



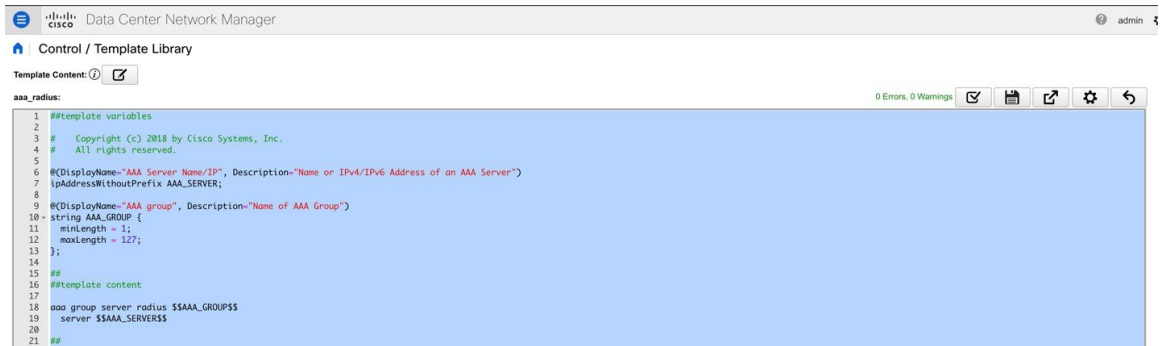
# Template Usage in Cisco DCNM LAN Fabric Deployment

templateType	Specifies the type of Template used.	<ul style="list-style-type: none"><li>• CLI</li><li>• POLICY</li><li>• SHOW</li><li>• PROFILE</li><li>• ABSTRACT</li></ul>
--------------	--------------------------------------	--

- [Policy Template, on page 1](#)
- [Fabric Template, on page 5](#)
- [Profile Template, on page 5](#)
- [Viewing, Editing, and Adding Policies, on page 6](#)
- [Deploying New Configurations, on page 10](#)
- [switch\\_freeform Template Usage, on page 11](#)
- [Changing the Contents of a Template in Use, on page 14](#)

## Policy Template

For the policy template, there are two template content types: CLI and PYTHON. With CLI content type, the policy templates are parameterized CLI templates. They can have a lot of variables and CLIs. Typically, CLI policy templates are small and do not have any if-else-for etc. like constructs. An example CLI policy template for AAA server configuration is shown below:








The screenshot shows the Cisco Data Center Network Manager interface. The top navigation bar includes the Cisco logo, the text 'Data Center Network Manager', and a user profile 'admin'. Below this, the breadcrumb 'Control / Template Library' is visible. The main content area is titled 'Template Content: [edit icon]' and shows a code editor for a template named 'aaa\_radius'. The code is as follows:

```
1 ##template variables
2
3 # Copyright (c) 2018 by Cisco Systems, Inc.
4 # All rights reserved.
5
6 @(DisplayName="AAA Server Name/IP", Description="Name or IPv4/IPv6 Address of an AAA Server")
7 ipAddressWithoutPrefix AAA_SERVER;
8
9 @(DisplayName="AAA group", Description="Name of AAA Group")
10 string AAA_GROUP {
11     minLength = 1;
12     maxLength = 127;
13 };
14
15 ##
16 ##template content
17
18 aaa group server radius $$AAA_GROUP$$
19 server $$AAA_SERVER$$
20
21 ##
```

But you can also have policy templates of template content type PYTHON. Essentially, this allows multiple CLI policy templates to be combined together with a common “source” so that they get all applied/un-applied at one go. For example, when you want to create a vPC host port, it has to be created symmetrically on both peers that are part of the vPC pair. In addition, you have to create port-channel, member interfaces, channel-group, etc. This is why a python vPC host policy template has been added. An example interface PYTHON template for setting up a routed interface is shown below:

Control / Template Library

Template Content: 

