

Managing the Template Library for LAN Fabrics, Release 4.1.1

Table of Contents

New and changed information . . . 1
Templates . . . 2
- Creating a New Template . . . 4
- Editing a Template . . . 5
- Importing a Template . . . 6
- Importing a Template from Git . . . 7
- Exporting a Template . . . 8
- Exporting a Template to Git . . . 8
- Template Structure . . . 9
  - Template Format . . . 9
  - Template Variables . . . 14
  - Variable Meta Property . . . 18
  - Variable Annotation . . . 25
  - Templates Content . . . 28
  - Advanced Features . . . 31
  - Report Template . . . 33

Template Usage . . . 36
- Policy Template . . . 36
- Fabric Template . . . 38
- Profile Template . . . 38
  - Additional Notes . . . 38
- Changing the Contents of a Template in Use . . . 39

New and changed information

The following table provides an overview of the significant changes up to this current release. The table does not provide an exhaustive list of all changes or of the new features up to this release.

Release Version	Feature	Description
Nexus Dashboard 4.1.1	Improved navigation and workflow for managing templates	Beginning with Nexus Dashboard 4.1.1, the navigation and workflow for managing templates in your Nexus Dashboard have been enhanced.

Templates

You can add, edit, or delete templates that are configured across different Cisco Nexus, IOS XE, IOS XR, and Cisco MDS platforms using the Nexus Dashboard. The following parameters are displayed for each template that is configured on Nexus Dashboard. Templates support JavaScript. You can use the JavaScript function in a template to perform arithmetic operations and string manipulations in the template syntax.



Beginning with Nexus Dashboard 4.1.1, users will not be able to Create, Edit, or Import any templates in the **Template Library** by default. This change was made to improve the security of the solution, as templates have the ability to execute scripts. Users with the **super-admin** role can optionally enable this function from **Admin > System setting > Fabric Management > Advanced Settings > Admin** tab, by checking the **Allow template create, edit, duplicate and upload operations** check box and then **Save**.

1. Choose **Manage > Template Library**.

Template Table Fields and Descriptions

Field	Description
Name	Specifies the template name.
Supported Platforms	Specifies the platforms that the template support.
Type	Specifies the template type.
Sub Type	Specifies the template sub type.
Modified	Specifies the date and time of the template modification.
Tags	Specifies if the template is tagged to a fabric or a device.
Description	Specifies the template description.
Reference Count	Specifies the number of times a template is used.

2. Click the table header to sort the entries in alphabetical order of that parameter.



Templates with errors are not listed on the **Templates** page. You cannot import templates with errors. To import templates with errors, fix the errors, and then import the templates.

The following table describes the action items, in the **Actions** drop-down list, that appear on the **Templates** page.

Templates Actions and Descriptions

Actions	Description
Create new template	Allows you to create a new template. For more information, see Creating a New Template.
Edit template properties	Allows you to edit the template properties. You can edit only one template at a time. For more information, see Editing a Template.

Actions	Description
Edit template content	Allows you to edit the template content. You can edit only one template at a time. For more information, see Editing a Template.
Duplicate template	Allows you to duplicate the selected template with a different name. You can edit the template as required. You can duplicate only one template at a time. To duplicate a template, select the check box next to the template that you want to duplicate and choose **Actions > Duplicate template**. The **Duplicate Template** page appears. Specify a name for the duplicated template. For more information about editing the duplicated template, see Editing a Template.
Delete template	Allows you to delete a template. You can delete more than one template in a single instance. You can delete the user-defined templates. However, you cannot delete the predefined templates. To delete a template, select the check box next to the template that you want to delete and choose **Delete template**. A warning message appears. If you are sure you want to delete the template, click **Confirm**. If not, click **Cancel**. If the template is in use or is a shipping template, you cannot delete it, and an error message appears.  Select multiple templates to delete them at the same time. To delete the template permanently, delete the template that is located in your local directory: Cisco Systems\dcn\ndfc\data\templates\
Import	Allows you to import a template from your local directory, one at a time. For more information, see Importing a Template.
Import as Zip	Allows you to import a .zip file containing more than one template that is bundled in a .zip format. All the templates in the .zip file are extracted and listed in the table as individual templates. For more information, see Importing a Template.
Import From Git	Allows you to import a nondefault Nexus Dashboard template file from your Git repository to your Nexus Dashboard templates. For more information, see Importing a Template from Git.

Actions	Description
Export	Allows you to export the template configuration to a local directory location. You can export multiple templates at a time. To export a template, use the check box next to the templates to select the required templates and then choose **Actions > Export**. Select a location on your local system directory to store the template file. Click **Save**. The template file is exported to your local directory as a single template file or as a compressed .zip file, if you have selected multiple templates. You can export multiple templates as a single .zip file.
Export To Git	Allows you to export a nondefault Nexus Dashboard template to a Git repository. For more information, see Exporting a Template to Git.

You can only view templates with the **network-operator** role. You cannot create, edit, or save templates with this role. However, you can create or edit templates with the **network-stager** role.

Creating a New Template

Cisco Nexus Dashboard UI Navigation

- Choose **Manage > Template Library**.

To create user-defined templates and schedule jobs from the Nexus Dashboard Web UI, perform the following steps:

1. In the **Templates** window, from the **Actions** drop-down list, choose **Create new template**.

   The **Create Template** window appears.

2. In the **Template Properties** page of the window, specify a template name, description, tags, and choose supported platforms for the new template. Next, choose a template type and a sub template type from the drop-down lists. Choose a content type for the template from the drop-down list.

   The base templates are CLI templates.

3. Click **Next** to continue editing the template or click **Cancel** to discard the changes.

   The edited template properties are displayed in the **Template Content** page of the **Edit Template** window. For information about the structure of the Configuration Template, see the *Template Structure* section.

4. Click **Validate** to validate the template syntax.

   You can continue to save the template if there are warnings only. However, if

there is an error, you must edit the templates to fix the errors before you proceed. Click the line number under the Start Line column to locate the error in the template content. You will get an error if you validate a template that does not have a template name.

5. Click **Help** to open the **Editor Help** pane on the right.

   This window contains more information about the format, variables, content and data types used to build the template. Close the **Editor Help** pane.

