Cisco Crosswork Workflow Manager Solutions 2.0 Golden Configuration Guide

First Published: 2025-06-25

Last Modified: 2025-06-25

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

CHAPTER 1

Golden Configuration

This section contains the following topics:

Golden Configuration Guide

May 2025

Copyright

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices**.

Preface

Abstract

This document is the user guide for the standalone version of the Cisco Crosswork Workflow Manager Solutions Golden Configuration package.

Audience

This document describes how to configure and use Crosswork Workflow Manager Solutions Golden Configuration. This document is intended for Cisco Advanced Services developers, network engineers, and system engineers who configure and deliver Crosswork Workflow Manager Solutions functionalities to Cisco customers.

Additional Documentation

This documentation requires the reader to have a good understanding of Cisco Crosswork and Cisco NSO and its use, as described in the Cisco documentation. For more information on NSO products, go to: https://developer.cisco.com/docs/nso/.

Cisco Crosswork Workflow Manager Solutions

CWM Solutions is a collection of common use cases designed to make field customizations simple and straightforward. It is built using **Cisco Crosswork Workflow** Manager (CWM) and **Cisco Network Services Orchestrator** (NSO).

This document explains how to use Golden Configuration to improve the efficiency and accuracy of network device configuration.

Note: Click these links for more information using on Cisco CWM and Cisco NSO.

Golden Configuration Package

The CWM Solutions Golden Configuration use case is a functional package that ensures device configurations adhere to standard configurations (“golden configurations”) established by your organization. It uses device templates to identify any differences between a device’s configuration and the standard configuration, and to apply the standard when deviations are detected.

Golden Configuration (GC)

The Golden Configuration (GC) application enables you to apply different templates in any format and bring up the network device configuration you need. GC enables you to establish a baseline configuration to activate on the network devices using templates in several supported styles: C-style (that is, Cisco XR style), J-style

(Juniper curly-bracket style), JSON, XML, native, and device template formats. These differing template styles provide the flexibility needed when working with a wide variety of network devices.

GC uses these two objects when upgrading network devices:

- Templates: Configure a template with the variables.
- Applications: Apply the values to the template variables.

Using Golden Configuration templates reduces configuration errors, increases operational efficiency, and maintains uniformity and integrity across your network devices. Once you have created a template, you create an application for that template (or templates, if you have several network devices using the same template). You use an application to set values for a template or templates. The process of assigning applications to templates enables you to customize your network devices to suit your needs.

Golden Configuration Workflow

The Golden Configuration workflow follows these basic phases:

- **Device Configuration**: This phase presents the device in the basic configuration.
- **Integrate with GC template**: In this phase you integrate the device with the configured GC template.
- **Create a GC application**: In this phase you create an application with customized variables and values used in the GC template.
- **Apply the configured device**: In this phase you need to load and merge both the template and the application into NSO.
- **Verify functionality in the network**: The GC application plan displays the status of the template application to the device.

Golden Configuration Prerequisites

For the Golden Configuration (GC) installation to function properly, these prerequisites need to be present and functioning.

- Jinja2 version 3.1.2
- MarkupSafe version 2.1.2

In addition to these prerequisites, Network Service Orchestrator (NSO) must fit these criteria:

- NSO **version 6.1.9** (recommended) must be up and running.
- Make sure that the devices to be configured must be onboarded to NSO.
- The **goldenconfig package** must be loaded on NSO.
- The required Network Element Drivers (NED) are loaded on NSO.

Note: To use CWM, you must also have additional functionality. See Crosswork Workflow Manager (CWM) Workflows and Deployment.

Device Templates and Applications

The power of Golden Configuration application is using templates and applications to meet your network requirements.

Templates

Templates are rendered using the Jinja2 template engine features to customize the configuration templates with conditionals and advanced features, such as looping, to meet system requirements. The templates provide the flexibility needed when working with a wide variety of network devices.

When you apply a template to a device, you can set the template to **Merge** (default) or **Replace** the existing configuration. The Merge setting merges the new configuration along with the existing configuration on the device. The Replace setting replaces the entire existing configuration.

Note: Templates can be applied on only a single device, but not on device groups.

Note: Variables must not be named using the reserved keywords **service** and/or **device**. These keywords are auto populated with the current service/device node data.

Applications

You configure applications with **values** that correspond to the variables used in devices. You set the variables in the template either while configuring the variables during template creation (template level) or while creating the application that you associate with the template (application level).

Note: The Golden Configuration application consists of templates with variables and applications with values for the templates.

Note: The application level variable configurations override the template level variable configurations.

Loading Templates and Applications to a Device

After you create a template, you:

SUMMARY STEPS

1. Load the template to the device with configured variables to the device.
2. Apply the values to the template to a device.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Load the template to the device with configured variables to the device.	

	Command or Action	Purpose
Step 2	Apply the values to the template to a device.	Shown here are two template (for IOS XR and Native) payloads with their respective application payloads.

Sample Template: (IOS XR device)

After you create a template, you apply that template to a device. This is a sample template applied to a Cisco IOS XR device along with the application payload.

Template Payload

```
    <devices xmlns="http://tail-f.com/ns/ncs">
    <template>
    <name>xr-bgp-template</name>
    <ned-id>
    <id
xmlns:cisco-iosxr-cli-7.52="http://tail-f.com/ns/ned-id/cisco-iosxr-cli-7.52">cisco-iosxr-cli-7.52:cisco-iosxr-cli-7.52</id>

    <config>
    <route-policy xmlns="http://tail-f.com/ned/cisco-ios-xr">
    <name>PASS_ALL</name>
    <value>pass</value>
    </route-policy>
    <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
    <bgp>
    <bgp-instance>
    <id>300</id>
    <instance>1</instance>
    <bgp>
    <router-id>4.4.4.4</router-id>
    </bgp>
    <address-family>
    <ipv4>
    <unicast/>
    </ipv4>
    </address-family>
    <neighbor>
    <id>{$NEIGHBOR_IP}</id>
    <remote-as>600</remote-as>
    </neighbor>
    </bgp-instance>
    </bgp>
    </router>
    </config>
    </ned-id>
    </template>
    </devices>
```

Application Payload

```
    <golden-config xmlns="http://example.com/golden-config">
    <application>
    <name>dt-bgp-app</name>
    <device>ncs540</device>
    <device-template>xr-bgp-template</device-template>
    <variable>
    <name>NEIGHBOR_IP</name>
    <value>10.10.1.5</value>
    </variable>
```

```
</application>
</golden-config>
```

Sample Template: Native Style

After you create a template, you:

SUMMARY STEPS

1. Load the template to the device with configured variables to the device.
2. Apply the values to the template to a device.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Load the template to the device with configured variables to the device.	
Step 2	Apply the values to the template to a device.	

Example

This is an example of the **native** template and application.

Template Payload

```
<config xmlns="http://tail-f.com/ns/config/1.0">
<golden-config xmlns="http://example.com/golden-config">
<template>
<name>xr-native-int</name>
<version>
<id>1</id>
<type>native</type>
<config>
interface TenGigE 0/0/0/1
ipv4 address {{ ADDR }} {{ MASK }}
</config>
</version>
</template>
</golden-config>
</config>
```

Application Payload

This is an example of the Golden Config application payload using the **native** template.

```
<config xmlns="http://tail-f.com/ns/config/1.0">
<golden-config xmlns="http://example.com/golden-config">
```

```
<application>
<name>native-xr-app</name>
<device>ncs540</device>
<jinja-template>
<template>xr-native-int</template>
<version>1</version>
</jinja-template>
<variable>
<name>ADDR</name>
<value><ip_address></value>
</variable>
<variable>
<name>MASK</name>
<value><ip_address></value>
</variable>
</application>
</golden-config>
</config>
```

Golden Configuration Template Tagging Feature

The Golden Configuration (GC) application allows you to **tag** templates with **specific actions**. These actions (see below) allow you to organize your templates. GC uses the following three actions to manage template maintenance and task performance.

Each action is described in a separate section:

- get-template
- get-application
- update-application

get-template: Get Template List and Version

Use this action to receive a list of templates and their associated versions. The action has an input and output shown here.

Input

```
leaf-list tag
```

Output

```
List of templates
Associated version
```

Use this sample command script to get a template.

```
admin@ncs% request golden-config actions get-template tag [ bgp c-style ] template {
name bgp-cstyle
version [ 1 2 ]
}
```

get-application: Get List of Applications, Devices, and Versions

Use this action to receive a list of applications, their associated devices, and the versions of each application.

Input

```
leaf jinja-template
leaf version
leaf device template
```

Output

```
List of applications
Associated device
Associated version
```

Sample command script to get an application.

```
admin@ncs% request golden-config actions get-application jinja-template
bgp-cstyle version 1
application {
name xr-bgp
device xr0
version 1
}
```

update-application: View Update Differences

Use this action to view the differences that occur due to an update and to obtain the list of updated applications.

Input

```
leaf jinja-template
leaf version
leaf device-template
leaf application
dry-run/outformat
```

Output

```
list of applications
dry-run diff
```

Sample command script to get an application.

```
    admin@ncs% request golden-config actions update-application application
    xr-bgp jinja-template bgp-cstyle version 1 dry-run
    result The following application(s) can be re-deployed when
    dry-run input
    is not opted for this action.
    application [ xr-bgp ]
    cli {
    local-node {
    data devices {
    device xr0 {
    config {
    router {
    bgp {
    bgp-no-instance 200 {
    vrf testXR {
    neighbor 10.10.1.2 {
    - remote-as 20;
```

```
+ remote-as 40;
}
}
}
}
}
}
}
}
}
}
}
```

Result

```
admin@ncs% request golden-config actions update-application application
xr-bgp jinja-template bgp-cstyle version 1
result The following application(s) are re-deployed.
application [ xr-bgp ]
```

Crosswork Workflow Manager (CWM) Workflows and Deployment

Golden Configuration has three CWM Workflows defined to create an application, delete an application, and update an application (remediation).

Prerequisites

To use CWM for Golden Configuration (GC) workflows, you need to have this functionality:

- NED packages, **goldenconfig** package, and required devices loaded in NSO.
- An NSO **secret** (password) used for REST call authentication with NSO.
- Create and deploy the **NSO adapter** in CWM.
- Create the NSO as a **resource** in CWM.
- Golden-config templates must be loaded in the NSO.