int\_routed\_host\_11\_1 0 Errors, 0 Warnings     

```

1  ##template variables
2
3  # Copyright (c) 2018 by Cisco Systems, Inc.
4  # All rights reserved.
5
6  @(IsInternal=true)
7  string SERIAL_NUMBER;
8
9  @(PrimaryAssociation=true, IsInternal=true)
10 interface INTF_NAME;
11
12 @(IsMandatory=false, DisplayName="Interface VRF", Description="Interface VRF name, default VRF if not specified")
13 string INTF_VRF {
14     minLength = 1;
15     maxLength = 32;
16 };
17
18 @(IsMandatory=false, DisplayName="Interface IP", Description="IP address of the interface")
19 ipv4Address IP;
20
21 @(IsMandatory="IP!=null", DisplayName="IP Netmask Length", Description="IP netmask length used with the IP address (Min:1, Max:31)")
22 integer PREFIX {
23     min = 1;
24     max = 31;
25 };
26
27 @(IsMandatory=false, DisplayName="Routing TAG", Description="Routing tag associated with interface IP")
28 string ROUTING_TAG;
29
30 @(DisplayName="MTU", IsMTU=true, Description="MTU for the interface (Min:576, Max:9216)")
31 integer MTU {
32     min = 576;
33     max = 9216;
34     defaultValue=9216;
35 };
36
37 @(DisplayName="SPEED", Description="Interface Speed")
38 enum SPEED {
39     validValues=Auto,100Mb,1Gb,10Gb,25Gb,40Gb,100Gb;
40     defaultValue=Auto;
41 };
42
43 @(IsMandatory=false, DisplayName="Interface Description", Description="Add description to the interface (Max Size 254)")
44 string DESC {
45     minLength = 1;
46     maxLength = 254;
47 };
48
49 @(IsMandatory=false, IsMultiLineString=true, DisplayName="Freeform Config", Description="Additional CLI for the interface")
50 string CONF;
51
52 @(DisplayName="Enable Interface", Description="Uncheck to disable the interface")
53 boolean ADMIN_STATE {
54     defaultValue=true;
55 };
56
57 ##
58 ##template content
59
60 from com.cisco.dcbu.vinc1.rest.services.jython import PTIWrapper
61 from com.cisco.dcbu.vinc1.rest.services.jython import Wrapper
62 from com.cisco.dcbu.vinc1.rest.services.jython import WrappersResp
63 from utility import *
64
65 def add():
66     try:
67         if CONF != "":
68             respObj, conf = Util.adjustIntfFreeformConfig(SERIAL_NUMBER, INTF_NAME, CONF)
69             if respObj.isRetCodeFailure():
70                 return respObj
71
72     # modify to be done, calling delete now to clean up PTIs before add
73     delete()
74
75     intfVrf = "default"
76     try:
77         if INTF_VRF != "":
78             intfVrf = INTF_VRF
79     except:
80         Wrapper.print("Switch/Intf = [%s/%s] - Template[int_routed_host_11_1]: INTF_VRF not defined" %
81             (SERIAL_NUMBER, INTF_NAME))
82         pass
83
84     routingTag = ""
85     try:
86         if ROUTING_TAG != "":
87             routingTag = ROUTING_TAG
88     except:
89         Wrapper.print("Switch/Intf = [%s/%s] - Template[int_routed_host_11_1]: ROUTING_TAG not defined" %
90             (SERIAL_NUMBER, INTF_NAME))
91         pass
92
93     # routed_interface has only one CLI command: no switchport
94     # It must be configured before interface_vrf
95     # p2p_routed_interface that configures the IP address must come after interface_vrf
96     Util.exe(PTIWrapper.createOrUpdate(SERIAL_NUMBER, "INTERFACE",
97         INTF_NAME, INTF_NAME,
98         ConfigPriority.CONFIG_PRIO_INTF,
99         "routed_interface",
100         {"INTF_NAME": INTF_NAME}))
101
102     if intfVrf != "default":
103         # Create/Update PTI for interface VRF
104         Util.exe(PTIWrapper.createOrUpdate(SERIAL_NUMBER, "INTERFACE",
105             INTF_NAME, INTF_NAME,
106             ConfigPriority.CONFIG_PRIO_INTF_SUB_LVL1,
107             "interface_vrf",
108             {"INTF_NAME": INTF_NAME, "INTF_VRF": intfVrf}))
109
110     if IP != "":
111         if routingTag == "":
112             Util.exe(PTIWrapper.createOrUpdate(SERIAL_NUMBER, "INTERFACE",
113                 INTF_NAME, INTF_NAME,
114                 ConfigPriority.CONFIG_PRIO_INTF_SUB_LVL2,
115                 "p2p_routed_interface",
116                 {"INTF_NAME": INTF_NAME, "IP": IP, "PREFIX": PREFIX}))

```

Each policy template has a template subtype like DEVICE, INTERFACE, etc. This allows the right policy template to appear at the right selection point. For example, in the Interface window, you will only see the interface policy templates.

The screenshot shows the 'Control / Template Library' window in Cisco Data Center Network Manager. The 'Templates' section is active, displaying a list of 17 templates. The 'Tags' column for all templates is 'interface'. The 'Template ...' column shows 'POLICY' and 'INTERFAC...'. The 'Published' column shows 'false' for all. The 'Modified T...' column shows dates from 2019-06-03.

Name	Supported Platforms	Tags	Template ...	Template ...	Published	Modified T...	D...
csr1kv_loopback	CSR1KV	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
epi_routed_intf	N9K	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
GigabitEthernet	CSR1KV	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
GigabitEthernet_freeform	CSR1KV	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_access_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_loopback_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_mgmt_11_1	N9K	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_monitor_ethernet_11_1	N9K	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_monitor_port_channel_11_1	N9K	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_port_channel_access_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_port_channel_trunk_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_routed_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_subif_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_trunk_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_vpc_access_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	
int_vpc_trunk_host_11_1	All	[interface_...]	POLICY	INTERFAC...	false	2019-06-03...	

In the View/Edit Policies window on the Fabric Builder, you will only see device policy templates.

The screenshot shows the 'Control / Template Library' window in Cisco Data Center Network Manager. The 'Templates' section is active, displaying a list of 20 templates. The 'Tags' column for all templates is 'device'. The 'Template ...' column shows 'POLICY' and 'DEVICE'. The 'Published' column shows 'false' for all. The 'Modified T...' column shows dates from 2019-06-03.

Name	Supported Platforms	Tags	Template ...	Template ...	Published	Modified T...	D...
aaa_radius	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_radius_deadtime	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_radius_key	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_radius_src_interface	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_radius_use_vrf	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_tacacs	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_tacacs_key	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_tacacs_src_interface	N9K		POLICY	DEVICE	false	2019-06-03...	
aaa_tacacs_use_vrf	N9K		POLICY	DEVICE	false	2019-06-03...	
anycast_gateway	N9K		POLICY	DEVICE	false	2019-06-03...	
anycast_rp	N9K		POLICY	DEVICE	false	2019-06-03...	
azure_network_selector	CSR1KV		POLICY	DEVICE	false	2019-06-03...	
banner	N9K		POLICY	DEVICE	false	2019-06-03...	
base_aaa	N9K		POLICY	DEVICE	false	2019-06-03...	
base_bgp	N9K		POLICY	DEVICE	false	2019-06-03...	
base_bgp_external	N9K,N7K		POLICY	DEVICE	false	2019-06-03...	
base_dhcp	N9K		POLICY	DEVICE	false	2019-06-03...	

You can make a copy of any of these templates and customize them as per their needs. That is the typical use-case for customization. **Do not** modify existing policies but make a copy, and then customize as per the requirements. Otherwise, after a DCNM upgrade, the changes may be lost.

