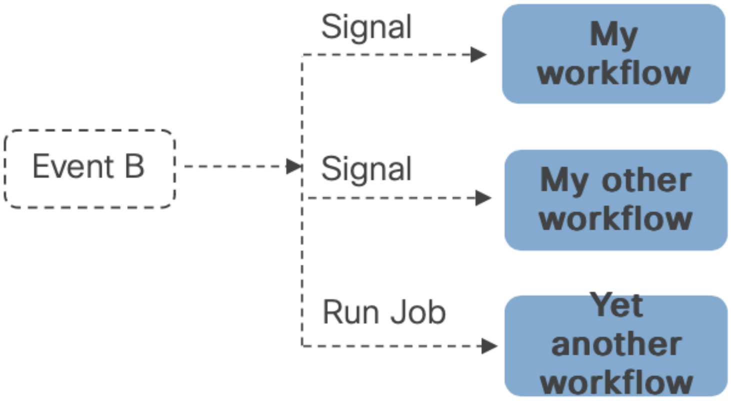
The adapter SDK automatically generates the structure for the required adapter's components. Developers can further define necessary activities they need and then extend integrations with the client environment.

Event

Events are signals from external sources with which CWM workflows can interact. This version of CWM adds support for an external Kafka data broker, so Kafka events can be either consumed or produced by an instantiated workflow job. A workflow can listen on one or multiple events and consume them to trigger one or more actions, as shown below:



An event coming into the system can also trigger a workflow, as shown below:



Execution engine

CWM uses an internal worker called the execution engine. It enables the execution of all other workflow definitions and is not visible to you in the user interface.

Job

A job represents the single execution of a particular workflow definition. To run a job in CWM, you first need to add your workflow definition. Running a new job instantiates a workflow definition.

Before starting a job run, you enter the initial start data (input variables). This ensures that your workflow executions are isolated and may use different data from other executions of the same workflow definition.

Job event

Events are created during workflow execution based on the occurrences defined in the workflow definition. The Job Event Log table in the user interface records all events that occurred during workflow execution.

Schedule

Scheduling a job allows you to define a specific start time and date for a workflow, either once or on a recurring schedule. You can create a scheduled job using the CWM user interface or the API. Currently, some of the scheduling functions, such as editing or pausing/unpausing are available only through the API. Each scheduled job is a separate entity with a unique Run ID, but all runs within a schedule share the same Schedule ID.

Worker

Workers execute the workflow definition code, relevant adapter code, and activities defined in the workflow. Depending on your needs and scale, you can have multiple workers for each workflow definition. Your worker can be associated with one adapter and its activities or with multiple ones.

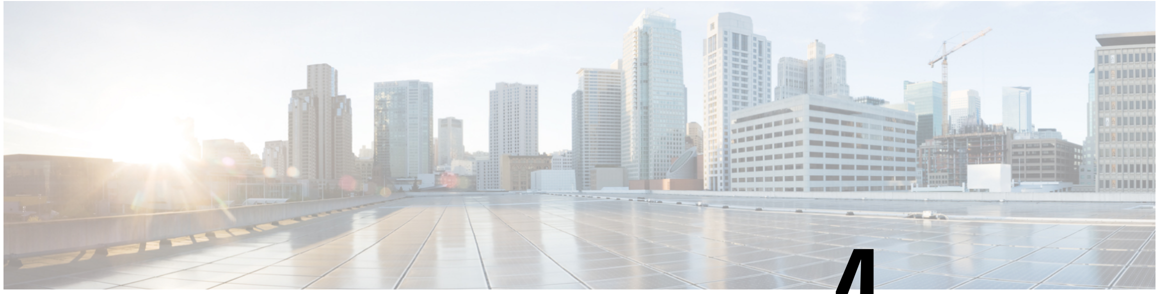
Workflow

Workflows help you capture, organize and automate processes with repeatable actions performed in a specified order. In the context of CWM, "workflow" can refer interchangeably to:

- **workflow definitions**: A workflow definition is a segment of code, written in JSON or YAML, that is based on the Serverless Workflow Specification and a vendor-neutral, domain-specific language.
- **workflow jobs**: A workflow job is single execution of a workflow definition.

Workflow engine

The workflow engine manages the way your workflow definitions are interpreted and conducted. It receives events, schedules tasks, and manages the execution of workflows.



CHAPTER 4

Example Workflows

This section covers the following topics:

Cisco NSO adapter workflow

This quick start uses a locally installed Cisco Crosswork Network Service Orchestrator (NSO) application and CWM with the Cisco NSO adapter to show you a basic use-case scenario for creating and running a successful workflow. It will guide you on installing an adapter, creating a worker for the workflow execution, and running the created workflow to quickly get tangible results in Cisco NSO.

NSO VPN Service Workflow overview

The purpose of this example workflow is to automatically create a VPN service for two NSO devices.