When the prerequisites are applied, you can then add the following workflows to CWM and run the required input to use Golden Configuration by CWM.

- create-application.sw.json
- delete-application.sw.json
- remediation-sw.json

Example: Use Golden Configuration to Install and Upgrade a Network Device

This example provides the information needed to install and upgrade a network device using the Golden-config template workflow in Crosswork Workflow Manager (CMW).

Sample Workflow

This is a sample template workflow.

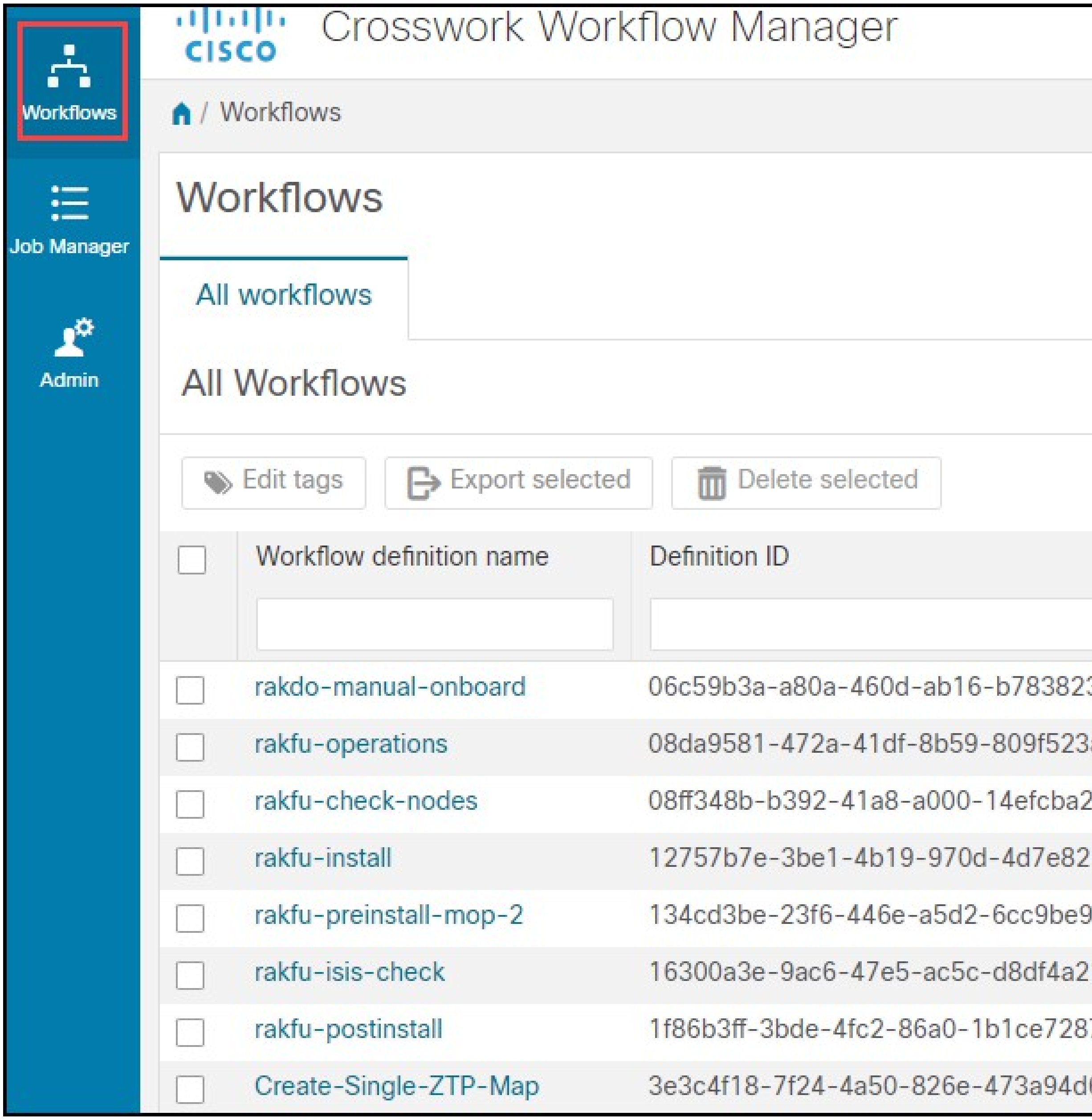
```
   admin@ncs% load merge terminal
   <config xmlns="http://tail-f.com/ns/config/1.0">
   <golden-config xmlns="http://example.com/golden-config">
   <template>
   <name>t2</name>
   <version>
   <id>2</id>
   <mode>merge</mode>
   <type>c-style</type>
   <config>ipv4 access-list acl-1
   40 deny tcp any any gt 200
   !
   10 permit icmp any any
   </config>
   <variable>
   <name>mask</name>
   <value><ip_address></value>
   </variable>
   </version>
   </template>
   </golden-config>
   </config>
   Type control + d to leave the terminal session
   [ok][2099-03-22 23:00:02]
   [edit]
   admin@ncs% commit Commit complete.
   [ok][2099-03-22 23:00:04]
   [edit]
   admin@ncs% show golden-config template t2
   version 2 {
   type c-style;
   mode merge;
   config "ipv4 access-list acl-1\n40 deny tcp any any gt 200\n!\n10 permit icmp any any\n
";
   variable mask {
   value <ip_address>;
   }
   }
   [ok][2024-03-22 23:00:17]
   [edit]
```

Workflow Procedure

Once you have the workflow template, complete steps 1-12 to apply application values to the template in CWM.

SUMMARY STEPS

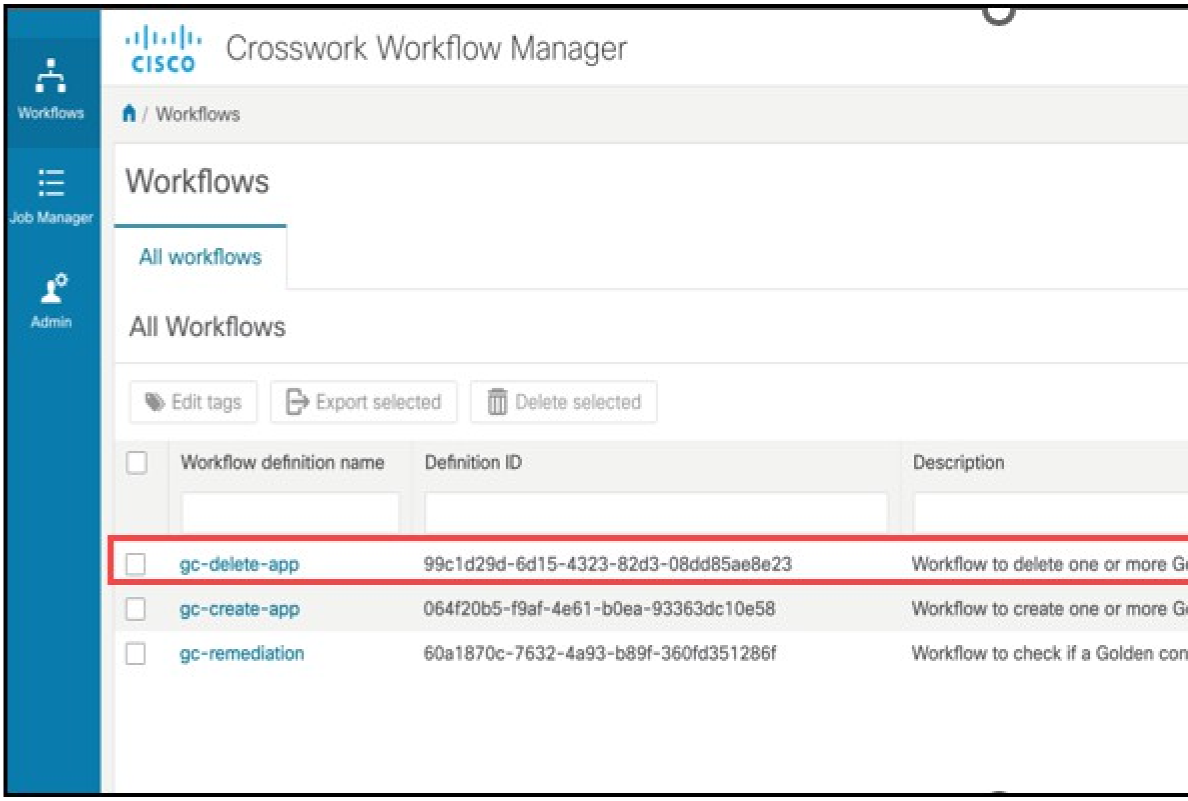
1. Log into **CWM** and choose the **Workflows** tab.