In general, a template already in use, meaning one that is already applied to some switch within any fabric, cannot be edited.



**Note** No Type-CLI templates are used in the LAN fabric installation mode. They are all replaced with more powerful Policy templates which are a super set.

## Fabric Template

A fabric template is basically a python template, specifically jython, which is java + python. A fabric template is quite comprehensive, and in that it embeds the rules that are required for deploying a fabric, including all the logic required to generate intended configuration of all switches within the entire fabric. Configuration is generated based on published Cisco best practice guidelines. In addition to the embedded rules, the fabric template also integrates with other entities such as resource manager, topology database, device roles, configuration compliance, etc. and generates the configuration accordingly for all the devices in the fabric. This is the inherent part of the DCNM fabric builder.

The expectation is that users will not create their own fabric templates. DCNM provides a few fabric templates out of the box such as Easy Fabric, External Fabric, MSD Fabric, eBGP Fabric (introduced in DCNM 11.2).

Name	Supported Platforms	Tags	Template ...	Template ...	Published	Modified T...	D...
Easy_Fabric_11_1	All		FABRIC	NA	false	2019-06-03...	F...
Easy_Fabric_eBGP	All		FABRIC	NA	false	2019-06-03...	F...
External_Fabric_11_1	All		FABRIC	NA	false	2019-06-03...	F...
MSD_Fabric_11_1	All		FABRIC	NA	false	2019-06-03...	F...

## Profile Template

A profile template is used for provisioning of overlays (networks or VRFs). The idea is that when you apply some overlay configuration, there are multiple pieces of configurations that should go together. For example, valid layer-3 network configuration in a VXLAN EVPN fabric requires VLAN, SVI, int nve config, EVPN route-target, etc. All of these pieces are put together into what is called a configuration profile (NX-OS construct) and then effectively applied at one go. Either the whole configuration profile gets applied or nothing gets applied, on the switch. In this way, you are not left with any dangling or stray configurations on the switches. For any kind of overlay configurations, whether it is on the leaf or on the borders, DCNM employs profile templates.

There are four kinds of profile templates that are distinguished with tags as depicted below:

- Network Profile (applied to all devices with role leaf)
- Network Extension Profile (applied to all devices with role 'border\*')

- VRF Profile (applied to all devices with role leaf)
- VRF Extension Profile (applied to all devices with role ‘border\*’)

The screenshot shows the Cisco Data Center Network Manager (DCNM) interface. The breadcrumb navigation indicates the user is in the 'Control / Template Library' section. Below the navigation, there is a search bar and a 'Show Quick Filter' dropdown. The main content area displays a table of templates. The table has columns for Name, Supported Platforms, Tags, Template Type, Published, Modified T..., and D... (Details). The 'Default\_VRF\_Extension\_Universal' template is highlighted in blue.

Name	Supported Platforms	Tags	Template ...	Template ...	Published	Modified T...	D...
base_external_router	N9K		PROFILE	NA	false	2019-06-03...	s...
Default_Network_Extension_Universal	All	[networkEx...	PROFILE	VXLAN	false	2019-06-03...	D...
Default_Network_Universal	All	[network]	PROFILE	VXLAN	false	2019-06-03...	D...
Default_VRF_Extension_Universal	All	[vrfExtension]	PROFILE	VXLAN	false	2019-06-03...	D...
Default_VRF_Universal	All	[vrf]	PROFILE	VXLAN	false	2019-06-03...	D...
ext_base_setup	All	[borderBase]	PROFILE	VXLAN	false	2019-06-03...	
ext_fabric_intf	All		PROFILE	VXLAN	false	2019-06-03...	
ext_fabric_multisite_intf_11_1	All		PROFILE	VXLAN	false	2019-06-03...	
ext_multisite_overlay_setup_11_1	All	[multiSiteO...	PROFILE	VXLAN	false	2019-06-03...	
ext_multisite_rs_base_feature	N9K,N7K	[multiSiteO...	PROFILE	VXLAN	false	2019-06-03...	s...
ext_multisite_rs_base_setup	N9K	[multiSiteO...	PROFILE	VXLAN	false	2019-06-03...	s...