First, we point to the devices in the data input and then try to perform the NSO `check-sync` operation on them. Then, depending on the result:

- If not in sync, we push a device to perform a `sync-from`, and only then try to create a VPN for it;
- If in sync, we don't perform `sync-from` but directly create a VPN for the device.

If all the steps are executed successfully, the execution engine reports workflow execution completion and displays the final data input. The results are visible in NSO too. If the engine encounters errors while performing a step, it uses the specified `retry` policy. In case errors persist beyond the retry limits, the engine ends the execution with a **Failed** status.

Go through the following topics to learn the details of how data input, functions, states, actions, and data filters are defined.

Prerequisites

- Cisco NSO 6.1.0 install. If you do not have it, follow the installation instructions.
- CWM 2.0 installed on Cisco Crosswork Network Controller 7.1.0 or later.

Step 1: Install NSO adapter

To interact with Cisco NSO, CWM needs a dedicated Cisco NSO adapter. Here is the process to install it using the CWM API:

Upload NSO adapter file

Procedure

	Command or Action	Purpose
Step 1	Get the latest NSO adapter installation file from the CWM software package.	
Step 2	In CWM, choose **Administration** > **Workflow Administration** > **Adapters**.	
Step 3	Click **Add Adapter**.	
Step 4	In the **Install adapter** window, click on the file uploader to select a `tar.gz` installable archive from your local machine and click **Upload**.	
Step 5	After the adapter file is uploaded to the database, check the **Automatically create worker for this adapter** checkbox if you want to create a worker, and click **Install Adapter** to finish the installation process.	
Step 6	In the adapter list, click on the name of your adapter to enter its details. Set the **Use as default version** option to **True**.	

Step 2: Create secret and resource

To define the resources and secrets to pass securely to the Cisco NSO adapter, create a secret and resource using the CWM API.

Create a secret

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Administration** > **Workflow Administration** > **Secrets**.	
Step 2	Click **Add Secret**.	
Step 3	In the **New secret** view, specify the following:	• **Secret ID**: `NSOSecret` • **Secret type**: `basicAuth`

	Command or Action	Purpose
Step 4	After selecting the secret type, a set of additional fields is displayed in the Secret type details section. Fill in the fields with the following:	• **password**: `admin` • **username**: `admin`
Step 5	Click **Create Secret**.	

Create a resource

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Administration** > **Workflow Administration** > **Resources**.	
Step 2	Click **Add Resource**.	
Step 3	In the **New resource** window, specify the following:	• **Resource name**: `NSOLocal` • **Resource type**: `cisco.nso.resource.v1.0.0` • **Secret ID**: `NSOSecret` • **Connection**: • **Host**: `127.0.0.1` (or, replace with the address where you host the NSO instance) • **Port**: `8080` (or, replace with the port where the NSO web UI is available) • **Scheme**: `http` • **Timeout**: `60` • **Allow Insecure**: `true`
Step 4	Click **Create resource**.	

Step 3: Set up the NSO example service

In this workflow example, we set up a Layer3 VPN in a service provider's MPLS network using two NSO-simulated devices.

Procedure

	Command or Action	Purpose
Step 1	In a terminal, open your main NSO directory and go to `mpls-vpn-new-template`:	`cd examples.ncs/service-provider/mpls-vpn-new-template`

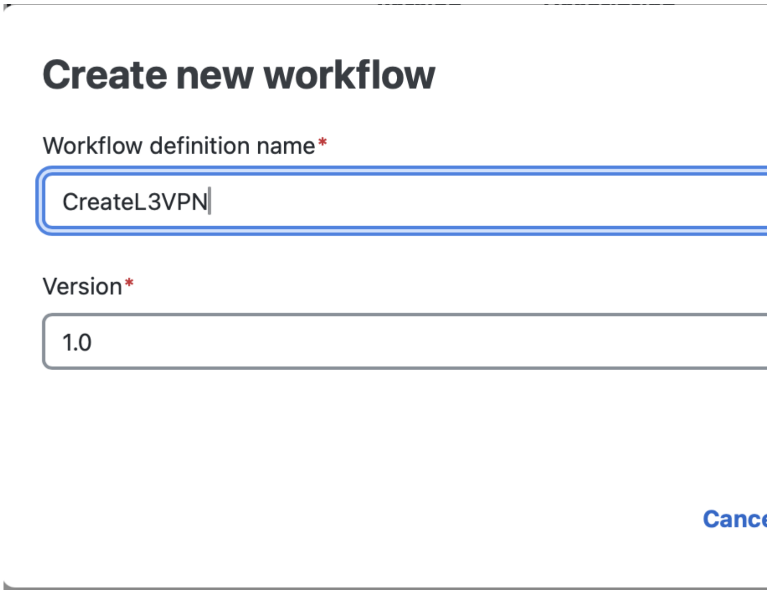
	Command or Action	Purpose
Step 2	Execute the makefile by running:	`make stop clean all start` This command will start your local NSO instance and the sample netsim devices.
Step 3	For the example workflow to execute successfully, execute a **sync-from** on all the netsim devices beforehand:	

Step 4: Run the workflow

Now that we have the NSO adapter, the worker, and the NSO example all up and running, we can create a workflow in the CWM user interface and run the job.