□	Workflow definition name	Definition ID
□	rakdo-manual-onboard	06c59b3a-a80a-460d-ab16-b78382
□	rakfu-operations	08da9581-472a-41df-8b59-809f523
□	rakfu-check-nodes	08ff348b-b392-41a8-a000-14efcba2
□	rakfu-install	12757b7e-3be1-4b19-970d-4d7e82
□	rakfu-preinstall-mop-2	134cd3be-23f6-446e-a5d2-6cc9be9
□	rakfu-isis-check	16300a3e-9ac6-47e5-ac5c-d8df4a2
□	rakfu-postinstall	1f86b3ff-3bde-4fc2-86a0-1b1ce728
□	Create-Single-ZTP-Map	3e3c4f18-7f24-4a50-826e-473a94d

2. Click **Create New Workflow**.

3. Click **Create Workflow**. The Workflow is listed in the Workflow Table. **Note:** In this example, the workflow is **gc-create-app.**

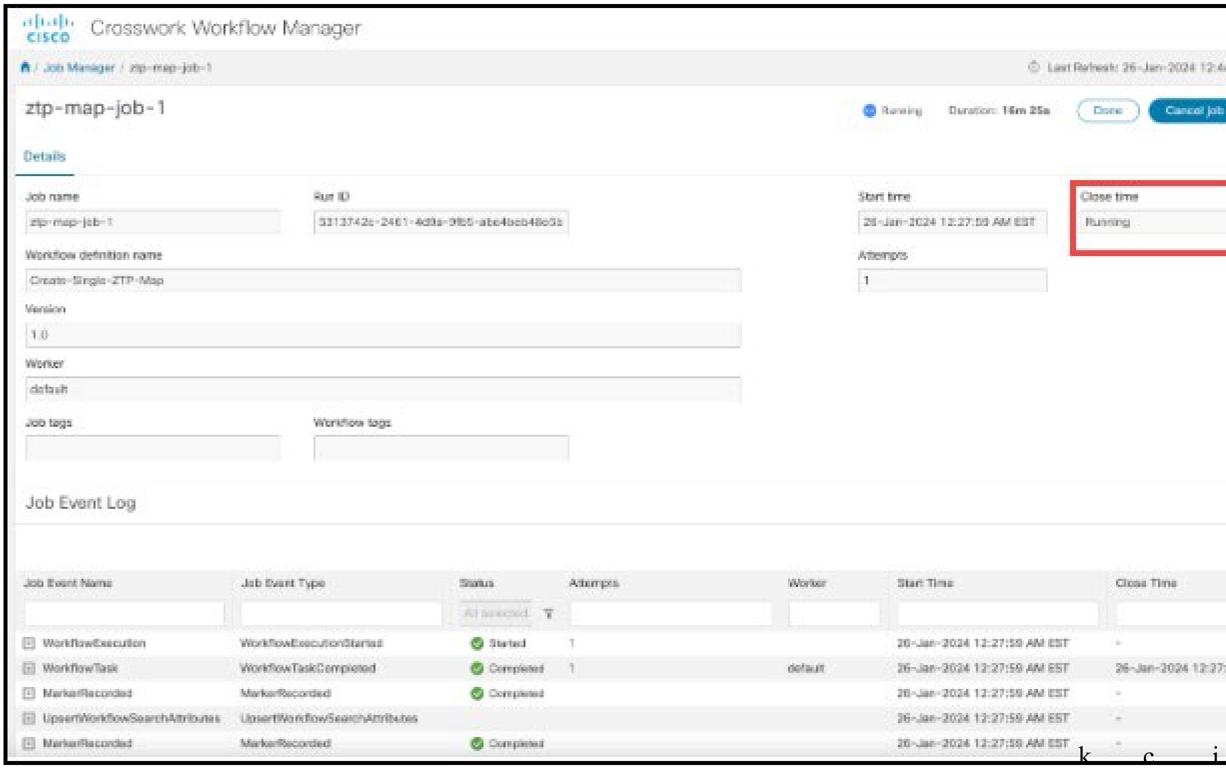


Workflows

All workflows

All Workflows

□	Workflow definition name	Definition ID	Description
□	gc-delete-app	99c1d29d-6d15-4323-82d3-08dd85ae8e23	Workflow to delete one or more G
□	gc-create-app	064f20b5-f9af-4e61-b0ea-93363dc10e58	Workflow to create one or more G
□	gc-remediation	60a1870c-7632-4a93-b89f-360fd351286f	Workflow to check if a Golden con



Crosswork Workflow Manager

Job Manager / ztp-map-job-1 Last Refresh: 26-Jan-2024 12:4

ztp-map-job-1 Running Duration: 16m 25s

Details

Job name: ztp-map-job-1

Run ID: 3313742c-2461-4d9a-9f65-abe4bcb48c3

Start time: 26-Jan-2024 12:27:59 AM EST

Close time: Running

Workflow definition name: Create-Single-ZTP-Map

Attempts: 1

Version: 1.0

Worker: default

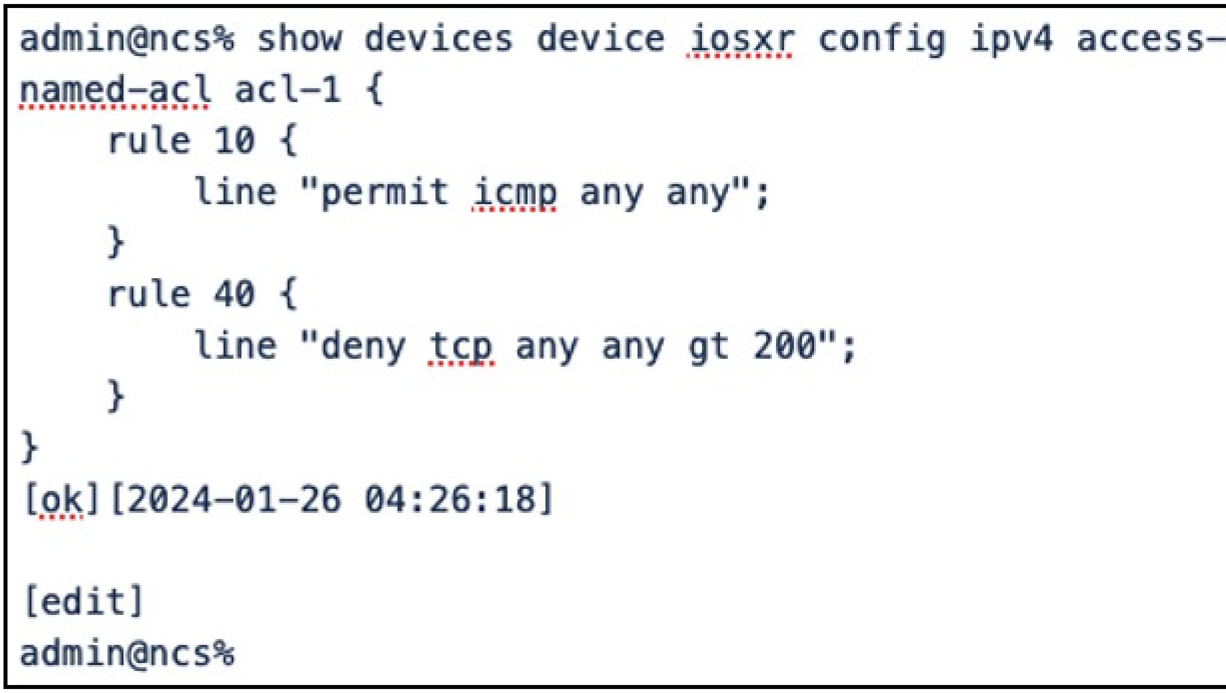
Job tags:

Workflow tags:

Job Event Log

Job Event Name	Job Event Type	Status	Attempts	Worker	Start Time	Close Time
WorkflowExecution	WorkflowExecutionStarted	✓ Started	1		26-Jan-2024 12:27:59 AM EST	-
WorkflowTask	WorkflowTaskCompleted	✓ Completed	1	default	26-Jan-2024 12:27:59 AM EST	26-Jan-2024 12:27:
MarkerRecorded	MarkerRecorded	✓ Completed			26-Jan-2024 12:27:59 AM EST	-
UpsertWorkflowSearchAttributes	UpsertWorkflowSearchAttributes				26-Jan-2024 12:27:59 AM EST	-
MarkerRecorded	MarkerRecorded	✓ Completed			26-Jan-2024 12:27:59 AM EST	-

4. k c i the **Workflow Name** to open the Workflow screen. (**Details** tab is default.) The Workflow Definition ID and Update Date are auto filled.
5. (Optional) Type any **Tags**.
6. Click the **Code** tab the **script** for the map.
7. Click **Run** the Run job window opens.

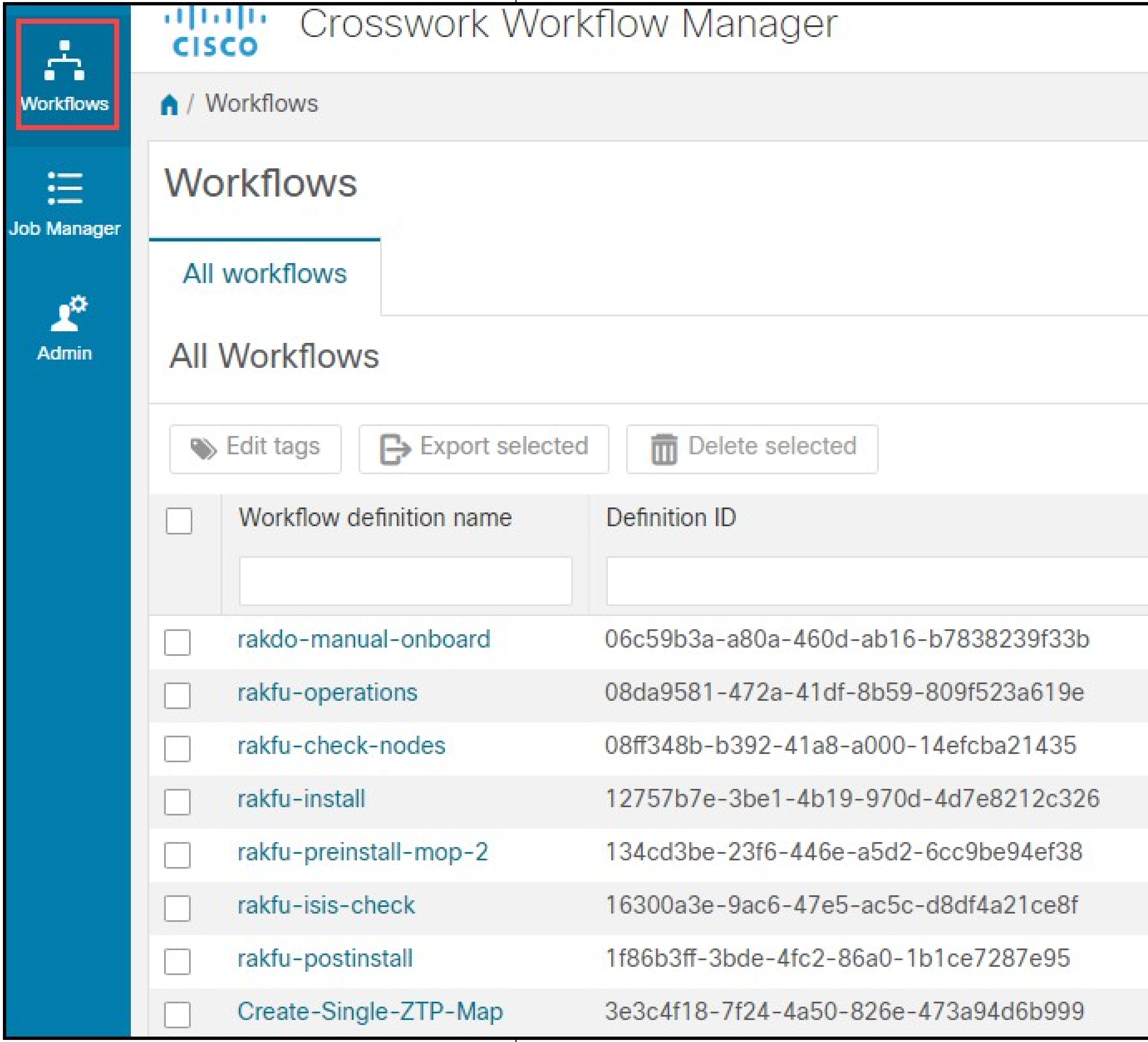


```
admin@ncs% show devices device iosxr config ipv4 access-
named-acl acl-1 {
    rule 10 {
        line "permit icmp any any";
    }
    rule 40 {
        line "deny tcp any any gt 200";
    }
}
[ok][2024-01-26 04:26:18]

[edit]
admin@ncs%
```