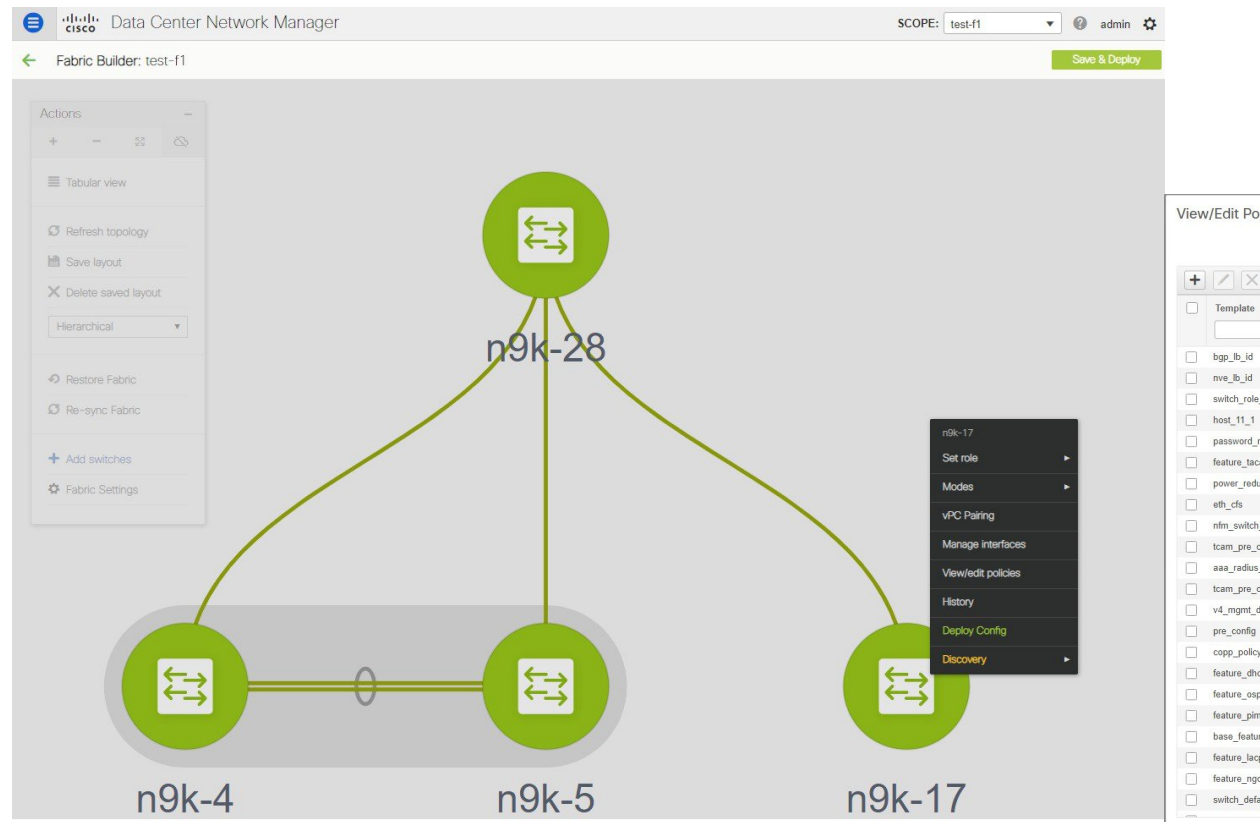
For more information about how to apply overlay configuration via the Networks & VRFs workflow in DCNM, see *Creating and Deploying Networks and VRFs* section.

### Additional Notes

When a policy or profile template is applied, an instance is created for each application of the template. The common terminology used for this is Policy Template Instance or PTI. A PTI is effectively a policy or profile template + the Name-value pairs that give it a specific instance, post substitution. PTIs created for a device can be viewed under the View/Edit policies option for that device in Fabric Builder. In the tabular view, the View/Edit policies button allows selection and bulk creation/deletion of policies across a subset of devices in the entire fabric. For more information, see *Viewing and Editing Policies* section.

## Viewing, Editing, and Adding Policies

To navigate to the View/Edit Policies window, right-click a device in the Fabric Builder window and select View/edit policies.



The View/Edit Policies window can be used to view, edit, or create a policy for a device. Note that Interface policies can only be viewed but cannot be edited/created from the View/Edit Policies window. Interfaces can only be edited, created, or deleted from the Interfaces window.

## Viewing Policies

To view certain policies for a device, you can use filters by specifying the search criteria in the empty boxes under each field. After the policies are found, you can view the content by selecting multiple policies and clicking on the “View” button. Below are examples that show how to use filters and how to view the configuration associated with a policy instance.

### Example: Viewing Policies for a Device

Enter `tcam` in the search field to filter the templates, select the template that you want to view, and click the View button to view TCAM policies created for the device.

Template	Policy ID	Fabric Name	Serial Number	Editable	Entity Type	Entity Name
tcam						
<input type="checkbox"/> tcam_pre_config_9300	POLICY-9300	test-f1	SAL18432P6M	true	SWITCH	SWITCH
<input type="checkbox"/> tcam_pre_config_vxlan	POLICY-9330	test-f1	SAL18432P6M	true	SWITCH	SWITCH

**Example: Viewing Policies for an Interface**

Enter the interface name in the search field under Entity Name to filter interfaces. Select an interface, and click the View button to view policies created for the interface.

Template	Policy ID	Fabric Name	Serial Number	Editable	Entity Type	Entity Name	Source	Priority	Content Type	Mark Deleted
						Ethernet1/29				
<input type="checkbox"/> trunk_interface	POLICY-9420	test-f1	SAL18432P6M	false	INTERFACE	Ethernet1/29	Ethernet1/29	350	TEMPLATE_CLI	false
<input type="checkbox"/> int_trunk_host_11_1	POLICY-9390	test-f1	SAL18432P6M	false	INTERFACE	Ethernet1/29	Ethernet1/29	350	PYTHON	false
<input type="checkbox"/> interface_mtu	POLICY-9450	test-f1	SAL18432P6M	false	INTERFACE	Ethernet1/29	Ethernet1/29	352	TEMPLATE_CLI	false
<input type="checkbox"/> porttype_fast_trunk	POLICY-9520	test-f1	SAL18432P6M	false	INTERFACE	Ethernet1/29	Ethernet1/29	352	TEMPLATE_CLI	false
<input type="checkbox"/> no_shut_interface	POLICY-9530	test-f1	SAL18432P6M	false	INTERFACE	Ethernet1/29	Ethernet1/29	352	TEMPLATE_CLI	false



**Note**

- Each interface should be associated with one interface jython policy template.
- An interface jython policy template does not have CLI in its content but rather creates PTIs of CLI policy templates. All these PTIs are combined to generate a complete configuration associated with an interface.



## Editing Policies

Not all device policies can be edited from the View/Edit policies window. Only the policies that are created with an empty Source and have the flag Editable = true, can be edited.

### Procedure

- Step 1** To edit a device policy, select an existing policy and click on the edit or ‘Pencil’ button. The ‘Edit Policy’ window opens.
- Step 2** After changing 1 or more Name-value pairs, press the ‘Save’ button to save the changes on the Edit Policy window.
- Step 3** To deploy the changed config, go back to the Fabric Builder window, right-click on the device and select ‘Deploy Config’.
- This will invoke Configuration Compliance to generate the pending config for the device. Pending config is the diff between the current config on the switch and the new intent config.
- Step 4** If the pending config is correct, click ‘Deploy Config’ to push the pending config onto the switch.