6. Click **Errors** and **Warnings** if the links are displayed. If there are no errors or warnings, the links are not available. If errors or warnings are present, and you click the links, the **Errors & Warnings** pane appears on the right displaying the errors and warnings. Close the **Errors & Warnings** pane.
7. To build the template content, select the required theme, key binding, and font size from the drop-down list.
8. Click **Finish** to complete editing of the template, click **Cancel** to discard the changes, click **Previous** to go to the **Template Properties** page.

   The page with the message that the template was created appears. The page also displays the template name, type, and sub type, and the platforms. You can also click **Create another template** to create one more template or click **Edit <template name> template** to edit the template that was just edited.

9. Close the **Edit Template** window or click **Back to template library** to go back to the **Templates** window.

Editing a Template

Nexus Dashboard allows you to edit user-defined templates. However, you cannot edit the predefined templates and templates that are already published.

Use the **Edit Template** page to first edit the template properties and then edit the template content. Furthermore, you can edit either only the template properties using the **Edit template properties** action or only the template content using the **Edit template content** action. In other words, you can edit the template properties at one instance, and then, edit the template content at another instance. You can also use this page to view the template properties and content.

Before You Begin

To edit templates, ensure you have the necessary permissions:

1. In Nexus Dashboard, choose **Admin > System Settings > Fabric management**.
2. In the **Advanced settings** area, click **Admin**.
3. Check the **Allow template create, edit, duplicate and upload operations** check box.
4. Click **Save**.

Perform the following steps to edit the template properties and then edit the template content:

1. In Nexus Dashboard, choose **Manage > Template Library**.
2. In the **Templates** page, select a template. From the **Actions** drop-down list, choose **Edit**

template properties.

The **Edit Template** page appears.

3. The **Template Properties** page displays the name of the template along with its description, supported platforms, tags, and content type. You can edit the template description and tags. To edit the supported platforms, clear the selected check boxes to select other switches. Next, choose a template type and a sub template type from the drop-down lists.
4. Click **Next** to continue editing the template or click **Cancel** to discard the changes.

   The edited template properties are displayed in the **Template Content** page of the **Edit Template** page.

5. Click **Validate** to validate the template syntax.

   You can continue to save the template if there are warnings only. However, if there is an error, you must edit the templates to fix the errors before you proceed. Click the line number under the Start Line column to locate the error in the template content. You will get an error if you validate a template that does not have a template name.

6. Click **Help** to open the **Editor Help** pane on the right.

   This page contains more information about the format, variables, content and data types used to build the template. Close the **Editor Help** pane.

7. Click **Errors** and **Warnings** if the links are displayed. If there are no errors or warnings, the links are not available. If errors or warnings are present, and you click the links, the **Errors & Warnings** pane appears on the right displaying the errors and warnings. Close the **Errors & Warnings** pane.
8. To build the template content, select the required theme, key binding, and font size from the drop-down list.
9. Click **Finish** to complete editing of the template, click **Cancel** to discard the changes, click **Previous** to go to the **Template Properties** page.

   The page with the message that the template is saved appears. The page also displays the template name, type, and sub type, and the platforms. You can also click **Create another template** to create one more template or click **Edit <template name> template** to edit the template that was just edited.

10. Close the **Edit Template** page or click **Back to template library** to go back to the **Templates** page.

Importing a Template

1. Navigate to **Manage > Template Library**.
2. Choose **Import** from the **Actions** drop-down list.

   The **Import Template** dialog box displays.

3. Browse to the template that you saved on your computer and drag it to the **Import Template** dialog box.

4. Click **OK** to import the template or click **Cancel** to discard the template.

   Follow the same procedure when importing a zipped template.

   > The "\n" in the template is considered a new-line character when importing and editing templates, but it works fine when importing a template as a zipped file.

   After importing a zipped template file, you receive either a success message or an error message.

5. Click **OK**.
6. You can edit the template parameters and content, if necessary.

   For more information, see Editing a Template.

   > When importing a zipped template file, the **Edit Template** page may not appear. You can edit the template parameters and content, if necessary, using either the **Edit template properties** or the **Edit template content** option.

7. If you do not want to edit the template properties or content, then keep clicking **Next** or **Finish**, and **Back to template library** to go back to the **Templates** page.

Importing a Template from Git

You can import your nondefault Nexus Dashboard template files from your Git repository to your Nexus Dashboard templates.

> Only Nexus Dashboard template files are imported from Git. Other files are not supported.

Before You Begin

1. Create a Git repository.
2. Generate a Git token on github.com to import your Nexus Dashboard templates from your Git repository to your Nexus Dashboard templates.

Configure Your Git IP Route Using Cisco Nexus Dashboard

1. Navigate to **Admin > System Settings > General > Routes (Management Network Routes)** in the Cisco Admin Console GUI.
2. Click the **Edit** icon in the **Routes** tile.
3. Click **Add Management Network Routes** and provide the github.com IP route.
4. Click **Save**.
5. Edit the template parameters and content in Nexus Dashboard.

   For more information, see Editing a Template.

Configure a Proxy Server Using Cisco Nexus Dashboard

If you need to access a global public Git repository, configure a proxy server using Cisco Nexus

Dashboard.

1. Navigate to **Admin > System Settings > General > Proxy Configuration** in the Cisco Admin Console GUI.
2. Click the **Edit** icon in the **Proxy Configuration** tile.
3. Click **Add Server** and provide the necessary information.
4. Click **Save**.

Import a Template from a Git Repository to Nexus Dashboard

1. Navigate to **Manage > Template Library**.
2. Choose **Import From Git** from the **Actions** drop-down list.

   The **Import From Git** dialog box appears.
3. Enter the required information.

Field	Description
Git Repository	Specifies the directory location of your Git repository.
Branch	Specifies the name of your Git branch.
Git Token	Specifies the Git token.

4. Click **Fetch**.

   You should receive an **Information** dialog box from Git listing all of the files to be imported on Nexus Dashboard before the import process.

   You should see your Nexus Dashboard templates imported from Git to Nexus Dashboard after the import process.

Exporting a Template

1. Navigate to **Manage > Template Library**.
2. Choose **Export** from the **Actions** drop-down list.

   Nexus Dashboard exports your template files to your local system.

Exporting a Template to Git

You can export your nondefault Nexus Dashboard templates to GIT.