Add new workflow

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Design** > **Workflows**.	
Step 2	Click **Create new workflow**.	
Step 3	In the **Create workflow** window, provide the required input:	• **Workflow definition name**: The example workflow definition (`CreateL3VPN`). • **Version**: The workflow definition version (`1.0`).
Step 4	Click **Create workflow**.	 Create new workflow Workflow definition name* CreateL3VPN Version* 1.0

Run job

Procedure

	Command or Action	Purpose
Step 1	In the **Workflows** panel, enter the newly created workflow definition by clicking its name.	
Step 2	Click the **Designer** tab, select **Code** and delete the sample content from the **Code** field.	
Step 3	Copy the workflow definition from the codeblock below and paste it inside the **Code** field, then click **Save changes**.	

```
{
  "id": "CreateL3VPN-1.0",
  "name": "CreateL3VPN",
  "start": "start",
  "states": [
    {
      "name": "start",
      "type": "operation",
      "actions": [
        {
          "name": "checkSync",
          "retryRef": "Default",
          "functionRef": {
            "refName": "NSO.RestconfPost",
            "arguments": {
              "input": {
                "path":
"restconf/operations/tailf-ncs:devices/device=${
.device0Name }/check-sync"
              },
              "config": {
                "resourceId": "${ .nsoResource }"
              }
            }
          },
          "actionDataFilter": {
            "results": "${ if (.data) then .data
| .\"tailf-ncs:output\".result else null end }",
            "toStateData": "${ .checkSyncResult0
}"
          }
        },
        {
          "name": "checkSync",
          "retryRef": "Default",
          "functionRef": {
            "refName": "NSO.RestconfPost",
            "arguments": {
              "input": {
                "path":
"restconf/operations/tailf-ncs:devices/device=${
.device1Name }/check-sync"
              },
              "config": {
                "resourceId": "${ .nsoResource }"
              }
            }
          },
          "actionDataFilter": {
            "results": "${ if (.data) then .data
```

	Command or Action	Purpose

```
| .\"tailf-ncs:output\".result else null end }",
            "toStateData": "${ .checkSyncResult1
}"
          }
        }
      ],
      "transition": {
        "nextState": "syncFromOrCreateVPN"
      },
      "stateDataFilter": {
        "input": "${ . }"
      }
    },
    {
      "name": "syncFromOrCreateVPN",
      "type": "switch",
      "dataConditions": [
        {
          "name": "shouldSyncFrom",
          "condition": "${ if (.checkSyncResult0)
 then  .checkSyncResult0 != \"in-sync\" else null
 end }",
          "transition": {
            "nextState": "syncFrom"
          }
        },
        {
          "name": "shouldCreateVPN",
          "condition": "${ if (.checkSyncResult0)
 then  .checkSyncResult0 == \"in-sync\" else null
 end }",
          "transition": {
            "nextState": "createVPN"
          }
        },
        {
          "name": "shouldSyncFrom",
          "condition": "${ if (.checkSyncResult1)
 then  .checkSyncResult1 != \"in-sync\" else null
 end }",
          "transition": {
            "nextState": "syncFrom"
          }
        },
        {
          "name": "shouldCreateVPN",
          "condition": "${ if (.checkSyncResult1)
 then  .checkSyncResult1 == \"in-sync\" else null
 end }",
          "transition": {
            "nextState": "createVPN"
          }
        }
      ],
      "defaultCondition": {
        "end": {
          "terminate": true
        }
      }
    },
    {
      "name": "syncFrom",
```

	Command or Action	Purpose

```
      "type": "operation",
      "actions": [
        {
          "name": "syncFrom",
          "retryRef": "Default",
          "functionRef": {
            "refName": "NSO.RestconfPost",
            "arguments": {
              "input": {
                "path":
"restconf/operations/tailf-ncs:devices/device=${
.device0Name }/sync-from"
              },
              "config": {
                "resourceId": "${ .nsoResource }"
              }
            }
          },
          "actionDataFilter": {
            "results": "${ if (.data) then .data
| .\"tailf-ncs:output\".result else null end }",
           "toStateData": "${ .syncFromResult0 }"

          }
        },
        {
          "name": "syncFrom",
          "retryRef": "Default",
          "functionRef": {
            "refName": "NSO.RestconfPost",
            "arguments": {
              "input": {
                "path":
"restconf/operations/tailf-ncs:devices/device=${
.device1Name }/sync-from"
              },
              "config": {
                "resourceId": "${ .nsoResource }"
              }
            }
          },
          "actionDataFilter": {
            "results": "${ if (.data) then .data
| .\"tailf-ncs:output\".result else null end }",
           "toStateData": "${ .syncFromResult1 }"

          }
        }
      ],
      "transition": {
        "nextState": "createVPN"
      }
    },
    {
      "end": {
        "terminate": true
      },
      "name": "createVPN",
      "type": "operation",
      "actions": [
        {
          "name": "createVPN",
```