### Example: Editing a Policy

This example shows how to change the IPv4 management default gateway.

The screenshot displays the 'Edit Policy' dialog box for policy POLICY-9140. The dialog is titled 'Edit Policy' and shows the following details:

- Policy ID: POLICY-9140
- Entity Type: SWITCH
- Template Name: v4\_mgmt\_default\_gateway
- Entity Name: SWITCH
- Priority (1-1000): 910
- General tab is active, showing the 'Default Gateway' field set to 22.0.0.88. A note indicates: '\* Default Gateway IP address to use with mgmt0'.
- Variables: (Empty)

The background shows a list of policies with 'v4\_mgmt\_default\_gat...' selected. To the right, a 'Config Deployment' panel shows a table with columns 'Switch Name' and 'IP Address', containing the entry 'n9k-17' with IP '22.0.0.17'.

## Adding Policies

### Procedure

**Step 1** To add a policy to a device, click the '+' button on the View/Edit Policies page.  
The 'Add Policy' windows opens.

**Step 2** From the Policy drop-down list, select a policy to be added to the device.

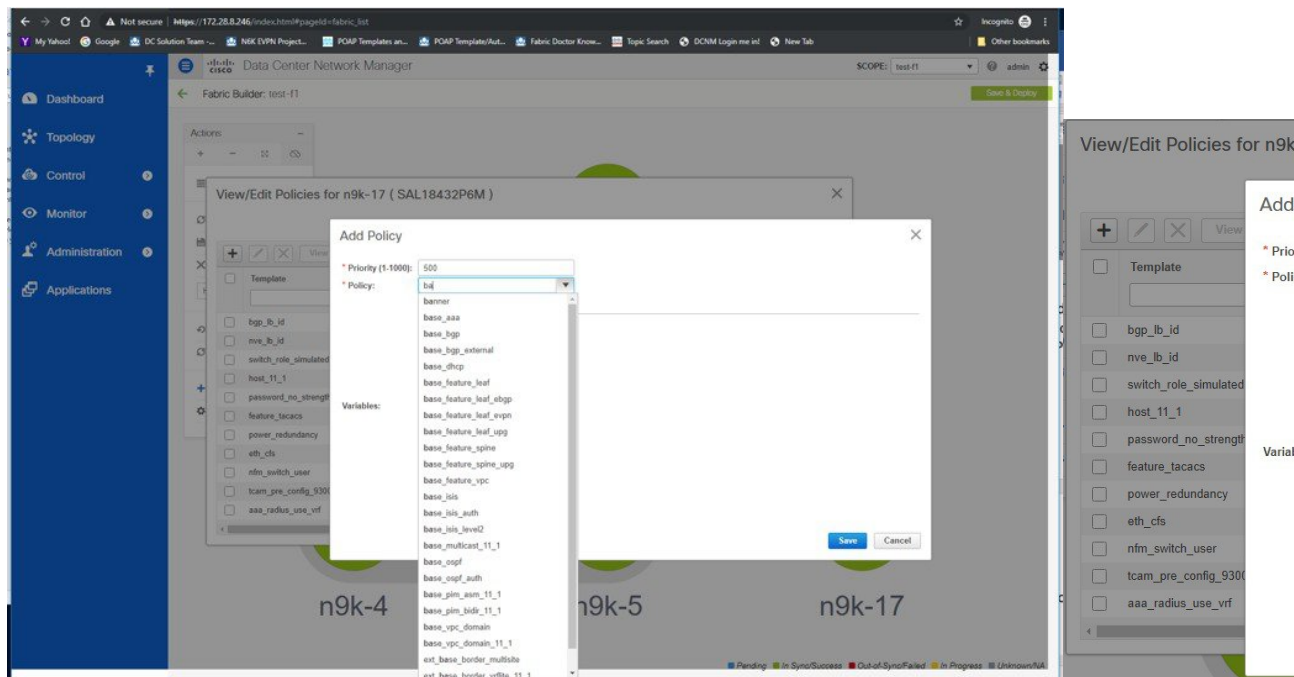
**Step 3** Set the policy priority and input the mandatory fields.

**Step 4** Click the 'Save' button to save and complete adding the policy.

**Note** Policy Priority is used to determine the order in which the configuration will be applied to the switch. Lower priority PTIs are placed before the higher priority PTIs in the expected configuration or intent and this in turn is the order to which the configuration will be pushed via the deployer module. Default priority is 500.

### Adding a Banner Policy

This example shows how to add a banner policy to a device.



## Deploying New Configurations

There are two ways to deploy the new configurations:

1. Navigate to the Fabric Builder window, right-click on the device and select 'Deploy Config' (this is the recommended way).
2. From the View/Edit Policies window, select the newly added policy, click 'View' to verify the config. If the new config looks good, click the 'Push Config' button to push the new config to the device. Note that 'Push Config' will bypass Configuration Compliance. This option should only be used for exception scenarios such as the case where a new user or SNMP user needs to be added to the switch.