Before You Begin

1. Create a Git repository.
2. Generate a Git token on github.com to export your nondefault Nexus Dashboard templates to your GIT repository.

Configure Your Git IP Route Using Cisco Nexus Dashboard

1. Navigate to **Admin > System Settings > General > Routes (Management Network Routes)** in the Cisco Admin Console GUI.
2. Click the **Edit** icon in the **Routes** tile.
3. Click **Add Management Network Routes** and provide the github.com IP route.
4. Click **Save**.
5. Edit the template parameters and content in Nexus Dashboard.

   For more information, see Editing a Template.

Configure a Proxy Server Using Cisco Nexus Dashboard

If you need to access a global public Git repository, configure a proxy server using Cisco Nexus Dashboard.

1. Navigate to **Admin > System Settings > General > Proxy Configuration** in the Cisco Admin Console GUI.
2. Click the **Edit** icon in the **Proxy Configuration** tile.
3. Click **Add Server** and provide the necessary information.
4. Click **Save**.

Export to git

1. Navigate to **Manage > Template Library**.
2. Choose **Export To Git** from the **Actions** drop-down list.

   The **Export To Git** dialog box appears.
3. Enter the required information.

Field	Description
Git Repository	Specifies the directory location of your Git repository.
Branch	Specifies the name of your Git branch.
Git Token	Specifies the Git token.

4. Click **Fetch** to fetch all of the directories from your Git repository.

   You should see your Nexus Dashboard templates in your Git repository.

Template Structure

The configuration template content mainly consists of four parts. Click the **Help** icon next to the **Template Content** for information about editing the content of the template.

Template Format

This section describes the basic information of the template. The possible fields are as detailed in the table below.

Property Name	Description	Valid Values	Optional?
name	The name of the template	Text	No
description	Brief description about the template	Text	Yes
userDefined	Indicates whether the user created the template. Value is "true" if user created.	"true" or "false"	Yes
supportedPlatforms	List of device platforms supports this configuration template. Specify "All" to support all platforms.	N1K, N3K, N3500, N4K, N5K, N5500, N5600, N6K, N7K, N9K, MDS, VDC, N9K-9000v, IOS-XE, IOS-XR, Others, All Nexus Switches list separated by comma.	No
templateType	Specifies the type of Template used.	• CLI • POAP POAP option is not applicable for Nexus Dashboard LAN Fabric deployment. • POLICY • SHOW • PROFILE • FABRIC • ABSTRACT • REPORT	Yes

Property Name	Description	Valid Values	Optional?

templateSubType	Specifies the sub type associated with the template.	• CLI
		  ◦ N/A
		• POAP
		  ◦ N/A
		  ◦ VXLAN
		  ◦ FABRICPATH
		  ◦ VLAN
		  ◦ PMN
		  POAP option is not applicable for Cisco Nexus Dashboard Fabric Controller LAN Fabric deployment.
		• POLICY
		  ◦ VLAN
		  ◦ INTERFACE_VLAN
		  ◦ INTERFACE_VPC
		  ◦ INTERFACE_ETHERNET
		  ◦ INTERFACE_BD
		  ◦ INTERFACE_PORT_CHANNEL
		  ◦ INTERFACE_FC
		  ◦ INTERFACE_MGMT
		  ◦ INTERFACE_LOOPBACK
		  ◦ INTERFACE_NVE
		  ◦ INTERFACE_VFC
		  ◦ INTERFACE_SAN_PORT_CHANNEL
		  ◦ DEVICE
		  ◦ FEX
		  ◦ INTRA_FABRIC_LINK
		  ◦ INTER_FABRIC_LINK
		  ◦ INTERFACE	

Property Name	Description	Valid Values	Optional?

templateSubType (continued)	Specifies the sub type associated with the template.	• SHOW
		  ◦ VLAN
		  ◦ INTERFACE_VLAN
		  ◦ INTERFACE_VPC
		  ◦ INTERFACE_ETHERNET
		  ◦ INTERFACE_BD
		  ◦ INTERFACE_PORT_CHANNEL
		  ◦ INTERFACE_FC
		  ◦ INTERFACE_MGMT
		  ◦ INTERFACE_LOOPBACK
		  ◦ INTERFACE_NVE
		  ◦ INTERFACE_VFC
		  ◦ INTERFACE_SAN_PORT_CHANNEL
		  ◦ DEVICE
		  ◦ FEX
		  ◦ INTRA_FABRIC_LINK
		  ◦ INTER_FABRIC_LINK
		  ◦ INTERFACE
		• PROFILE
		  ◦ VXLAN
		• FABRIC
		  ◦ NA	

Property Name	Description	Valid Values	Optional?

templateSubType (continued)	Specifies the sub type associated with the template.	• ABSTRACT
		  ◦ VLAN
		  ◦ INTERFACE_VLAN
		  ◦ INTERFACE_VPC
		  ◦ INTERFACE_ETHERNET
		  ◦ INTERFACE_BD
		  ◦ INTERFACE_PORT_CHANNEL
		  ◦ INTERFACE_FC
		  ◦ INTERFACE_MGMT
		  ◦ INTERFACE_LOOPBACK
		  ◦ INTERFACE_NVE
		  ◦ INTERFACE_VFC
		  ◦ INTERFACE_SAN_PORT_CHANNEL
		  ◦ DEVICE
		  ◦ FEX
		  ◦ INTRA_FABRIC_LINK
		  ◦ INTER_FABRIC_LINK
		  ◦ INTERFACE
		• REPORT
		  ◦ UPGRADE
		  ◦ GENERIC	

Property Name	Description	Valid Values	Optional?

contentType		• CLI
		  ◦ TEMPLATE_CLI
		• POAP
		  ◦ TEMPLATE_CLI
		POAP option is not applicable for Cisco Nexus Dashboard Fabric Controller LAN Fabric deployment.
		• POLICY
		  ◦ TEMPLATE_CLI
		  ◦ PYTHON
		• SHOW
		  ◦ TEMPLATE_CLI
		• PROFILE
		  ◦ TEMPLATE_CLI
		  ◦ PYTHON
		• FABRIC
		  ◦ PYTHON
		• ABSTRACT
		  ◦ TEMPLATE_CLI
		  ◦ PYTHON
		• REPORT
		  ◦ PYTHON
			Yes

implements	Used to implement the abstract template.	Text	Yes
dependencies	Used to select the specific feature of a switch.	Text	Yes
published	Used to Mark the template as read only and avoids changes to it.	" true" or " false"	Yes

Template Variables

This section contains declared variables, the data type, default values, and valid values conditions for the parameters that are used in the template. These declared variables are used for value substitution in the template content section during the dynamic command generation process. Also these variables are used in decision making and in iteration blocks in the template content section. Variables have predefined data types. You can also add a description about the variable. The following table describes the syntax and usage for the available datatypes.