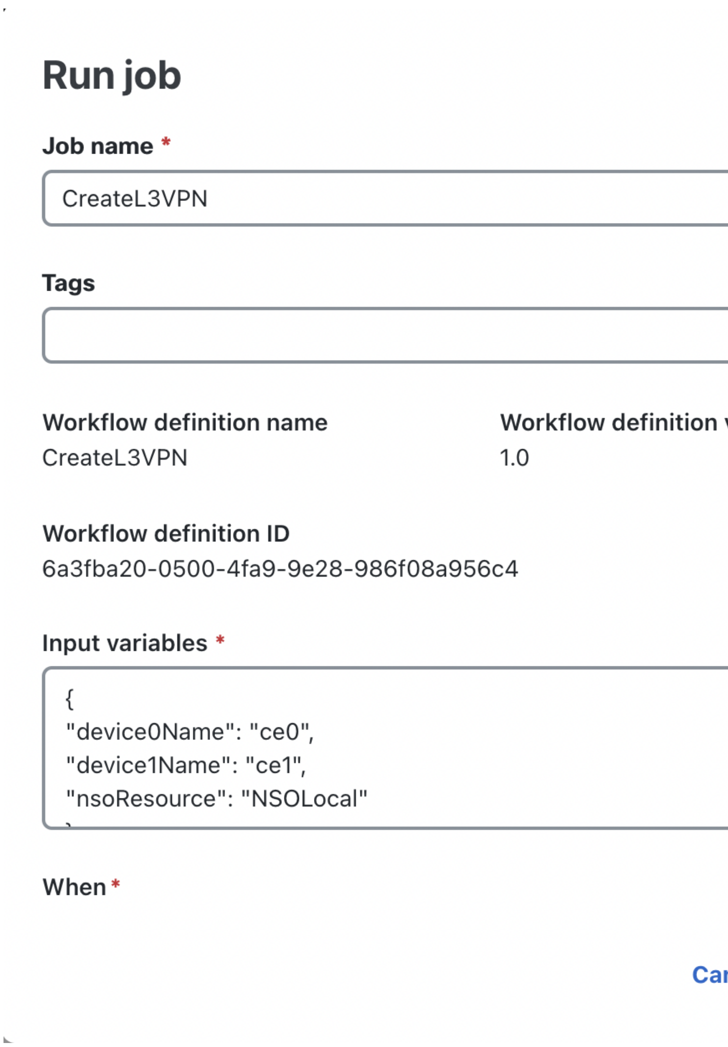
	Command or Action	Purpose

```
          "retryRef": "Custom",
          "functionRef": {
            "refName": "NSO.RestconfPost",
            "arguments": {
              "input": {
                "data":
[illegible]
 ",
                "path": "restconf/data/l3vpn:vpn"
              },
              "config": {
                "resourceId": "${ .nsoResource }"
              }
            }
          },
          "actionDataFilter": {
            "results": "${ if (.status) then
.status else null end }",
          "toStateData": "${ .createServiceResult
 }"
          }
        }
      ]
    }
  ],
  "retries": [
    {
      "name": "Default",
      "delay": "PT30S",
      "multiplier": 2,
      "maxAttempts": 4
    },
    {
      "name": "Custom",
      "delay": "PT10S",
      "multiplier": 1,
      "maxAttempts": 2
    }
  ],
  "version": "1.0",
  "functions": [
    {
      "name": "NSO.RestconfPost",
     "operation": "cisco.nso.v1.0.3.restconf.Post"

    }
  ],
  "description": "",
  "specVersion": "0.9"
}
```

Step 4	Click **Run**.	

Step 5	In the **Run job** view, provide a name for the job and in the **Input** field, paste the data input from the section below inside the brackets:	`"device0Name": "ce0",`
		`"device1Name": "ce1",`
		`"nsoResource": "NSOLocal"`

	Command or Action	Purpose
Step 6	Click **Run job**.	 Run job Job name * CreateL3VPN Tags Workflow definition name CreateL3VPN Workflow definition v 1.0 Workflow definition ID 6a3fba20-0500-4fa9-9e28-986f08a956c4 Input variables * { "device0Name": "ce0", "device1Name": "ce1", "nsoResource": "NSOLocal" When *