## switch\_freeform Template Usage

The **switch\_freeform** is a special policy template that allows users to specify any freeform config for a device. Usage of the template is as follows:

- Specify switch-level config in the **Switch Freeform Config** parameter.
- The specified config must match the **show run** output with respect to case and newlines. Any mismatch will yield unexpected diffs during deploy.
- An internal **switch\_freeform\_config** CLI policy is created for the specified config.
- Should not use this template for interface configuration except for the SVI interface, as SVI interfaces cannot be configured on the Interfaces page currently.
- Users can create many **switch\_freeform** policies for different configs.
- **switch\_freeform** PTIs are sorted together with the other PTIs based on their policy priorities from low to high.
- A **switch\_freeform** policy can be edited before or after the config is deployed.
- If there is any change in the config content, the previously created internal **switch\_freeform\_config** policy will have its priority changed from a positive to a negative number, and a new internal policy is created for the new config.
- A **negative** priority PTI means that CLIs in the PTI need to be deleted; **Configuration Compliance** will generate the **no** commands accordingly.
- Deleting a **switch\_freeform** policy will change the PTI priority of its internal policy to a negative number.

The following section shows how to create a **switch\_freeform** policy, deploy the policy, and subsequently edit and redeploy the updated policy.

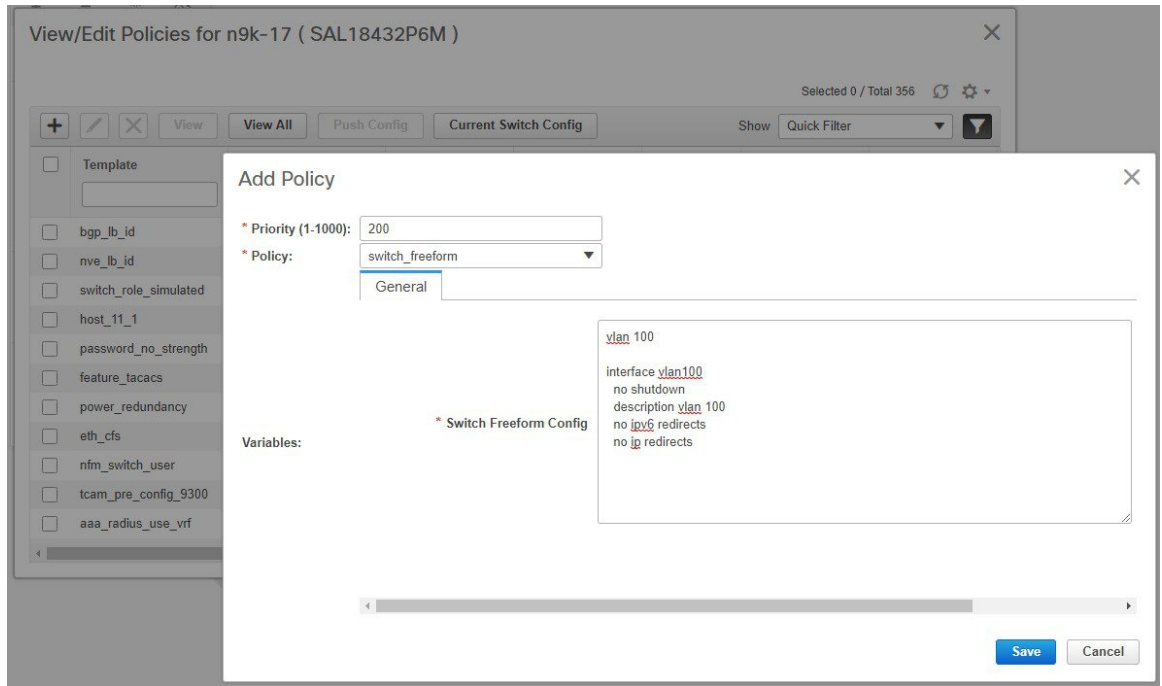
### Example: Create a switch\_freeform policy

To create a **switch\_freeform** policy, perform the following steps:

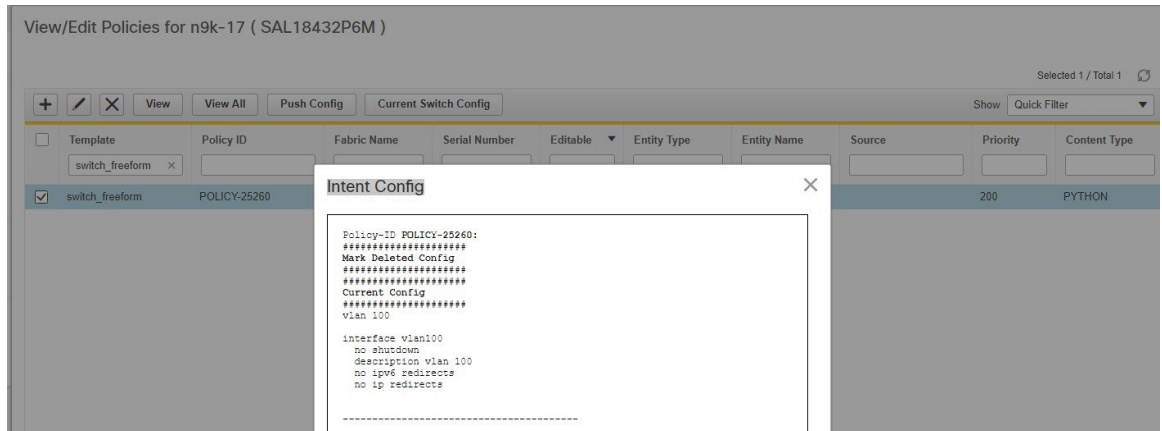
#### Procedure

- 
- Step 1** Select the **switch\_freeform** template from the policy list in the **Add Policy** screen.  
Set the priority and switch freeform config. Save the policy.