8. (Required) Type in a **Job Name**. **Note:** You can type in any job name to any unique string.
9. (Optional) Type in any **Tags**.
10. (Required) In the **Input variables** field, provide the **json input** file.
11. Click **Run Job** to start the workflow. **Note:** If you want to schedule the workflow, configure the **fields** in the When section (see step 12. Skip to **Running the Map** if you want to run the workflow immediately.
12. (Optional) in the **When** section configure the time, frequency, and order that the map runs.

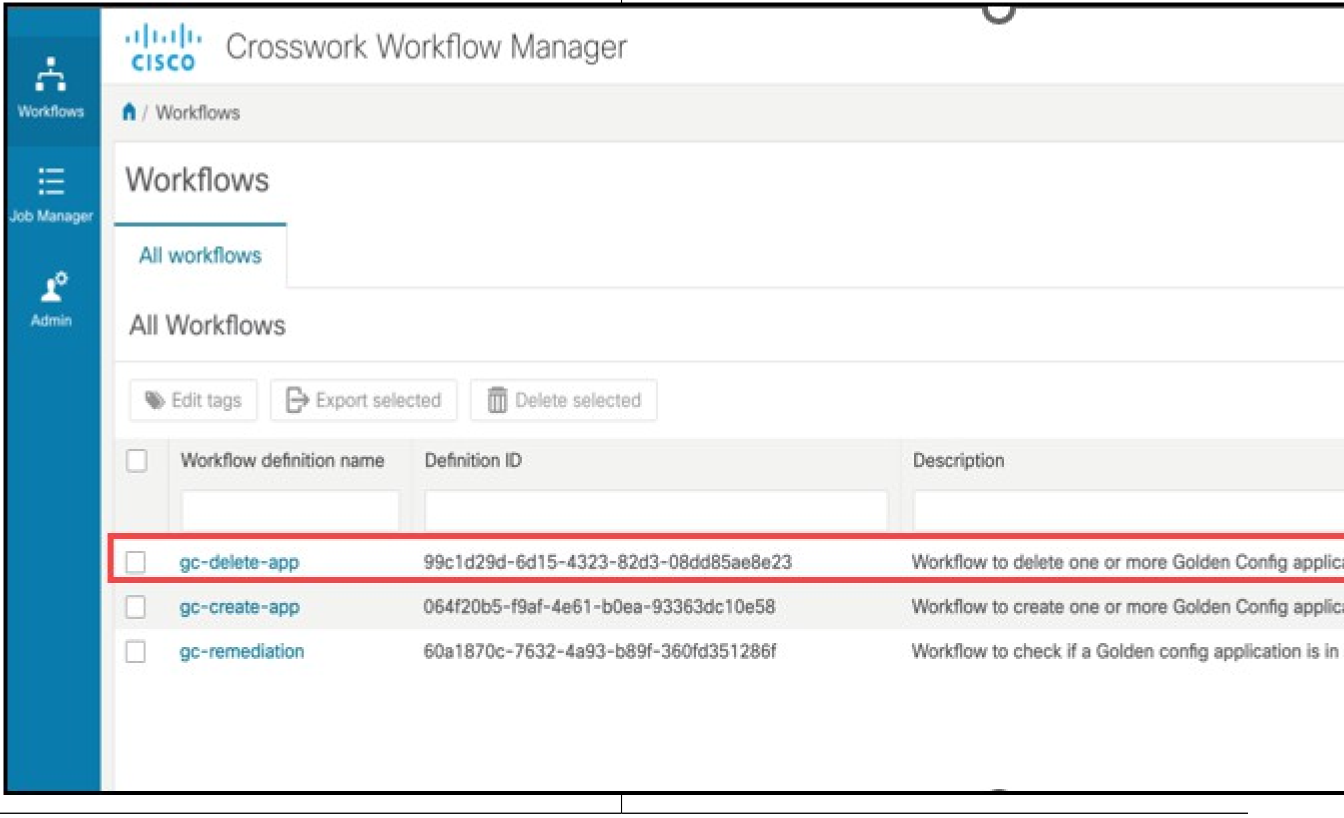
DETAILED STEPS

Procedure

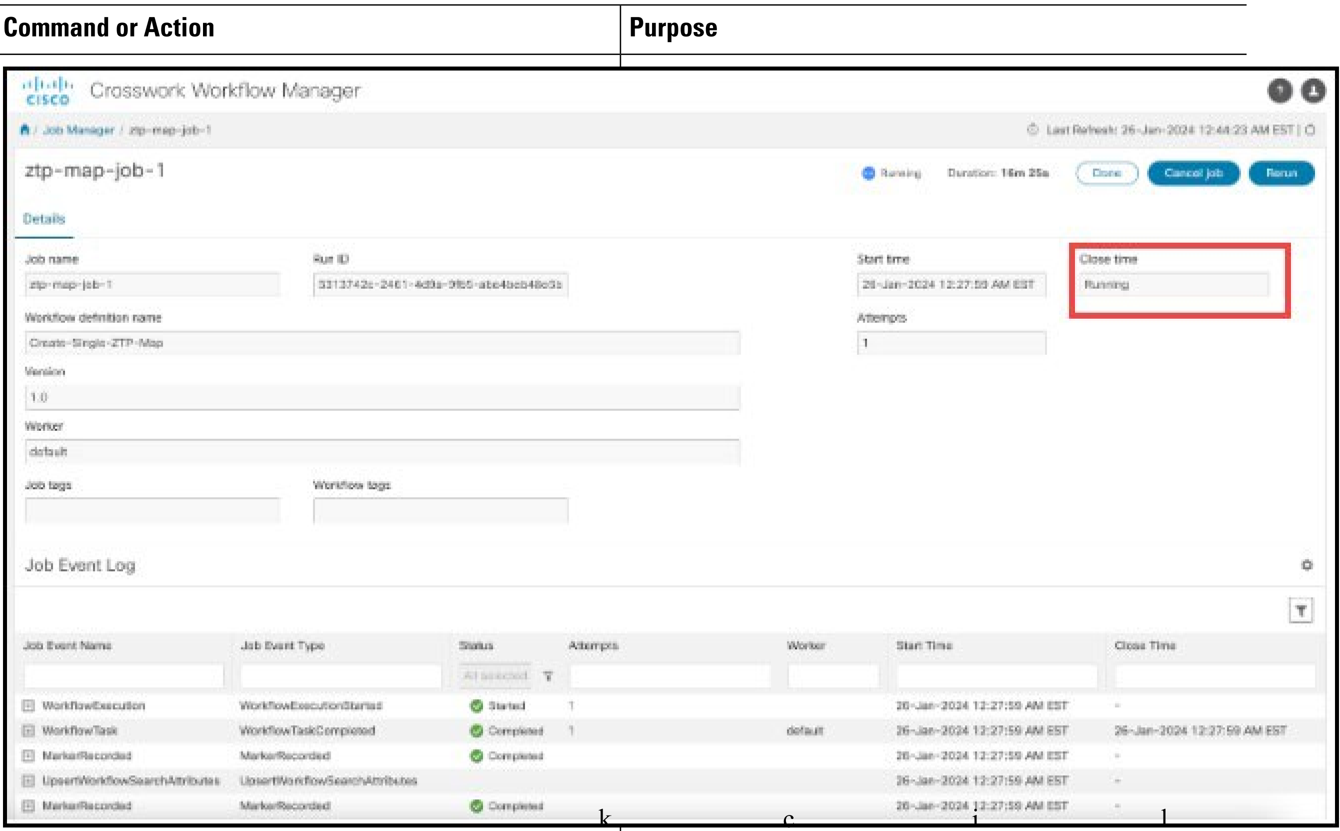
<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td><b>Step 1</b></td><td>Log into <b>CWM</b> and choose the <b>Workflows</b> tab.


<table>
<tr><th>☐</th><th>Workflow definition name</th><th>Definition ID</th></tr>
<tr><td>☐</td><td>rakdo-manual-onboard</td><td>06c59b3a-a80a-460d-ab16-b7838239f33b</td></tr>
<tr><td>☐</td><td>rakfu-operations</td><td>08da9581-472a-41df-8b59-809f523a619e</td></tr>
<tr><td>☐</td><td>rakfu-check-nodes</td><td>08ff348b-b392-41a8-a000-14efcba21435</td></tr>
<tr><td>☐</td><td>rakfu-install</td><td>12757b7e-3be1-4b19-970d-4d7e8212c326</td></tr>
<tr><td>☐</td><td>rakfu-preinstall-mop-2</td><td>134cd3be-23f6-446e-a5d2-6cc9be94ef38</td></tr>
<tr><td>☐</td><td>rakfu-isis-check</td><td>16300a3e-9ac6-47e5-ac5c-d8df4a21ce8f</td></tr>
<tr><td>☐</td><td>rakfu-postinstall</td><td>1f86b3ff-3bde-4fc2-86a0-1b1ce7287e95</td></tr>
<tr><td>☐</td><td>Create-Single-ZTP-Map</td><td>3e3c4f18-7f24-4a50-826e-473a94d6b999</td></tr>
</table></td><td></td></tr>
<tr><td><b>Step 2</b></td><td>Click <b>Create New Workflow</b>.</td><td></td></tr>
<tr><td><b>Step 3</b></td><td>Click <b>Create Workflow</b>. The Workflow is listed in the Workflow Table. <b>Note:</b> In this example, the workflow is</td><td></td></tr>
</table>

	Command or Action	Purpose
	gc-create-app.	