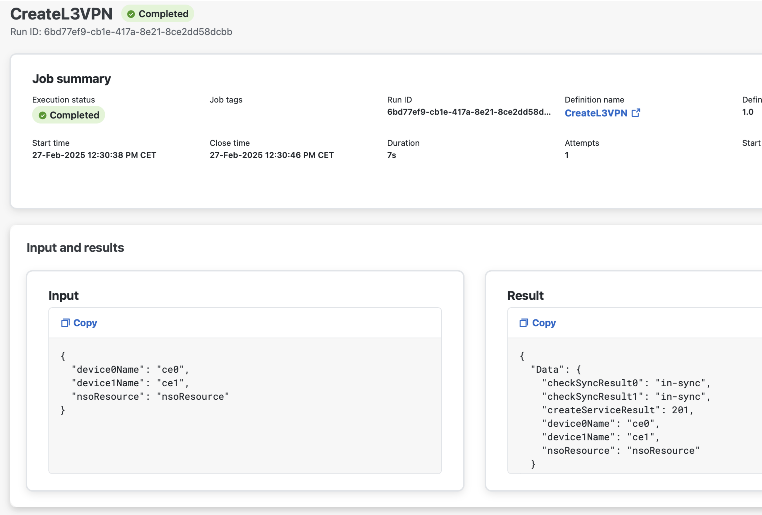
Step 5: Check results

Workflows, by nature, bridge systems and applications. You can verify the results of this workflow by checking your results in two places:

- in CWM UI, or
- In NSO Service Manager, on page 20

In the CWM User Interface

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Workflow Automation** > **Operate** > **Jobs**.	
Step 2	In the **Jobs** view, find your job and check the status of the workflow execution in the **Status** table column:	If the workflow is executed correctly, a green checkmark with **Completed** status will be visible. If the workflow execution is still in progress or the engine is retrying an action, a blue label with the **Running** status will be displayed.
Step 3	Click the job name to view its details.	
Step 4	Expand the **Input and result** section by clicking its name.	
Step 5	The **Result** card presents the final data output updated by the successful execution of the workflow actions for which `toStateData` inside the `actionDataFilter` was defined:	

CreateL3VPN ✓ Completed

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

Job summary

Execution status	Job tags	Run ID	Definition name	Defin
✓ Completed		6bd77ef9-cb1e-417a-8e21-8ce2dd58d...	CreateL3VPN	1.0
Start time	**Close time**	**Duration**	**Attempts**	**Start**
27-Feb-2025 12:30:38 PM CET	27-Feb-2025 12:30:46 PM CET	7s	1	

Input and results

Input

Copy

```
{
  "device0Name": "ce0",
  "device1Name": "ce1",
  "nsoResource": "nsoResource"
}
```

Result

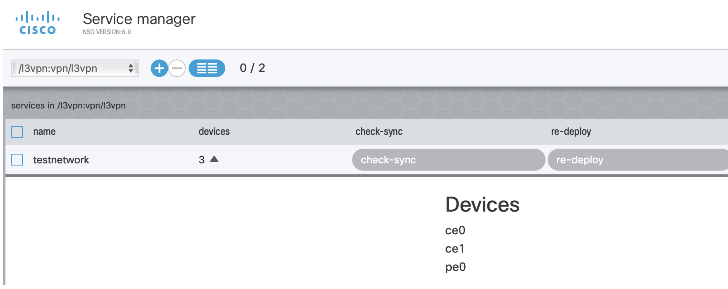
Copy

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

In NSO Service Manager

Procedure

	Command or Action	Purpose
Step 1	Log in to your NSO account.	
Step 2	In the **Application hub** view, click the **Service manager** tile.	
Step 3	From the **Select service points** drop-down, select **/l3vpn:vpn/l3vpn**.	

	Command or Action	Purpose
Step 4	In the table, find `testnetwork` and click the **devices** arrow to see that your netsim devices `ce0` and `ce1` now belong to the `testnetwork`, together with a `pe0` device.	 Service manager NSO VERSION:6.0 /l3vpn:vpn/l3vpn 0 / 2 services in /l3vpn:vpn/l3vpn □ name / devices / check-sync / re-deploy □ testnetwork / 3 ▲ / check-sync / re-deploy Devices ce0 ce1 pe0

NetBox, NSO and Webex as child workflows

This workflow example explores the possibilities of using multiple external services in a workflow, and calling them by means of separate subworkflows (called *child workflows*) for each service. The aim of the workflow is to automatically allocate subnet prefixes in NetBox, spin up a VPN service instance in Cisco NSO with network endpoint configurations, and send a confirmation message through Cisco Webex when the workflow is completed.

In this example, there is one main workflow (parent) and three subworkflows (children). This modularity allows you to try each child workflow separately without setting up and running the others."

Using child workflows allows you to encapsulate each service (NetBox, NSO, and Webex) in its workflow logic, making debugging and updates easier. The parent workflow in turn, controls the overall process and ensures the child workflows execute in sequence. It also handles inter-workflow data passing and monitors the overall status.