Example: Create a switch\_freeform policy

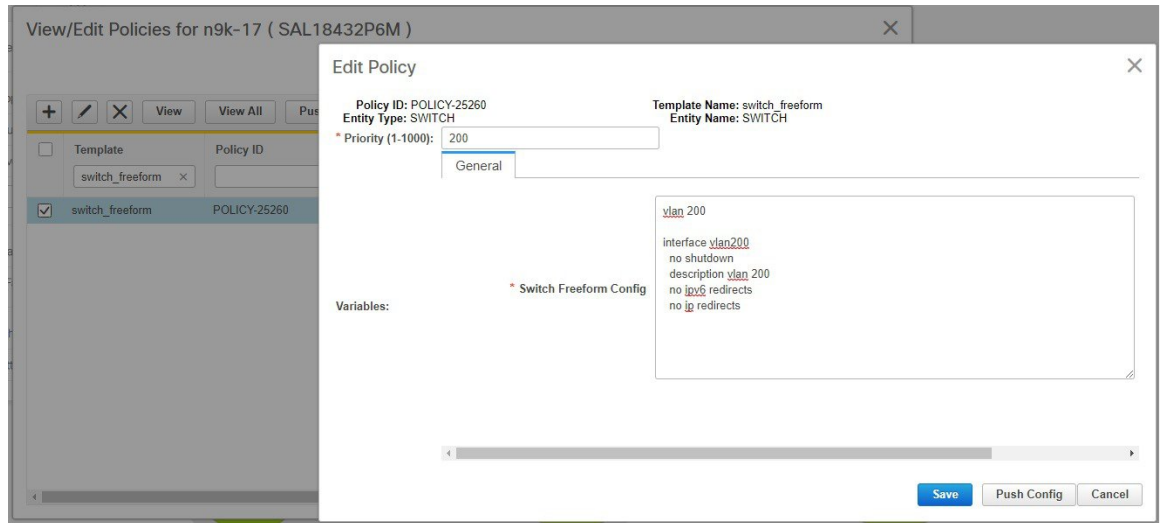


**Step 2** View the intent config of the **switch\_freeform** policy.



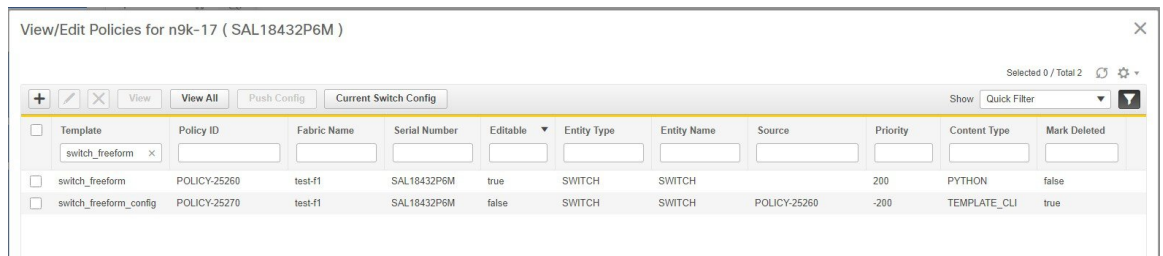
**Step 3** Deploy the switch\_freeform policy from Fabric Builder.

**Step 4** Edit the switch\_freeform policy from the View/Edit Policies window.  
Change the config.

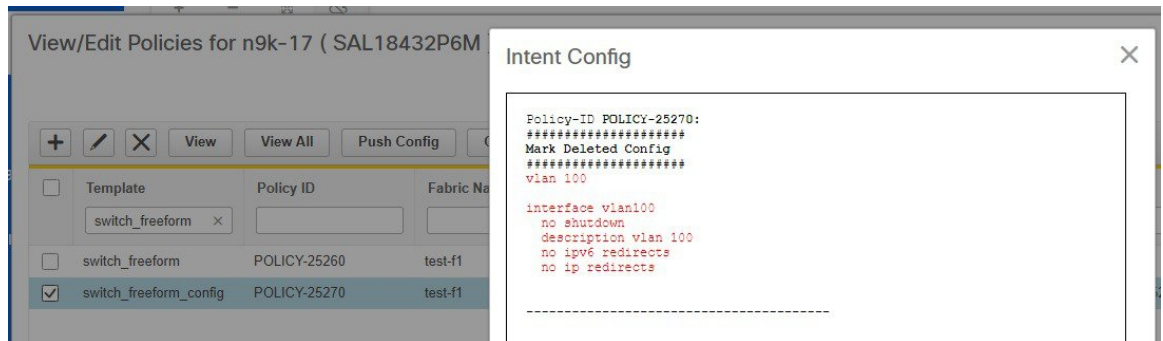


**Step 5** Save the change.

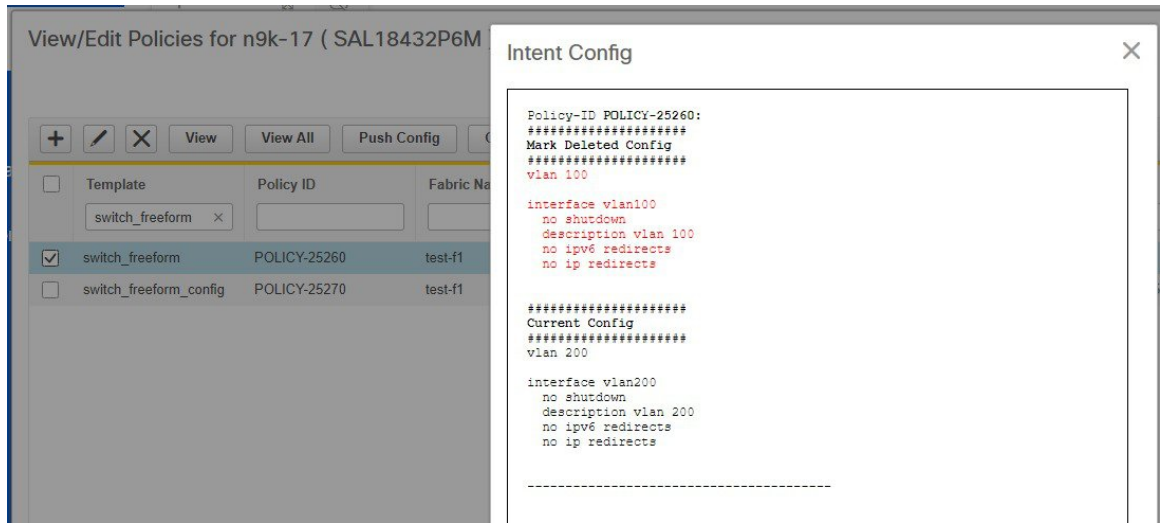
As shown below, the previously created internal **switch\_freeform\_config** policy has its priority changed to a negative number (-200), and the **Mark Deleted** flag is set to true. However, by design, the newly created internal **switch\_freeform\_config** policy is NOT shown.