□	Workflow definition name	Definition ID	Description
□	gc-delete-app	99c1d29d-6d15-4323-82d3-08dd85ae8e23	Workflow to delete one or more Golden Config applica
□	gc-create-app	064f20b5-f9af-4e61-b0ea-93363dc10e58	Workflow to create one or more Golden Config applica
□	gc-remediation	60a1870c-7632-4a93-b89f-360fd351286f	Workflow to check if a Golden config application is in

	Command or Action	Purpose
Step 4	 the **Workflow Name** to open the Workflow screen. (**Details** tab is default.) The Workflow Definition ID and Update Date are auto filled.	
Step 5	(Optional) Type any **Tags**.	
Step 6	Click the **Code** tab the **script** for the map.	

Crosswork Workflow Manager
Job Manager / ztp-map-job-1
Last Refresh: 26-Jan-2024 12:44:23 AM EST
Done
Cancel job
Rerun
Details

ztp-map-job-1

Running Duration: 16m 25s

Job name: ztp-map-job-1

Run ID: 3313742c-2461-4d9a-9f65-abe4bcb48c5b

Start time: 26-Jan-2024 12:27:59 AM EST

Close time: Running

Workflow definition name: Create-Single-ZTP-Map

Attempts: 1

Version: 1.0

Worker: default

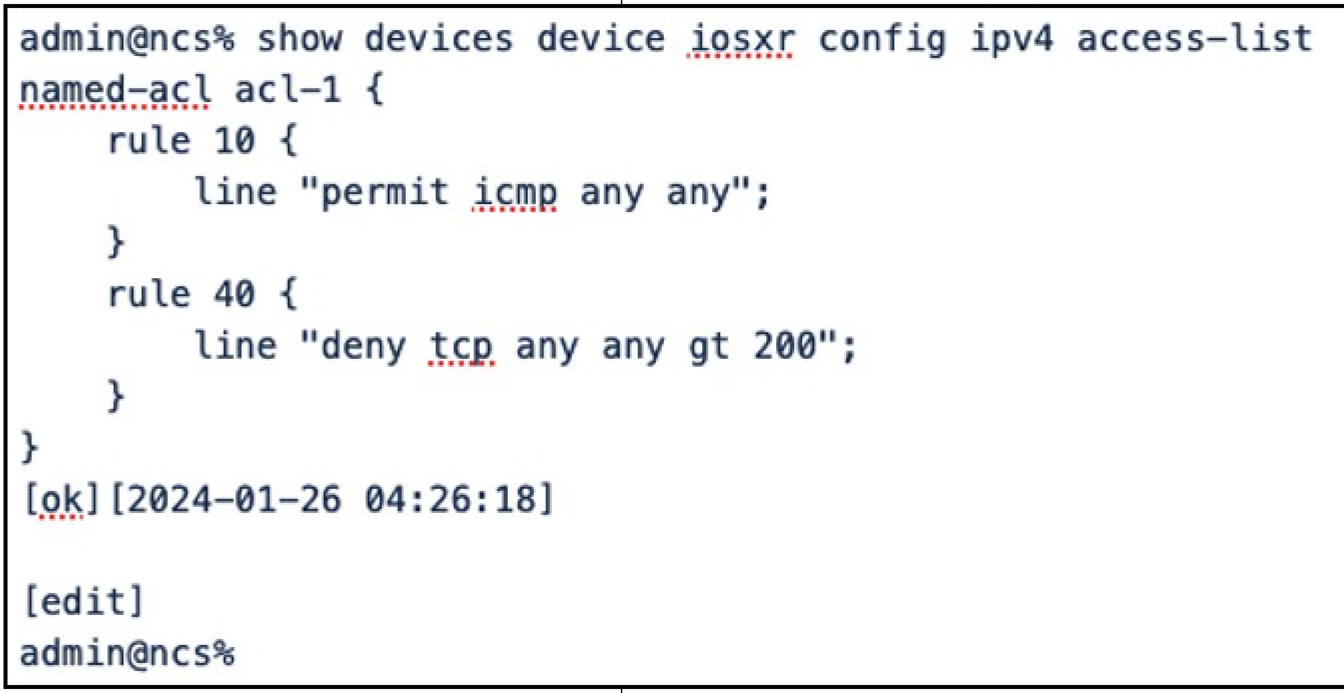
Job tags:

Workflow tags:

Job Event Log

Job Event Name	Job Event Type	Status	Attempts	Worker	Start Time	Close Time
		All selected				
WorkflowExecution	WorkflowExecutionStarted	✓ Started	1		26-Jan-2024 12:27:59 AM EST	-
WorkflowTask	WorkflowTaskCompleted	✓ Completed	1	default	26-Jan-2024 12:27:59 AM EST	26-Jan-2024 12:27:59 AM EST
MarkerRecorded	MarkerRecorded	✓ Completed			26-Jan-2024 12:27:59 AM EST	-
UpsertWorkflowSearchAttributes	UpsertWorkflowSearchAttributes				26-Jan-2024 12:27:59 AM EST	-
MarkerRecorded	MarkerRecorded	✓ Completed			26-Jan-2024 12:27:59 AM EST	-

	Command or Action	Purpose
Step 7	Click **Run** the Run job window opens.	



```
admin@ncs% show devices device iosxr config ipv4 access-list
named-acl acl-1 {
    rule 10 {
        line "permit icmp any any";
    }
    rule 40 {
        line "deny tcp any any gt 200";
    }
}
[ok][2024-01-26 04:26:18]

[edit]
admin@ncs%
```

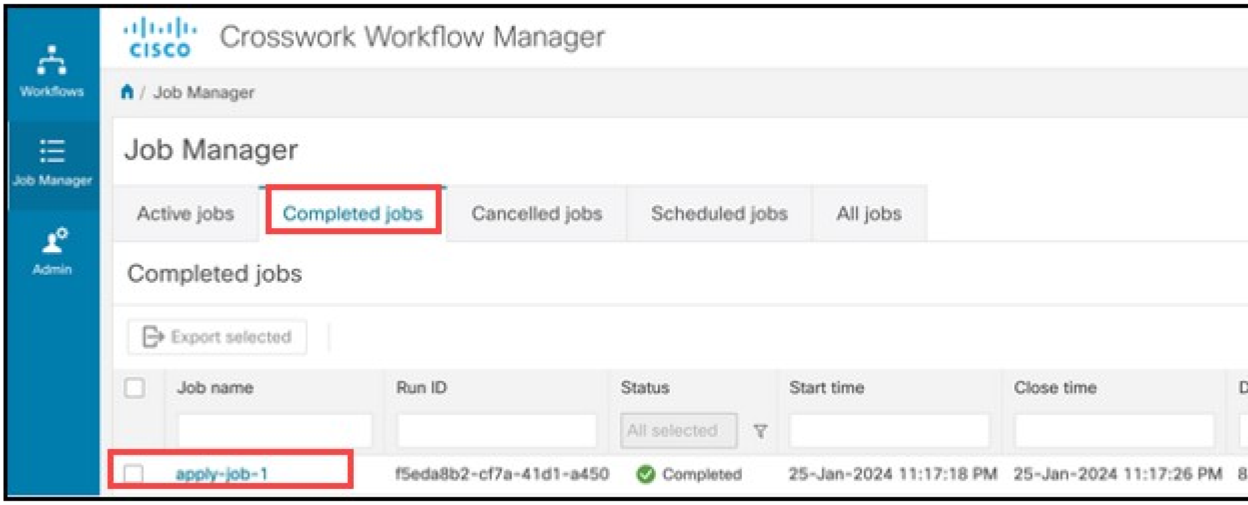
Step 8	(Required) Type in a **Job Name**. **Note:** You can type in any job name to any unique string.	
Step 9	(Optional) Type in any **Tags**.	
Step 10	(Required) In the **Input variables** field, provide the **json input** file.	
Step 11	Click **Run Job** to start the workflow. **Note:** If you want to schedule the workflow, configure the **fields** in the When section (see step 12. Skip to **Running the Map** if you want to run the workflow immediately.	
Step 12	(Optional) in the **When** section configure the time, frequency, and order that the map runs.	

Running the Map

After you click **Run Job**.