Prerequisites

- Cisco NSO 6.1 or later, with an example service set up (`mpls-vpn-template` or other).
- NetBox version 4.1 or later.
- a Webex account with a personal room.

Download the main workflow

Download and extract the the NSO NetBox Workflow file from Cisco DevNet here. This download requires a Cisco DevNet login, and contains the complete JSON workflow file, plus all sub-workflow files and input data.

The main workflow coordinates subworkflows to perform actions in external services and fills in the sequence with auxiliary actions. After the first subworkflow allocates prefixes in NetBox, it performs a `sync from` on specific NSO devices and creates an L3VPN service for them.

For the NSO part of the workflow, we'll need to install the Cisco NSO adapter and create a secret and a resource in CWM.

Install NSO adapter

To install the Cisco NSO adapter, locate the latest adapter `tar.gz` file on your machine and follow the steps for installing adapters given in the Operator Guide.

Create NSO secret

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Administration** > **Workflow Administration** > **Secrets** tab.	
Step 2	Click **Add Secret**.	
Step 3	In the **New secret** view, specify the following:	
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:	
Step 5	Click **Create Secret**.	

Create an NSO resource

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Administration** > **Workflow Administration** > **Resources** tab.	
Step 2	Click **Add Resource**.	
Step 3	In the **New resource** view, specify the following:	• **Resource name**: `NSOResource` • **Resource type**: `cisco.nso.resource.v1.0.0` • **Secret ID**: `NSOSecret` • **Connection**: • **Host**: Provide the address where your NSO instance is hosted. • **Port**: Provide the port on which the NSO web UI is available. • **Scheme**: `http` • **Timeout**: `60` • **Allow Insecure**: `true`

	Command or Action	Purpose
Step 4	Click **Create resource**.	

NetBox subworkflow #1

This subworkflow involves allocating a subnet in NetBox, which will subsequently be used in the configuration of an L3VPN. The communication with NetBox will use the Generic REST adapter. Therefore, the exact resource path and payload must be clearly defined. Specifically, this example requires a POST request to the `/api/ipam/prefixes/` endpoint in NetBox, with the prefix and description provided in the request body.

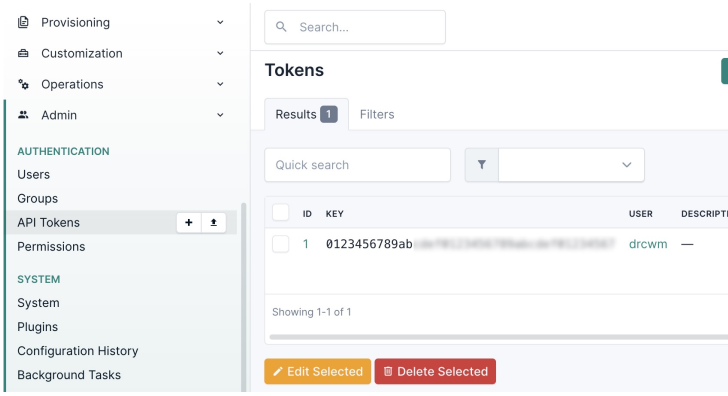
Install Generic REST adapter for NetBox

To install the Generic REST adapter, locate the latest adapter `tar.gz` file on your machine and follow the steps in the "Install Adapter" topic in the Operator Guide.

Create NetBox secret

To authorize NetBox in CWM, you must first retrieve the token for your NetBox installation and then add it in the secret.

Procedure

	Command or Action	Purpose
Step 1	Log in to NetBox.	
Step 2	In the left menu, click the **Admin** tab to expand it and select **API Tokens**.	
Step 3	Add a new token and copy it or, if a token exists, copy it to the clipboard.	 Provisioning Customization Operations Admin AUTHENTICATION Users Groups API Tokens Permissions SYSTEM System Plugins Configuration History Background Tasks Search... Tokens Results 1 Filters Quick search Edit Selected Delete Selected ID KEY USER DESCRIPT 1 0123456789ab[redacted] drcwm — Showing 1-1 of 1
Step 4	Log in to CWM and select **Administration** > **Workflow Administration** > **Secrets** tab.	
Step 5	Click **Add Secret**.	
Step 6	In the **New secret** view, specify the following:	- **Secret ID**: `NetBoxSecret` - **Secret type**: `token`

	Command or Action	Purpose
Step 7	In the **token** field, provide your NetBox token that you've copied.	
Step 8	Click **Create Secret**.	

Create NetBox resource

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Administration** > **Workflow Administration** > **Resources** tab.	
Step 2	Click **Add Resource**.	
Step 3	In the **New resource** window, specify the following:	• **Resource name**: `NetBoxResource` • **Resource type**: `generic.rest.resource.v1.0.0` • **Secret ID**: `NetBoxSecret` • **Connection**: • **Host**: provide the address where your NetBox instance is hosted. • **Port**: provide the port on which the NetBox web UI is available. • **Scheme**: `http` • **Timeout**: `60` • **Allow Insecure**: `true`
Step 4	Click **Create resource**.	