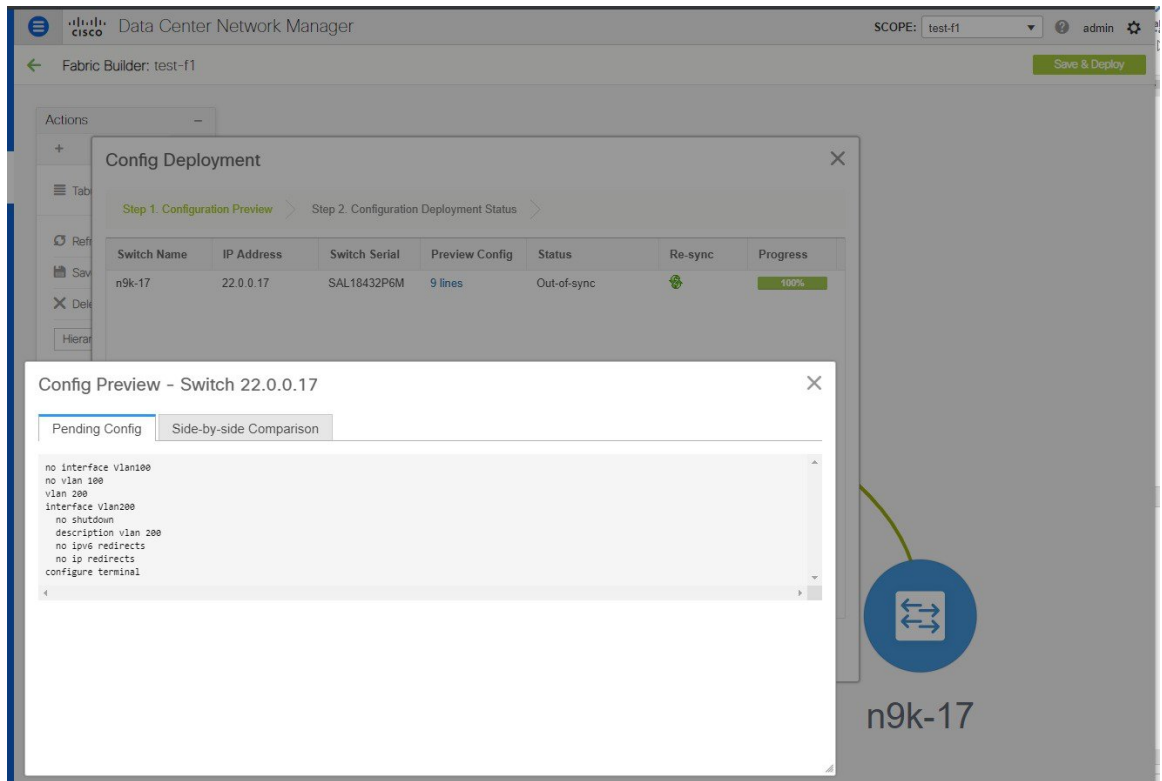
**Step 6** View the intent config of the **mark deleted** internal policy.



**Step 7** View the intent config of the **changed** switch\_freeform policy before deployment. Note that both the **mark-deleted** and **current configs** are shown.



**Step 8** Deploy the changed config from Fabric Builder.



## Changing the Contents of a Template in Use

A template in general, whether it is a policy, fabric or profile template, cannot be modified once it has been instantiated. However, there could be cases where you want to edit the content of a template, like fixing a bug

in the template or changing an already deployed config. This can be achieved by toggling the `template.in_use.check` option in the **Administration > Server Properties** tab.

## Procedure

- Step 1** Change the `template.in_use.check` from **true (default)** to **false**.
- Step 2** Click 'Apply Changes' at the upper righthand corner.
- A warning will be popped up indicating that a restart of DCNM is needed.
- Ignore this warning as no restart is needed for the `in_use` flag to take effect.
- Step 3** Edit the desired template(s).
- Step 4** Go to the Fabric Builder page and click 'Save & Deploy' for the entire fabric.
- This will regenerate PTIs and the updated content will be picked up and used for the expected configuration (or intent).
- Step 5** Once the contents are re-generated and deployed, change the `template.in_use.check` back to **true** to avoid performance issues.

The top screenshot shows the 'Administration / DCNM Server / Server Properties' page in Cisco DCNM. The 'Template Properties' section is expanded, showing the following configuration:

- `template.in_use.check`: false
- `template.use_cache`: true
- `template.server_validation_check`: false
- `template.server_validation_continue_on_error`: false

The bottom screenshot shows the same page with a warning dialog box overlaid. The dialog box contains the following text:

Please restart DCNM SAN service if you update properties other than EMC Callhome properties(server.callhome.enable, server.callhome.xmlDir), Event Forwarding properties (server.forward.event.enable), Template properties (template.in\_use.check property), Event Registration properties(sylog.disable) or fabric.enableNpvDiscovery properties. (Note: please restart all instances if federation is deployed). Please resync vmm if you updated vmm.resync.timer

The 'template.in\_use.check' field is now set back to 'true'.