SUMMARY STEPS

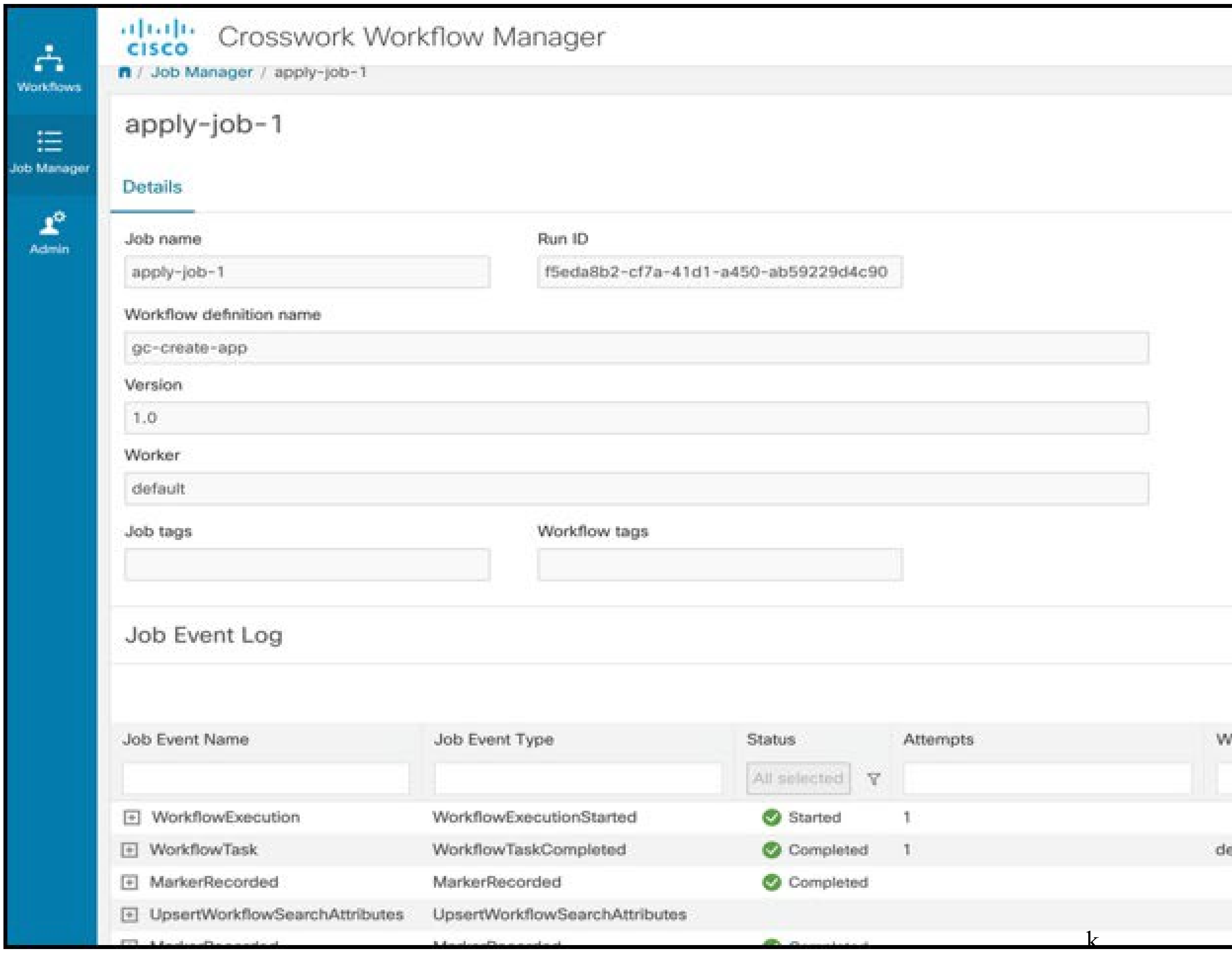
1. Select **Job Manager** > **Completed Jobs**



Job Manager

Completed jobs

	Job name	Run ID	Status	Start time	Close time
□			All selected		
□	apply-job-1	f5eda8b2-cf7a-41d1-a450	✓ Completed	25-Jan-2024 11:17:18 PM	25-Jan-2024 11:17:26 PM



apply-job-1

Job name: apply-job-1

Run ID: f5eda8b2-cf7a-41d1-a450-ab59229d4c90

Workflow definition name: gc-create-app

Version: 1.0

Worker: default

Job tags:

Workflow tags:

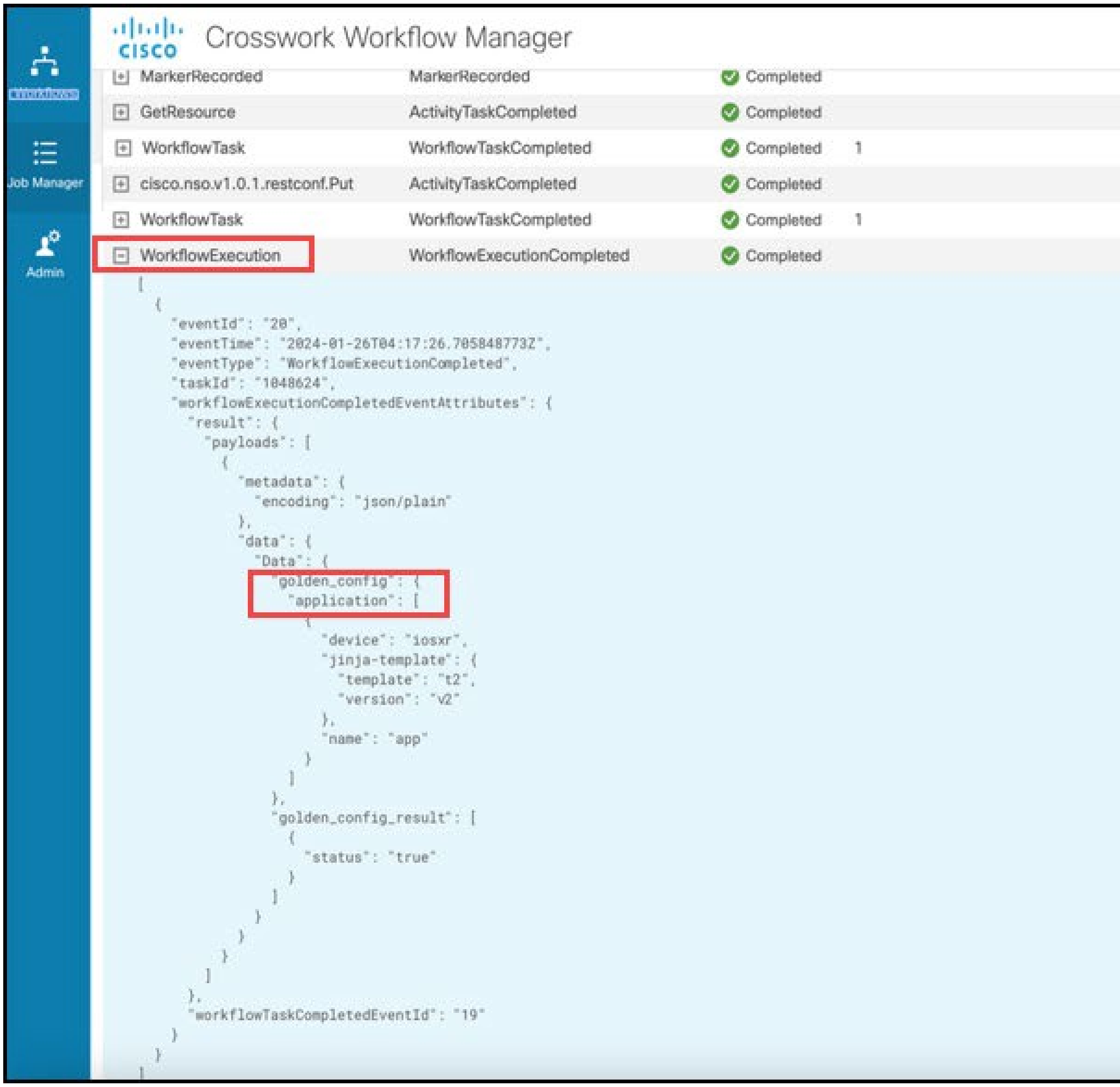
Job Event Log

Job Event Name	Job Event Type	Status	Attempts
		All selected	
⊞ WorkflowExecution	WorkflowExecutionStarted	✓ Started	1
⊞ WorkflowTask	WorkflowTaskCompleted	✓ Completed	1
⊞ MarkerRecorded	MarkerRecorded	✓ Completed	
⊞ UpsertWorkflowSearchAttributes	UpsertWorkflowSearchAttributes		

2. k

the **job name** you want to open. (Apply-job-1 In this example. The job status shows the date and time that the job was closed.)

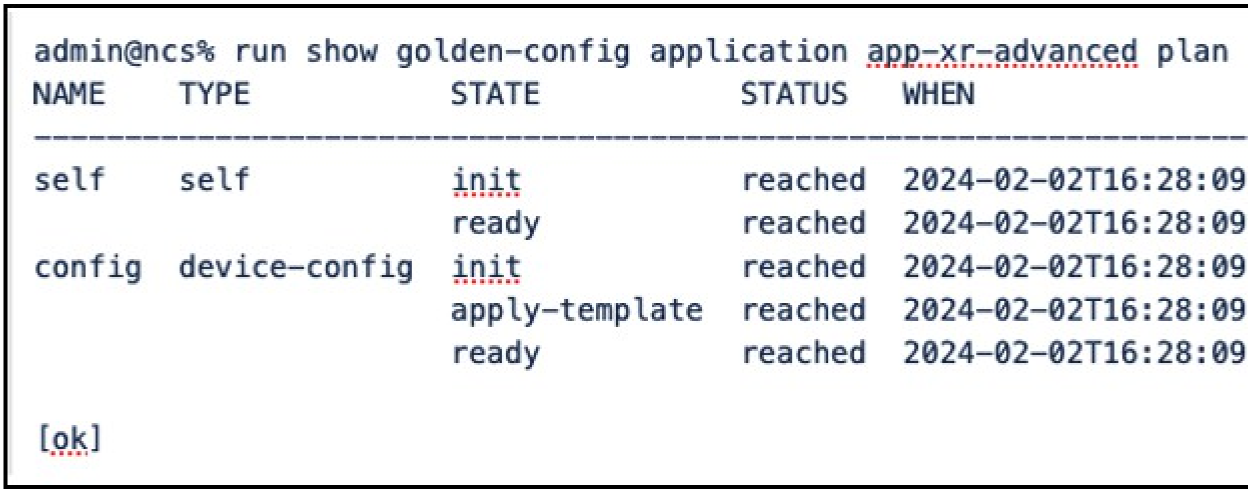
3. Once the workflow is finished. Choose **Job Manager** > **Completed Jobs** tab. The **job** is listed in the table.
4. Click the **Job Name**. The Job page opens showing the job details and **Job Event Log.**
5. In the **Job Event Log** section, click the plus (+) sign to the left of the **WorkflowExecution** (last event in the list).