NSO subworkflow #2

This subworkflow uses the function `NSO.RestconfPost` to interact with the NSO RESTCONF API. It iterates over a collection of network endpoints to configure each one. For each endpoint, it sends a POST request with the endpoint-specific data (including device ID, interface, IP network, and bandwidth) to the l3vpn:vpn resource for the VPN service created by the main workflow (if you run the full example). The results of each configuration action are filtered and stored in an output collection called `endpointsConfigureResponses`.

To enable communication between CWM and NSO, you need the NSO adapter, secret, and resource. If already set up during the main workflow configuration, you can re-use them in the subworkflow without changes.

Note If you only run this subworkflow separately, you need to add an L3VPN service to your NSO instance. You can do this manually using the NSO CLI by running these commands:

```
ncs_cli -C -u admin
vpn l3vpn network1
route-distinguisher 999
```

Webex subworkflow #3

The purpose of this child workflow is to notify a Webex user in a Webex room about the completion status of a workflow. It consists of two actions. Both use the REST Post function to send messages to the Webex API.

The first action posts a message to the specified Webex room (roomId) stating "Workflow completed." The second action posts a status message to the same room, with the content depending on the terminate flag. If terminate is true, the message is "Status: Failed"; otherwise, it's "Status: Success."

Both actions send requests to the `v1/messages/` endpoint of the Webex API and use **webex_room** as the resource configuration. The response data for each message is stored in webexResponse.

Note If you run the subworkflow independently from the main workflow, you'll need a reduced version of the input data. Remember to replace the "roomId" value with your personal room ID. Follow the instructions in Webex subworkflow #3, on page 25 to learn how to retrieve it.

Install Generic REST adapter for Webex

To communicate with Webex, add the Generic REST adapter and create a Webex secret and resource in CWM. To install the adapter, locate the latest Generic REST adapter `tar.gz` file on your machine and follow the steps in the "Install Adapter" topic in the Operator Guide.

Create Webex secret

To authorize Webex in CWM, you first need to retrieve the bearer from the Webex API and the room ID of your personal room.

Procedure

	Command or Action	Purpose
Step 1	Go to developer.webex.com/docs/getting-started and log in (or sign up).	
Step 2	From the **Your Personal Access Token** field, copy the bearer.	
Step 3	Log in to CWM and select **Administration** > **Workflow Administration** > **Secretss** tab.	

	Command or Action	Purpose
Step 4	Click **Add Secret**.	
Step 5	In the **New secret** view, specify the following:	- **Secret ID**: `webex_secret` - **Secret type**: `bearer`
Step 6	In the **bearer** field, provide the Webex bearer that you've copied.	
Step 7	Click **Create Secret**.	

Create a Webex resource

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Administration** > **Workflow Administration** > **Resources** tab.	
Step 2	Click **Add Resource**.	
Step 3	In the **New resource** window, specify the following:	- **Resource name**: `WebexResource` - **Resource type**: `generic.rest.resource.v1.0.0` - **Secret ID**: `webex_secret` - **Connection**: - **Host**: `webexapis.com`. - **Port**: `443`. - **Scheme**: `https` - **Timeout**: `60` - **Allow Insecure**: `true`

Run the main workflow

You can then run the main workflow:

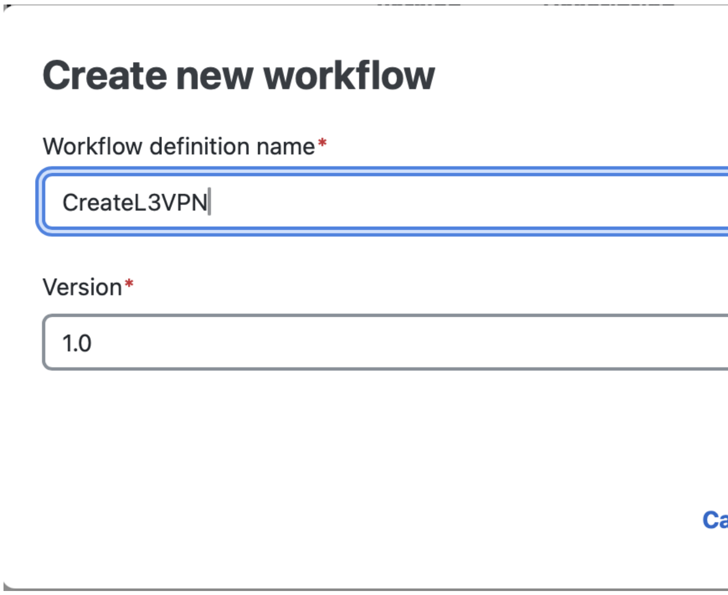
Procedure

	Command or Action	Purpose
Step 1	If needed: In CWM, select **Design** > **Workflows**.	
Step 2	In the **Workflows** view, enter the newly created workflow definition by clicking its name.	