Variable Type	Valid Value	Iterative?
boolean	true|false	No
enum	Example: running-config, startup-config	No
float	Floating number format	No
floatRange	Example: 10.1,50.01	Yes
Integer	Any number	No
integerRange	Contiguous numbers separated by "-" Discrete numbers separated by "," Example: 1-10,15,18,20	Yes
interface	Format: <if type><slot>[/<sub slot>]/<port> Example: eth1/1, fa10/1/2 etc.	No
interfaceRange	Example: eth10/1/20-25, eth11/1-5	Yes
ipAddress	IPv4 OR IPv6 address	No

Variable Type	Valid Value	Iterative?
ipAddressList	You can have a list of IPv4, IPv6, or a combination of both types of addresses. Example 1: 172.22.31.97, 172.22.31.99, 172.22.31.105, 172.22.31.109 Example 2: 2001:0db8:85a3:0000:0000:8a2e:0370:7334, 2001:0db8:85a3:0000:0000:8a2e:0370:7335, 2001:0db8:85a3:1230:0000:8a2f:0370:7334 Example 3: 172.22.31.97, 172.22.31.99, 2001:0db8:85a3:0000:0000:8a2e:0370:7334, 172.22.31.254	Yes
ipAddressWithoutPrefix	Example: 192.168.1.1 or Example: 1:2:3:4:5:6:7:8	No
ipV4Address	IPv4 address	No
ipV4AddressWithSubnet	Example: 192.168.1.1/24	No
ipV6Address	IPv6 address	No
ipV6AddressWithPrefix	Example: 1:2:3:4:5:6:7:8 22	No
ipV6AddressWithSubnet	IPv6 Address with Subnet	No

Variable Type	Valid Value	Iterative?
ISISNetAddress	Example: 49.0001.00a0.c96b.c490.00	No
long	Example: 100	No
macAddress	14 or 17 character length MAC address format	No

string	Free text, for example, used for the description of a variable
	
	Example:
	string scheduledTime
	{
	regularExpr=^([01]\d|2[0-3]):([0-5]\d)$;
	}
		No

string[]	Example: {a,b,c,str1,str2}	Yes

struct	Set of parameters that are bundled under a single variable.
	
	struct <structure name declaration > {
	<parameter type> <parameter 1>;
	<parameter type> <parameter 2>;
	.....
	} [<structure_inst1>] [, <structure_inst2>]
	[, <structure_array_inst3 []>];
	
	struct interface_detail {
	string inf_name;
	string inf_description;
	ipAddress inf_host;
	enum duplex {
	validValues = auto, full, half;
	};
	}myInterface, myInterfaceArray[];
		No
		
		If the struct variable is declared as an array, the variable is iterative.

wwn (Available only in Cisco Nexus Dashboard Fabric Controller Web Client)	Example: 20:01:00:08:02:11:05:03	No

Variable Meta Property

Each variable that is defined in the template variable section has a set of meta properties. The meta properties are mainly the validation rules that are defined for the variable.

The following tables describe the various meta properties applicable for the available variable types.

Variable Meta Properties Table, Part 1

Variable Type	Description	Variable Meta Property				
		default Value	valid Values	decimal Length	min	max
boolean	A boolean value. Example: true	Yes				
enum			Yes			
float	Signed real number. Example: 75.56, -8.5	Yes	Yes	Yes	Yes	Yes
floatRange	Range of signed real numbers. Example: 50.5 - 54.75	Yes	Yes	Yes	Yes	Yes
integer	Signed number. Example: 50, -75	Yes	Yes		Yes	Yes
integerRange	Range of signed numbers. Example: 50-65	Yes	Yes		Yes	Yes
interface	Specifies interface/port. Example: Ethernet 5/10	Yes	Yes			
interfaceRange		Yes	Yes			
ipAddress	IP address in IPv4 or IPv6 format.	Yes				

ipAddressList	You can have a list of IPv4, IPv6, or a combination of both types of addresses. Example 1: 172.22.31.97, 172.22.31.99, 172.22.31.105, 172.22.31.109 Example 2: 2001:0db8:85a3:0000:0000:8a2e:0370:7334, 2001:0db8:85a3:0000:0000:8a2e:0370:7335, 2001:0db8:85a3:1230:0000:8a2f:0370:7334 Example 3: 172.22.31.97, 172.22.31.99, 2001:0db8:85a3:0000:0000:8a2e:0370:7334, 172.22.31.254 Separate the addresses in the list using commas and not hyphens.	Yes				
ipAddressWithoutPrefix	IPv4 or IPv6 Address (does not require prefix/subnet).					
ipV4Address	IPv4 address	Yes				
ipV4AddressWithSubnet	IPv4 Address with Subnet	Yes				
ipV6Address	IPv6 address	Yes				
ipV6AddressWithPrefix	IPv6 Address with prefix	Yes				
ipV6AddressWithSubnet	IPv6 Address with Subnet	Yes				
ISISNetAddress	Example: 49.0001.00a0.c96b.c490.00					
long	Example: 100	Yes			Yes	Yes

macAddress	MAC address					

string	literal string
	
	Example for string Regular expression:
	string scheduledTime
	{
	
	regularExpr=^([01]\d|2[0-3]):([0-5]\d)$;
	}
		Yes				

string[]	string literals that are separated by a comma (,) Example: {string1, string2}	Yes				
struct	Set of parameters that are bundled under a single variable. struct <structure name declaration > { <parameter type> <parameter 1>; <parameter type> <parameter 2>; ..... } [<structure_inst1>] [, <structure_inst2>] [, <structure_array_inst3[]>];					
wwn	WWN address					