⊞ MarkerRecorded	MarkerRecorded	✓ Completed	
⊞ GetResource	ActivityTaskCompleted	✓ Completed	
⊞ WorkflowTask	WorkflowTaskCompleted	✓ Completed	1
⊞ cisco.nso.v1.0.1.restconf.Put	ActivityTaskCompleted	✓ Completed	
⊞ WorkflowTask	WorkflowTaskCompleted	✓ Completed	1
⊟ WorkflowExecution	WorkflowExecutionCompleted	✓ Completed	

```
[
  {
    "eventId": "20",
    "eventTime": "2024-01-26T04:17:26.705848773Z",
    "eventType": "WorkflowExecutionCompleted",
    "taskId": "1048624",
    "workflowExecutionCompletedEventAttributes": {
      "result": {
        "payloads": [
          {
            "metadata": {
              "encoding": "json/plain"
            },
            "data": {
              "Data": {
                "golden_config": {
                  "application": [
                    {
                      "device": "iosxr",
                      "jinja-template": {
                        "template": "t2",
                        "version": "v2"
                      },
                      "name": "app"
                    }
                  ]
                },
                "golden_config_result": [
                  {
                    "status": "true"
                  }
                ]
              }
            }
          }
        ]
      },
      "workflowTaskCompletedEventId": "19"
    }
  }
]
```

6. After the workflow has run, check the golden-config application plan in **NSO**.

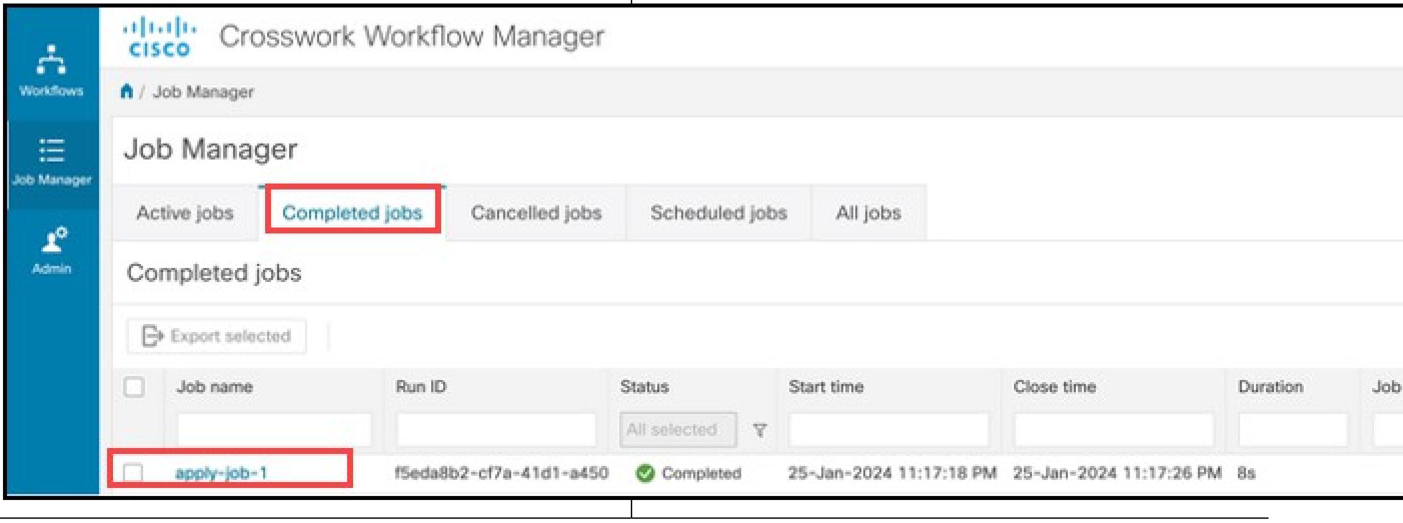


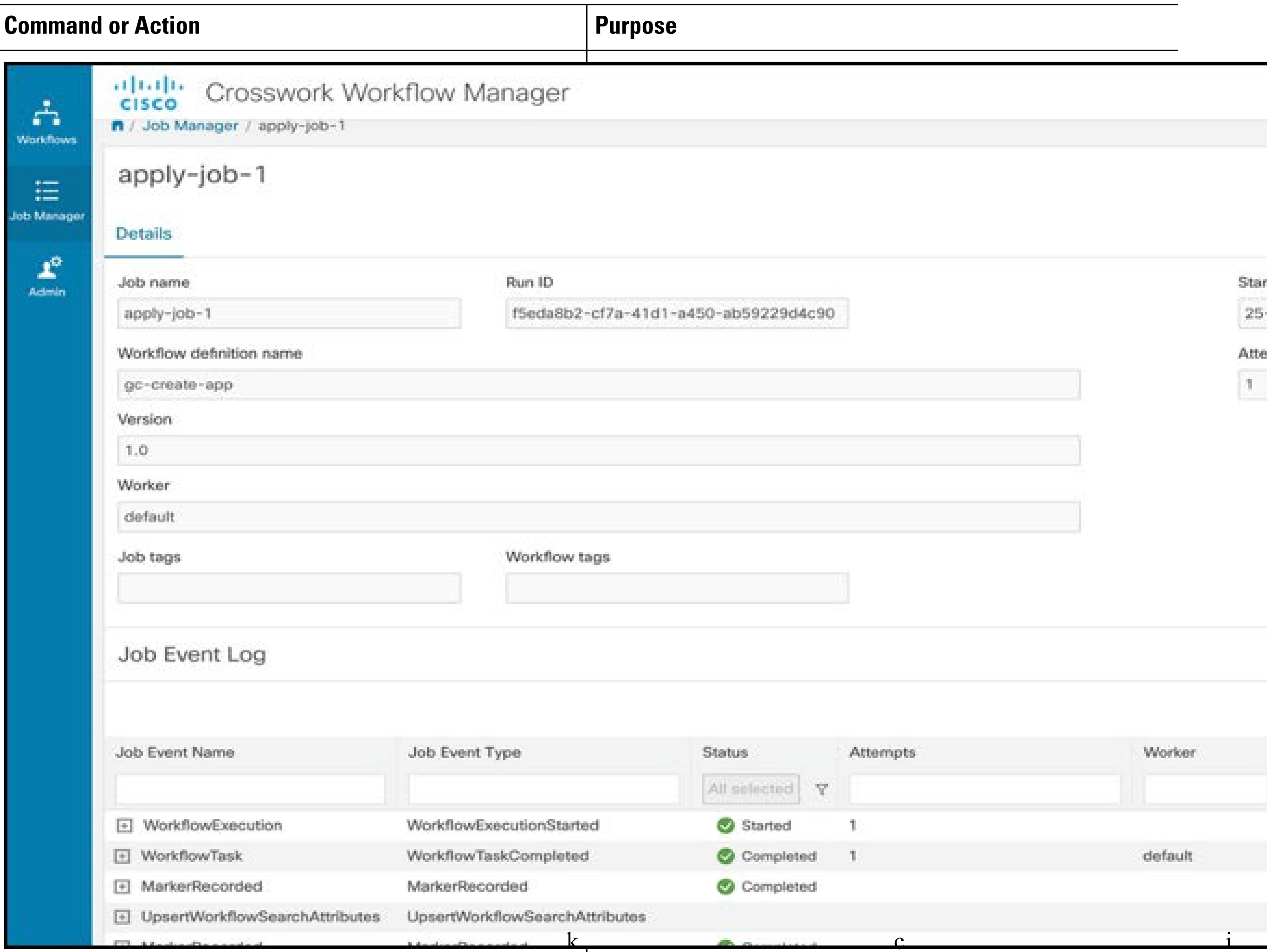
```
admin@ncs% run show golden-config application app-xr-advanced plan
NAME    TYPE           STATE          STATUS   WHEN
---------------------------------------------------------------------
self    self           init           reached  2024-02-02T16:28:09
                       ready          reached  2024-02-02T16:28:09
config  device-config  init           reached  2024-02-02T16:28:09
                       apply-template reached  2024-02-02T16:28:09
                       ready          reached  2024-02-02T16:28:09

[ok]
```

7. In **NSO**, you can verify that the **template configuration** has been pushed on the device.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Select **Job Manager** > **Completed Jobs**  Crosswork Workflow Manager Workflows Job Manager Admin / Job Manager Job Manager Active jobs Completed jobs Cancelled jobs Scheduled jobs All jobs Completed jobs Export selected All selected Job name | Run ID | Status | Start time | Close time | Duration | Job □ apply-job-1 | f5eda8b2-cf7a-41d1-a450 | ✓ Completed | 25-Jan-2024 11:17:18 PM | 25-Jan-2024 11:17:26 PM | 8s	

	Command or Action	Purpose
Step 2	 k c i the **job name** you want to open. (Apply-job-1 In this example. The job status shows the date and time that the job was closed.)	
Step 3	Once the workflow is finished. Choose **Job Manager** > **Completed Jobs** tab. The **job** is listed in the table.	
Step 4	Click the **Job Name**. The Job page opens showing the job details and **Job Event Log.**	