	Command or Action	Purpose
Step 3	Click **Run**.	
Step 4	In the **Run job** window, provide a name for the job and in the **Input** field, paste the data input that you have updated with your Webex room ID.	
Step 5	Click **Run job**.	

Add workflows

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Design** > **Workflows** tab.	
Step 2	Click **Create workflow**.	
Step 3	In the **Create workflow** window, enter:	- **Workflow definition name**: Provide the name for the example workflow definition: `NetBox_NSO_Webex_example`. - **Version**: Provide workflow definition version: `1.0`.
Step 4	Click **Create workflow**.	 Create new workflow Workflow definition name* CreateL3VPN Version* 1.0
Step 5	Enter the newly created workflow definition by clicking its name.	
Step 6	Click the **Designer** tab, select **Code** and delete the sample content from the **Code** field.	
Step 7	Copy the downloaded workflow definition and paste it inside the **Code** field, then click **Save changes**.	

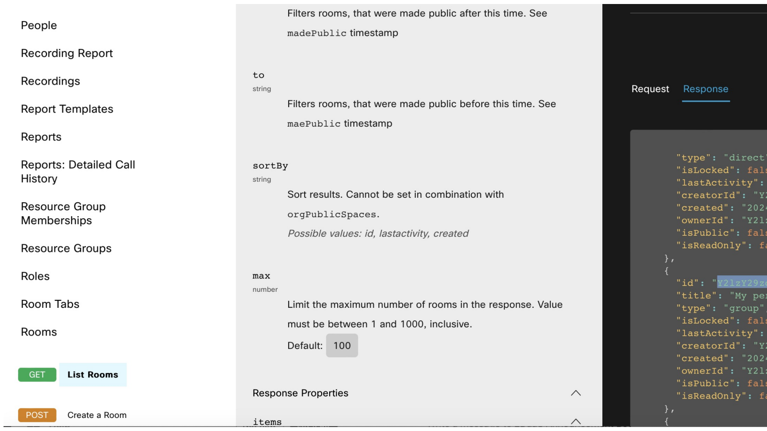
	Command or Action	Purpose
Step 8	Repeat this process for the remaining three subworkflows to add them.	

Run job

Before you run the job, ensure that you fill in the value of the `roomId` key correctly. To do that, retrieve the room ID of your personal room in Webex.

Retrieve the Webex room ID

Procedure

	Command or Action	Purpose
Step 1	Point your browser to https://developer.webex.com/calling/docs/getting-started and either log in or sign up.	
Step 2	Once you are logged in, point your browser to the List rooms endpoint in the API Reference: https://developer-usgov.webex.com/docs/api/v1/rooms/list-rooms	
Step 3	Click **Run**.	
Step 4	In the **Response** field, find `My personal room` and copy its `"id"`.	
Step 5	Paste the Room ID in the input data as the value of the `roomId` key.	

Filters rooms, that were made public after this time. See `madePublic` timestamp

`to`
string

Filters rooms, that were made public before this time. See `maePublic` timestamp

`sortBy`
string

Sort results. Cannot be set in combination with `orgPublicSpaces`.

Possible values: id, lastactivity, created

`max`
number

Limit the maximum number of rooms in the response. Value must be between 1 and 1000, inclusive.

Default: 100

Response Properties

`items`

```
      "type": "direct
      "isLocked": fals
      "lastActivity":
      "creatorId": "Y2
      "created": "2024
      "ownerId": "Y2l
      "isPublic": fals
      "isReadOnly": fa
    },
    {
      "id": "Y2lzY29z
      "title": "My per
      "type": "group",
      "isLocked": fals
      "lastActivity":
      "creatorId": "Y2
      "created": "2024
      "ownerId": "Y2l
      "isPublic": fals
      "isReadOnly": fa
    },
    {
```

Run the main workflow

You can then run the main workflow:

Procedure

	Command or Action	Purpose
Step 1	If needed: In CWM, select **Design** > **Workflows**.	

	Command or Action	Purpose
Step 2	In the **Workflows** view, enter the newly created workflow definition by clicking its name.	
Step 3	Click **Run**.	
Step 4	In the **Run job** window, provide a name for the job and in the **Input** field, paste the data input that you have updated with your Webex room ID.	
Step 5	Click **Run job**.	

Check Results in CWM

Procedure

	Command or Action	Purpose
Step 1	In CWM, select **Operate** > **Jobs**.	
Step 2	In the **Jobs** view, find your job and check the status of the workflow execution in the **Status** table column.	
Step 3	Click the job name to view its details.	
Step 4	Expand the **Input and Results** entry by clicking its name to see the output of your workflow.	