Variable Meta Properties Table, Part 2

Variable Type	Description	Variable Meta Property						
		min Slot	max Slot	min Port	max Port	min Length	max Length	regular Expr

boolean	A boolean value. Example: true							
enum								
float	Signed real number. Example: 75.56, -8.5							
floatRange	Range of signed real numbers. Example: 50.5 - 54.75							
integer	signed number Example: 50, -75							
integerRange	Range of signed numbers. Example: 50-65							
interface	Specifies interface/port. Example: Ethernet 5/10	Yes	Yes	Yes	Yes			
interfaceRange		Yes	Yes	Yes	Yes			
ipAddress	IP address in IPv4 or IPv6 format .							

ipAddressList	You can have a list of IPv4, IPv6, or a combination of both types of addresses. Example 1: 172.22.31.97, 172.22.31.99, 172.22.31.105, 172.22.31.109 Example 2: 2001:0db8:85a3:0000:0000:8a2e:0370:7334, 2001:0db8:85a3:0000:0000:8a2e:0370:7335, 2001:0db8:85a3:1230:0000:8a2f:0370:7334 Example 3: 172.22.31.97, 172.22.31.99, 2001:0db8:85a3:0000:0000:8a2e:0370:7334, 172.22.31.254 Separate the addresses in the list using commas and not hyphens.							
ipAddressWithoutPrefix	IPv4 or IPv6 address (does not require prefix/subnet).							
ipV4Address	IPv4 address.							
ipV4AddressWithSubnet	IPv4 address with Subnet.							
ipV6Address	IPv6 address.							
ipV6AddressWithPrefix	IPv6 address with prefix.							
ipV6AddressWithSubnet	IPv6 address with Subnet.							

ISISNetAddress	Example: 49.0001.00a0.c96b.c490.00							
long	Example: 100							
macAddress	MAC address							
string	literal string Example for string Regular expression: string scheduledTime { regularExpr=^([01]\d|2[0-3]):([0-5]\d)$; }					Yes	Yes	Yes
string[]	string literals that are separated by a comma (,) Example: {string1, string2}							

struct	Set of parameters that are bundled under a single variable.
	
	struct <structure name declaration > {
	<parameter type> <parameter 1>;
	<parameter type> <parameter 2>;
	.....
	}
	[<structure_inst1>] [, <structure_inst2>] [, <structure_array_inst3 []>];							

wwn	WWN address							

Example: Meta Property Usage

```
##template variables

integer VLAN_ID {
min = 100;
max= 200;
};

string USER_NAME {
defaultValue = admin123;
minLength = 5;
};

struct interface_a{
    string inf_name;
    string inf_description;
    ipAddress inf_host;
    enum duplex {
        validValues = auto, full, half;
    };
}myInterface;
```

```
##
```

Variable Annotation

You can configure the variable properties marking the variables using annotations.



Variable Annotations are available for POAP only. However, the annotations do not impact on the template type "CLI". The following annotations can be used in the template variable section.

Annotation Key	Valid Values	Description
AutoPopulate	Text	Copies values from one field to another
DataDepend	Text	
Description	Text	Description of the field appearing in the window
DisplayName	Text Enclose the text with quotes, if there is space.	Display name of the field appearing in the window
Enum	Text1, Text2, Text3, and so on	Lists the text or numeric values to select from
IsAlphaNumeric	"true" or "false"	Validates if the string is alphanumeric
IsAsn	"true" or "false"	
IsDestinationDevice	"true" or "false"	
IsDestinationFabric	"true" or "false"	
IsDestinationInterface	"true" or "false"	
IsDestinationSwitchName	"true" or "false"	
IsDeviceID	"true" or "false"	
IsDot1qId	"true" or "false"	
IsFEXID	"true" or "false"	
IsGateway	"true" or "false"	Validates if the IP address is a gateway
IsInternal	"true" or "false"	Makes the fields internal and does not display them on the window Use this annotation only for the ipAddress variable.

Annotation Key	Valid Values	Description
IsManagementIP	" true" or " false" This annotation must be marked only for variable " ipAddress" .	
IsMandatory	" true" or " false"	Validates if a value should be passed to the field mandatorily
IsMTU	" true" or " false"	
IsMultiCastGroupAddress	" true" or " false"	
IsMultiLineString	" true" or " false"	Converts a string field to multiline string text area
IsMultiplicity	" true" or " false"	
IsPassword	" true" or " false"	
IsPositive	" true" or " false"	Checks if the value is positive
IsReplicationMode	" true" or " false"	
IsShow	" true" or " false"	Displays or hides a field on the window
IsSiteId	" true" or " false"	
IsSourceDevice	" true" or " false"	
IsSourceFabric	" true" or " false"	
IsSourceInterface	" true" or " false"	
IsSourceSwitchName	" true" or " false"	
IsSwitchName	" true" or " false"	
IsRMID	" true" or " false"	
IsVPCDomainID	" true" or " false"	
IsVPCID	" true" or " false"	
IsVPCPeerLinkPort	" true" or " false"	
IsVPCPeerLinkPortChannel	" true" or " false"	
IsVPCPortChannel	" true" or " false"	
Password	Text	Validates the password field
PeerOneFEXID	" true" or " false"	
PeerTwoFEXID	" true" or " false"	
PeerOnePCID	" true" or " false"	
PeerTwoPCID	" true" or " false"	
PrimaryAssociation		
ReadOnly	" true" or " false"	Makes the field read-only
ReadOnlyOnEdit	" true" or " false"	
SecondaryAssociation	Text	

Annotation Key	Valid Values	Description
Section		
UsePool	" true" or " false"	
UseDNSReverseLookup		
Username	Text	Displays the username field on the window
Warning	Text	Provides text to override the Description annotation

Example: AutoPopulate Annotation

```
##template variables
string BGP_AS;
 @(AutoPopulate=" BGP_AS" )
  string SITE_ID;
##
```

Example: DisplayName Annotation

```
##template variables
@(DisplayName=" Host Name" , Description = " Description of the host" )
String hostname;
@(DisplayName=" Host Address" , Description = "  test description"  IsManagementIP=true)
ipAddress hostAddress;
##
```

Example: IsMandatory Annotation

```
##template variables
@(IsMandatory=" ipv6!=null" )
ipV4Address ipv4;
@(IsMandatory=" ipv4!=null" )
ipV6Address ipv6;
##
```