Crosswork Workflow Manager

Job Manager / apply-job-1

apply-job-1

Details

Job name: apply-job-1

Run ID: f5eda8b2-cf7a-41d1-a450-ab59229d4c90

Star 25-

Workflow definition name: gc-create-app

Atte 1

Version: 1.0

Worker: default

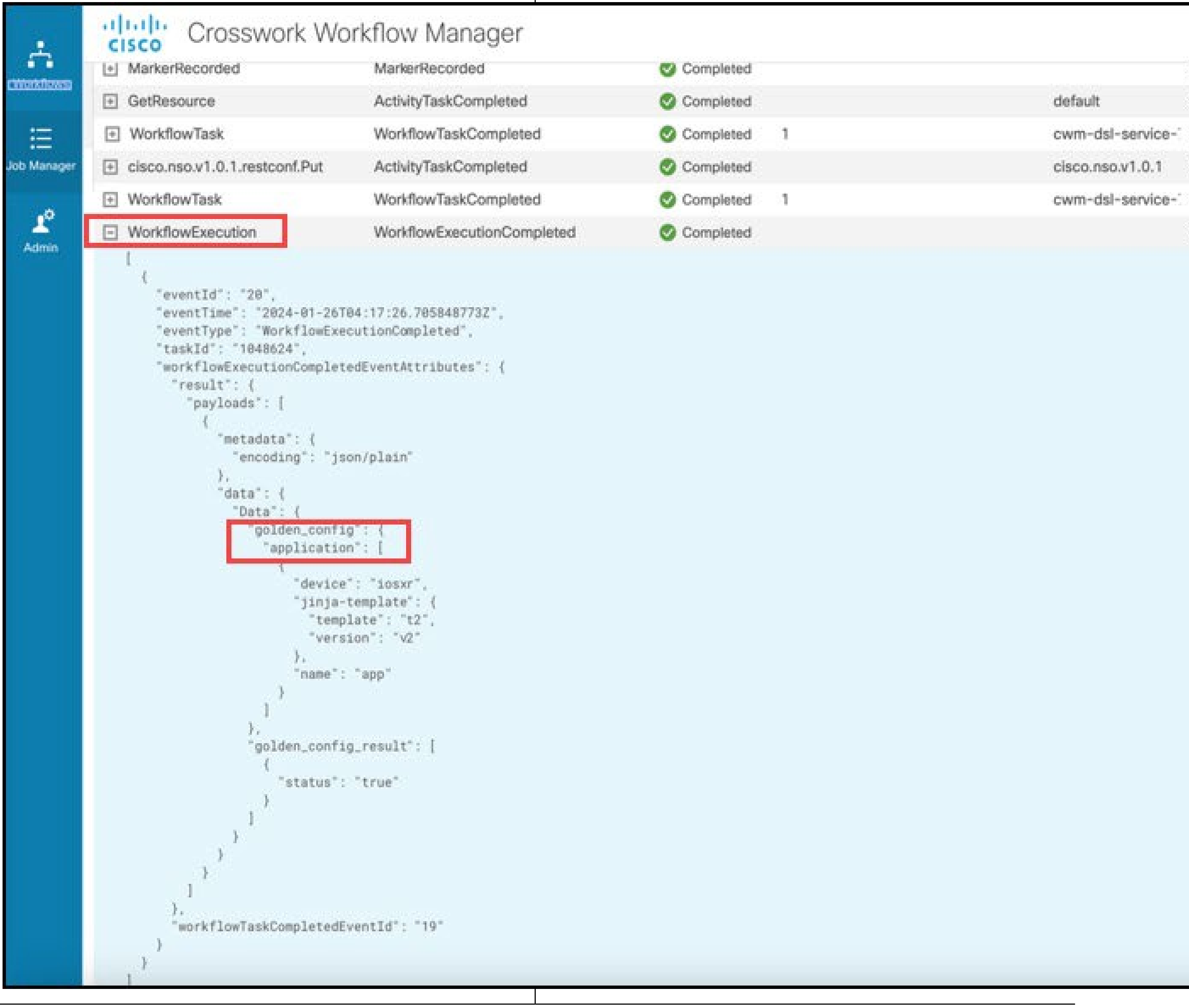
Job tags:

Workflow tags:

Job Event Log

Job Event Name	Job Event Type	Status	Attempts	Worker
		All selected		
⊞ WorkflowExecution	WorkflowExecutionStarted	✓ Started	1	
⊞ WorkflowTask	WorkflowTaskCompleted	✓ Completed	1	default
⊞ MarkerRecorded	MarkerRecorded	✓ Completed		
⊞ UpsertWorkflowSearchAttributes	UpsertWorkflowSearchAttributes			

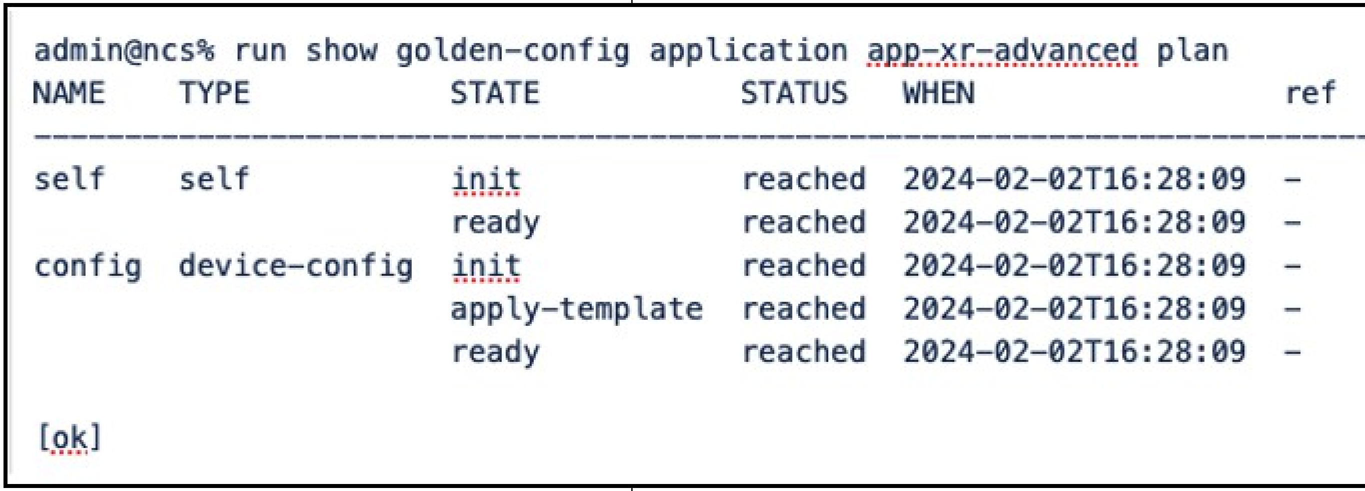
	Command or Action	Purpose
Step 5	In the **Job Event Log** section, click the plus (+) sign to the left of the **WorkflowExecution** (last event in the list).	



⊞ MarkerRecorded	MarkerRecorded	✓ Completed		
⊞ GetResource	ActivityTaskCompleted	✓ Completed		default
⊞ WorkflowTask	WorkflowTaskCompleted	✓ Completed	1	cwm-dsl-service-
⊞ cisco.nso.v1.0.1.restconf.Put	ActivityTaskCompleted	✓ Completed		cisco.nso.v1.0.1
⊞ WorkflowTask	WorkflowTaskCompleted	✓ Completed	1	cwm-dsl-service-
⊟ WorkflowExecution	WorkflowExecutionCompleted	✓ Completed		

```
[
  {
    "eventId": "20",
    "eventTime": "2024-01-26T04:17:26.705848773Z",
    "eventType": "WorkflowExecutionCompleted",
    "taskId": "1048624",
    "workflowExecutionCompletedEventAttributes": {
      "result": {
        "payloads": [
          {
            "metadata": {
              "encoding": "json/plain"
            },
            "data": {
              "Data": {
                "golden_config": {
                  "application": [
                    {
                      "device": "iosxr",
                      "jinja-template": {
                        "template": "t2",
                        "version": "v2"
                      },
                      "name": "app"
                    }
                  ]
                },
                "golden_config_result": [
                  {
                    "status": "true"
                  }
                ]
              }
            }
          }
        ]
      },
      "workflowTaskCompletedEventId": "19"
    }
  }
]
```

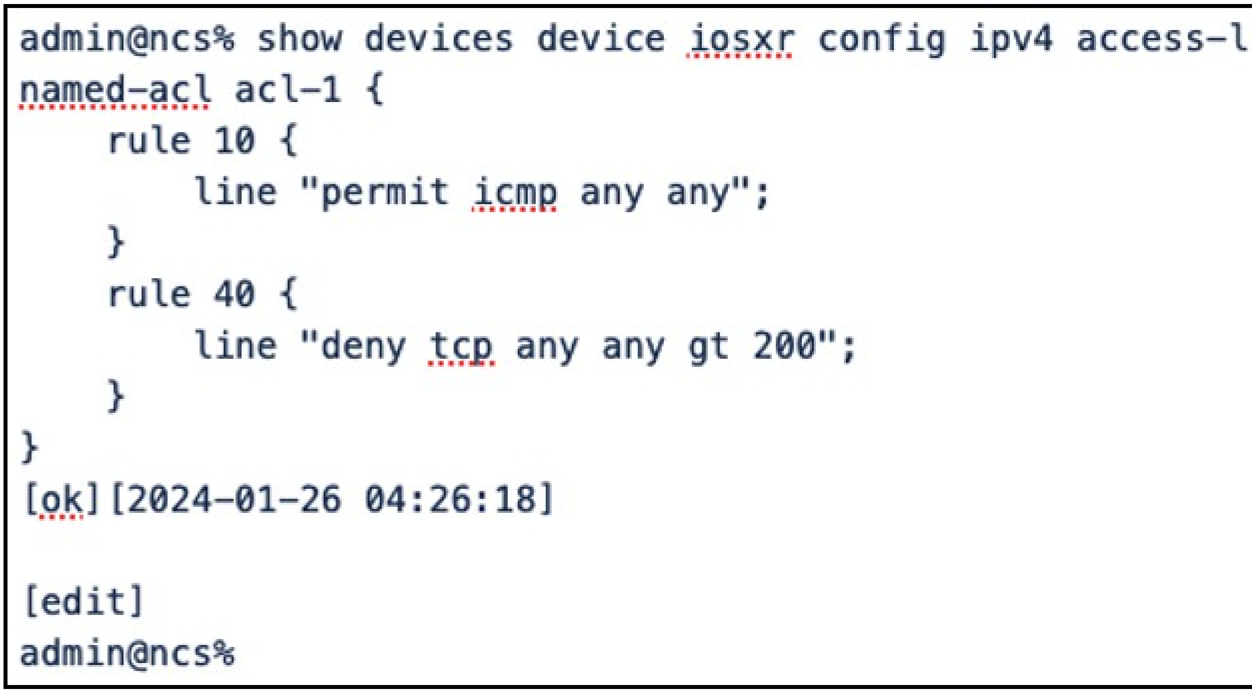
	Command or Action	Purpose
Step 6	After the workflow has run, check the golden-config application plan in **NSO**.	



```
admin@ncs% run show golden-config application app-xr-advanced plan
NAME    TYPE           STATE           STATUS   WHEN                 ref
--------------------------------------------------------------------------
self    self           init            reached  2024-02-02T16:28:09  -
                       ready           reached  2024-02-02T16:28:09  -
config  device-config  init            reached  2024-02-02T16:28:09  -
                       apply-template  reached  2024-02-02T16:28:09  -
                       ready           reached  2024-02-02T16:28:09  -

[ok]
```

Step 7	In **NSO**, you can verify that the **template configuration** has been pushed on the device.	

Example



```
admin@ncs% show devices device iosxr config ipv4 access-l
named-acl acl-1 {
    rule 10 {
        line "permit icmp any any";
    }
    rule 40 {
        line "deny tcp any any gt 200";
    }
}
[ok][2024-01-26 04:26:18]

[edit]
admin@ncs%
```