Example: IsMultiLineString Annotation

```
##template variables
@(IsMultiLineString=true)
string EXTRA_CONF_SPINE;
##
```

IsShow Annotation

```
Example 1##template variables
boolean isVlan;
@(IsShow=" isVlan==true" )
integer vlanNo;
##

Example 2##template variables
boolean enableScheduledBackup;
@(IsShow=" enableScheduledBackup==true" ,Description=" Server time" )
string scheduledTime;
##
The condition " enableScheduledBackup==true"  evaluates to true/false

Example 3##template variables
@(Enum=" Manual,Back2BackOnly,ToExternalOnly,Both" )
string VRF_LITE_AUTOCONFIG;
@(IsShow=" VRF_LITE_AUTOCONFIG!=Manual" ,  Description=" Target Mask" )
integer DCI_SUBNET_TARGET_MASK
##
The condition " VRF_LITE_AUTOCONFIG!=Manual"  matches string comparison to evaluate to
true or false
```

Example: Warning Annotation

```
##template variables
@(Warning=" This is a warning msg" )
  string SITE_ID;
##
```

Templates Content

This section includes the configuration commands and any parameters that you want to include in the template. These commands can include the variables declared in the template variables section. During the command generation process the variable values are substituted appropriately in the template content.



You must specify the commands that you include as if you were entering them in the global configuration command mode on any device. You must consider the command mode when you include commands.

Template content is governed by the usage of variables.

- Scalar variables: does not take a range or array of values which cannot be used for iteration (In

the variable types table those marked iterate-able as 'No'). Scalar variables must be defined inside the template content.

```
Syntax: $$<variable name>$$
Example: $$USER_NAME$$
```

- Iterative variables: used for block iteration. These loop variable must be accessed as shown below inside the iteration block.

```
Syntax:@<loop variable>
Example:
foreach val in $$INTEGER_RANGE_VALUE$$ {
@val
}
```

- Scalar Structure Variable: Structure member variables can be accessed inside the template content.

```
Syntax: $$<structure instance name>.<member variable name>$$
Example: $$myInterface.inf_name$$
```

- Array Structure Variable: Structure member variables can be accessed inside the template content.

```
Syntax: $$<structure instance name>.<member variable name>$$
Example: $$myInterface.inf_name$$
```

In addition to the template variables, you can use the conditional and iterative command generation using the following statements:

- if-else if-else Statement: makes a logical decision in inclusion/exclusion of set of configuration command based on the value assigned for the variable in it.

```
Syntax: if(<operand 1>  <logical operator>  <operand 2>){
command1 ..
command2..
..
}
else  if (<operand 3> <logical operator> <operand 4> )
{
Command3 ..
Command4..
..
```

```
}
else
{
Command5 ..
Command6..
..
}
Example: if-else if-else statement
if($$USER_NAME$$ == 'admin'){
Interface2/10
no shut
}
else {
Interface2/10
shut
}
```

- foreach Statement: used for iterating a block of commands. The iteration is performed based on the assigned loop variable value.

```
Syntax:
foreach <loop index variable> in $$<loop variable>$$ {
@<loop index variable> ..
}
Example: foreach Statement
foreach ports in $$MY_INF_RANGE$${
interface @ports
no shut
}
```

- Optional parameters: By default all parameters are mandatory. To make a parameter optional, you must annotate the parameter.

In the variable section, you can include the following command:

- @(IsMandatory=false)
- Integerfrequency;

  In the template content section, a command can be excluded or included without using "if" condition check, by assigning a value to the parameter. The optional command can be framed as below:

- probeicmp[frequencyfrequency-value][timeoutseconds][retry-countretry-count-value]

Advanced Features

The following are the advanced features available to configure templates.

- Assignment Operation

  Config template supports assignment of variable values inside the template content section. The values are validated for the declared data type of the variable. If there is a mismatch, the value is not assigned.

  Assignment operation can be used under the following guidelines:

  - The operator on the left must be any of the template parameters or a for loop parameter.
  - The operator on the right values can be any of the values from template parameters, for loop parameters, literal string values surrounded by quotes or simple string values.

  If a statement does not follow these guidelines, or if it does not suit this format, it will not be considered as assignment operation. It is substituted during command generation like other normal lines.

```
Example: Template with assignment operation
##template properties
name =vlan creation;
userDefined= true;
supportedPlatforms = All;
templateType = CLI;
published = false;
##
##template variables
integerRange vlan_range;
@(internal=true)
integer vlanName;
##
##template content
foreach vlanID in $$vlan_range$${
vlan @vlanID
$$vlanName$$=@vlanID
name myvlan$$vlanName$$
}
##
```

- Evaluate methods

  Config template uses the Java runtime provided Java script environment to perform arithmetic operations (such as ADD, SUBTRACT, and so on), string manipulations, and so on.

  Locate the JavaScript file in the template repository path. This file contains primary set of arithmetic, string functions. You can also add custom JavaScript methods.

These methods can be called from config template content section in below format:

```
Example1:
$$somevar$$ = evalscript(add, " 100" , $$anothervar$$)
```

Also the *evalscript* can be called inside if conditions as below:

```
if($$range$$ > evalscript(sum, $$vlan_id$$,  -10)){
do something...
}
```

You can call a method that is located at the backend of the Java script file.

- Dynamic decision

Config template provides a special internal variable " LAST_CMD_RESPONSE" . This variable stores the last command response from the device during the execution of the command. This can be used in the config template content to make dynamic decisions to deliver the commands that are based on the device condition.



The if block must be followed by an else block in a new line, which can be empty.

An example use case to create a VLAN, if it is does not exist on the device.

```
Example: Create VLAN
##template content
show vlan id $$vlan_id$$
if ($$LAST_CMD_RESPONSE$$ contains " not found" ) {
vlan $$vlan_id$$
} else {
}
##
```

This special implicit variable can be used only in the " IF" blocks.

- Template referencing

You can have a base template with all the variables defined. This base template can be imported to multiple templates. The base template content is substituted in the appropriate place of the extending template. The imported template parameters and the contents can be accessed inside the extending template.

```
Example: Template Referencing
Base template:
##template properties
```

```
    name =a vlan base;
    userDefined= true;
    supportedPlatforms = All;
    templateType = CLI;
    published = false;
    timestamp = 2015-07-14 16:07:52;
    imports = ;
##
##template variables
    integer vlan_id;
##
##template content
    vlan $$vlan_id$$
##

Derived Template:
##template properties
    name =a vlan extended;
    userDefined= true;
    supportedPlatforms = All;
    templateType = CLI;
    published = false;
    timestamp = 2015-07-14 16:07:52;
    imports = a vlan base,template2;
##
##template variables
    interface vlanInterface;
##
##template content
    <substitute a vlan base>
    interface $$vlanInterface$$
    <substitute a vlan base>
##
```

When you launch the extended template, the parameter inputs for the base template are also obtained. In addition, the substituted content is used for complete CLI command generation.

Report Template

The template type of REPORT template is python, and it has two subtypes, UPGRADE and GENERIC.

UPGRADE

The UPGRADE template is used for pre-ISSU and post-ISSU scenarios. These templates are listed in the ISSU wizard.

Refer to the default upgrade template packaged in Nexus Dashboard for more information on pre-

ISSU and post-ISSU handling. The default upgrade template is issu_vpc_check.



In order to execute any ISSU operations, any new Nexus Dashboard user must first set the necessary device credentials under the Credential Management page. You will not be able to execute ISSU operations without first setting the proper device credentials.

GENERIC

The GENERIC template is used for any generic reporting scenarios, such as, collecting information about resources, switch inventory, SFPs, and NVE VNI counters. You can also use this template to generate troubleshooting reports.

Resources Report

This report displays information about resource usage for a specific fabric.

The **Summary** section shows all resource pools with the current usage percentages. Use the horizontal scroll bar at the bottom of the window to display more columns.

POOL NAME: Specifies the name of the pool.

POOL RANGE: Specifies the IP address range of the pool.

SUBNET MASK: Specifies the subnet mask.

MAX ENTRIES: Specifies the maximum number of entries that can be allocated from the pool.

USAGE INSIDE RANGE: Specifies the current number of entries allocated inside the pool range.

USAGE OUTSIDE RANGE: Specifies the current number of entries set outside the pool range.

USAGE PERCENTAGE: This is calculated by using the formula: (Usage Inside Range/Max Entries) *100.

Click **View Details** to display a view of resources allocated or set in each resource pool. For example, the detailed section for a SUBNET has information about the resources that have been allocated within the subnet.

Switch Inventory Report

This report provides a summary about the switch inventory.

Click **View Details** to display more information about the modules and licenses.

SFP Report

This report provides information about utilization of SFPs at a fabric and device level.



The switch inventory and SFP reports are supported only on Cisco Nexus devices.

Troubleshooting Reports

These reports are generated to help in troubleshooting scenarios. Currently, the **NVE VNI Counters**

report is the only pre-defined troubleshooting report. Generating **NVE VNI Counters** reports involves performing periodic checks to identify the VNIs that are among the top hits based on network traffic. In a large-scale setup, we recommend limiting the report generation frequency to a minimum of 60 minutes.

NVE VNI Counters Report

This report collects the show nve vni counters command output for each VNI in the fabric.

After comparing the oldest report and the newest report, the **Summary** section shows the top-10 hit VNIs. The top hit VNIs are displayed in these categories:

- L2 or L3 VNIs for unicast traffic
- L2 or L3 VNIs for multicast traffic
- L2 only VNIs for unicast traffic
- L2 only VNIs for multicast traffic
- L3 only VNIs for unicast traffic
- L3 only VNIs for multicast traffic

The oldest report refers to the first report that is saved in the current reporting task. If you want to select a specific report as the first report against which the current report has to be compared, delete all reports that are older than the one selected so that the selected report becomes the first and oldest report.

For example, three reports were run yesterday at 8:00 a.m, 4:00 p.m. and 11:00 p.m. If you want to use the report at 11:00 p.m. as the first and oldest report for today's reporting, delete the two reports that were run yesterday at 8:00 a.m. and 4:00 p.m.

For a periodic report, the oldest report is the first report that is run at the start time of a period. For daily and weekly reports, the current report is compared against the previously generated report.

The **Summary** section displays a column-wise report with information about the total transmitted bytes and the VNIs. Use the horizontal scroll bar at the bottom of the window to display more columns.



The **Summary** section in the NVE VNI Counters report displays negative numbers in the TOTAL TX BYTES column if a report is generated after a switch reload or after clearing the counters on the switch. The numbers are displayed correctly in the subsequent reports. As a workaround, we recommend deleting all old reports or creating a new job before reloading switches or clearing counters. Click **View Details** to display more information. This section shows NVE VNIs and counters on a per-switch basis.

For more information on how the reports are displayed, refer *Programmable Reports* chapter.

Template Usage

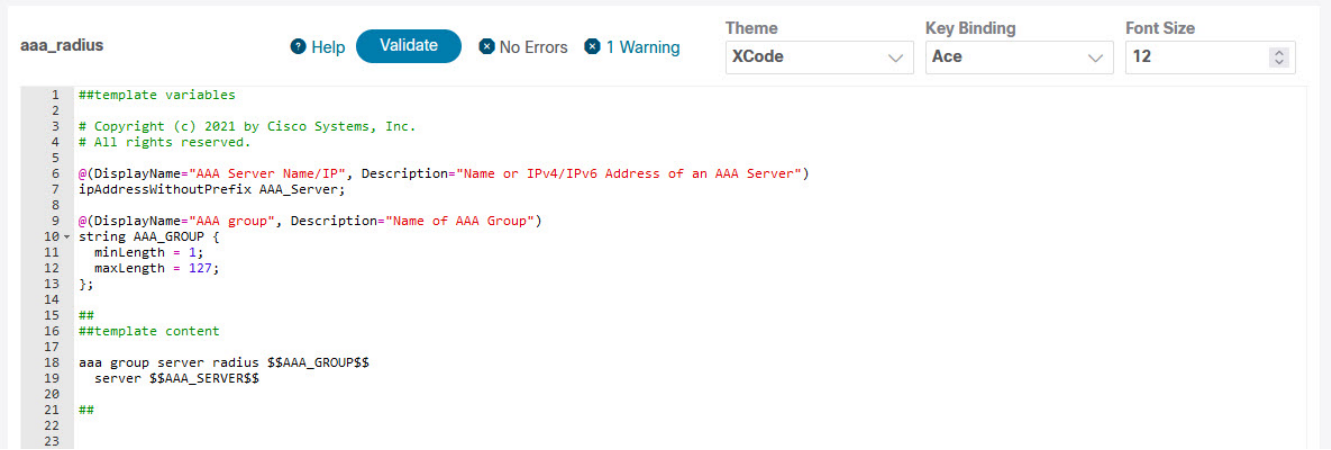
template Type	Specifies the type of Template used.	• POLICY • SHOW • PROFILE • FABRIC • REPORT • INTERNAL • EXEC
template content type	Specifies the type of content in the template.	• CLI • PYTHON • PYTHON3 • PYTHON3_CLI • PYTHON_CLI • TEXT

Policy Template

For the policy template, there are these template content types:

- TEMPLATE_CLI
- PYTHON
- PYTHON_CLI

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:



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:

ext_int_routed_host_11_1 | Help | Validate | No Errors | No Warnings | Theme: XCode | Key Binding: Ace | Font Size: 12

```
##template variables

#    Copyright (c) 2019-2022 by Cisco Systems, Inc.
#    All rights reserved.
@(IsInternal=true)
string SERIAL_NUMBER;

@(PrimaryAssociation=true, IsInternal=true)
interface INTF_NAME;

@(IsMandatory=false, DisplayName="Interface IP", Description="IP address of the interface", ReadOnly=true)
ipV4Address IP;

@(IsMandatory="IP!=null", DisplayName="IP Netmask Length", Description="IP netmask length used with the IP address (Min:1, Max:31)", ReadOnly=true)
integer PREFIX {
  min = 1;
  max = 31;
};

@(IsMandatory=false, DisplayName="Interface IPv6", Description="IPv6 address of the interface", ReadOnly=true)
ipV6Address IPv6;

@(IsMandatory="IPv6!=null", DisplayName="IPv6 Netmask Length", Description="IPv6 netmask length used with the IPv6 address (Min:1, Max:128)", ReadOnly=true)
integer PREFIXv6 {
  min = 1;
  max = 128;
};

@(IsMandatory=false, DisplayName="Interface VRF", Description="Interface VRF name, default VRF if not specified", ReadOnly=true)
string INTF_VRF {
  minLength = 1;
  maxLength = 32;
};

@(IsMandatory=false, DisplayName="Routing TAG", Description="Routing tag associated with interface IP", ReadOnly=true)
string ROUTING_TAG;

@(DisplayName="MTU", IsMTU=true, Description="MTU for the interface", ReadOnly=true)
integer MTU {
min = 576;
max = 9216;
defaultValue=9216;
};

@(DisplayName="SPEED", Description="Interface Speed", ReadOnly=true)
enum SPEED {
validValues=Auto,100Mb,1Gb,2.5Gb,5Gb,10Gb,25Gb,40Gb,50Gb,100Gb,200Gb,400Gb;
defaultValue=Auto;
};

@(IsMandatory=false, DisplayName="Interface Description", Description="Add description to the interface", ReadOnly=true)
string DESC {
  minLength = 1;
  maxLength = 254;
};

@(IsMandatory=false, IsMultiLineString=true, DisplayName="Freeform Config", Description="Additional CLI for the interface", ReadOnly=true)
string CONF;

@(DisplayName="Enable Interface", Description="Uncheck to disable the interface", ReadOnly=true)
boolean ADMIN_STATE {
defaultValue=true;
};

@(IsInternal=true)
string SOURCE;

##
##template content

from com.cisco.dcbu.vinci.rest.services.jython import PTIWrapper
from com.cisco.dcbu.vinci.rest.services.jython import Wrapper
from com.cisco.dcbu.vinci.rest.services.jython import WrappersResp
from utility import *

def add():
    try:

        respObj = WrappersResp.getRespObj()
        try:
            adminState = ADMIN_STATE
        except:
            adminState = "true"
            pass
        try:
            source = SOURCE
        except:
            source = INTF_NAME
            pass
        Wrapper.print("ext_int_routed_host_11_1_add : Source sn = %s, "
                      "source interface= %s: source: %s"
                      % (SERIAL_NUMBER, INTF_NAME, source))

        routingTag = ""
        try:
            if ROUTING_TAG != "":
                routingTag = ROUTING_TAG
        except:
            pass

        #Only valid operation is shut/no-shut from interface page
        #In addition, this can only happen if someone does a save on the interface edit for an interface attached to this policy
        #After that shut/no-shut from interface manager starts sending source = INTF instead of source = LINK-UUID of VRF_LITE IFC
        #This is a bug that needs to be fixed but right now putting a workaround here
        if source == INTF_NAME:
```

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.

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.

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 Nexus Dashboard fabrics.

It should not be necessary for you to create your own fabric templates. Nexus Dashboard provides a few fabric templates out of the box, such as Data Center VXLAN EVPN, External Connectivity Network, VXLAN EVPN Multi-Site, BGP Fabric, and so on.

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, Nexus Dashboard 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*')

For more information about how to apply overlay configuration via the Networks & VRFs workflow in Nexus Dashboard, see the *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 the *Viewing and Editing Policies* section.

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 Override** option below.

1. Navigate to **Admin > System Settings > General**.
2. In the **Advanced settings** area, click **Edit**, then put a check in the check box in the **Display advanced server settings options for TAC support** field to enable that option.
3. Choose **Fabric Management > Advanced settings**.
4. Under the **LAN-Fabric** tab, verify that the **Template in-use override** field is enabled.

   Put a check in the check box in the **Template In-Use Override** field to enable that option if it's not enabled already.

5. Click **Save**.
6. Edit the desired template(s).
7. Go to Fabrics Overview and click **Recalculate and deploy**.

   This will regenerate PTIs and the updated content will be picked up and used for the expected configuration (or intent).

8. After all the intents/policies are re-generated and deployed, enable/check the **Template In-Use Override** checkbox to avoid unnecessary delays in the recalculate and deploy operation.



Keeping this checkbox in the disabled/unchecked state triggers the regeneration of all policy instances, which can significantly increase the time required for the Recalculate & Deploy operation.