了解Catalyst Center上的模板

简介

本文档介绍Cisco Catalyst Center以及三层或折叠核心园区架构配置模板的经验。

背景信息

本文档面向对Cisco Catalyst Center有基本了解，并且拥有配置模板经验的企业专业人员。对于已经或计划使用三层或折叠核心园区架构的人来说，这一点尤其重要。

主要目标是帮助读者使用Cisco Catalyst Center中的模板实施并自动化配置和管理解决方案。通过介绍先进的见解、实用技术和实际示例，本文档为那些希望通过自动化和基于模板的管理来增强其LAN基础架构技能并优化工作流程的人提供了实用资源。

执行摘要

随着企业网络的不断发展，对可扩展、一致和自动化管理的需求空前高涨。Cisco Catalyst Center提供基于意图的集中式平台，可简化整个园区网络的配置、调配和保证。本白皮书探讨网络专业人员如何利用Cisco Catalyst Center的CLI模板编辑器和自动化功能来简化网络运营、减少配置错误，以及加快跨三层核心架构和折叠核心架构的部署。其中详细介绍了以下方面的最佳实践：设计基于Jinja2的模块化模板，将自动化集成到第0天和第N天的工作流程中，以及跨核心、分布层和接入层实现操作一致性。通过采用本文档中介绍的策略，您可以将传统手动网络管理转变为与思科的基于意图的网络愿景相一致的灵活、标准化和自动化驱动的模型。

园区网络的挑战

随着园区网络不断发展以满足现代组织的需求，它们面临着以下几个主要挑战：

2a.网络管理的复杂性

许多网络功能仍然需要手动管理，这增加了人为错误的风险。这不仅会增加维护工作，还会使IT资源紧张，尤其是预算静态或有限时。

2b。部署和自动化挑战

有线和无线网络的新设备自注册通常既耗时又复杂，导致部署延迟和管理开销增加。

2c。软件映像管理

在整个网络中保持一致的“黄金映像”具有挑战性。许多网络最终会为有线和无线设备配备多个操作系统，导致效率低下和管理困难。

2d。网络配置不一致

网络配置的变化可能导致合规性问题和运营效率低下，使维护可靠安全的网络更加困难。
2e。用户期望不断提高

用户需要无间断的连接和无缝的应用体验，无论其位置或设备如何。满足这些期望要求网络具有恢复力、智能性并能适应实时变化。

除了这些挑战外，现代LAN基础设施还面临着其他各种复杂性。

使用Cisco Catalyst Center简化园区网络

Cisco Catalyst Center是适用于园区网络的集中式网络管理解决方案，支持总部、分支机构、有线和无线连接以及IT/OT环境。它提供灵活的部署选项，包括物理设备、VMware ESXi服务器或AWS云。Catalyst Center具有全面的功能，可简化操作、提高性能并增强安全性。

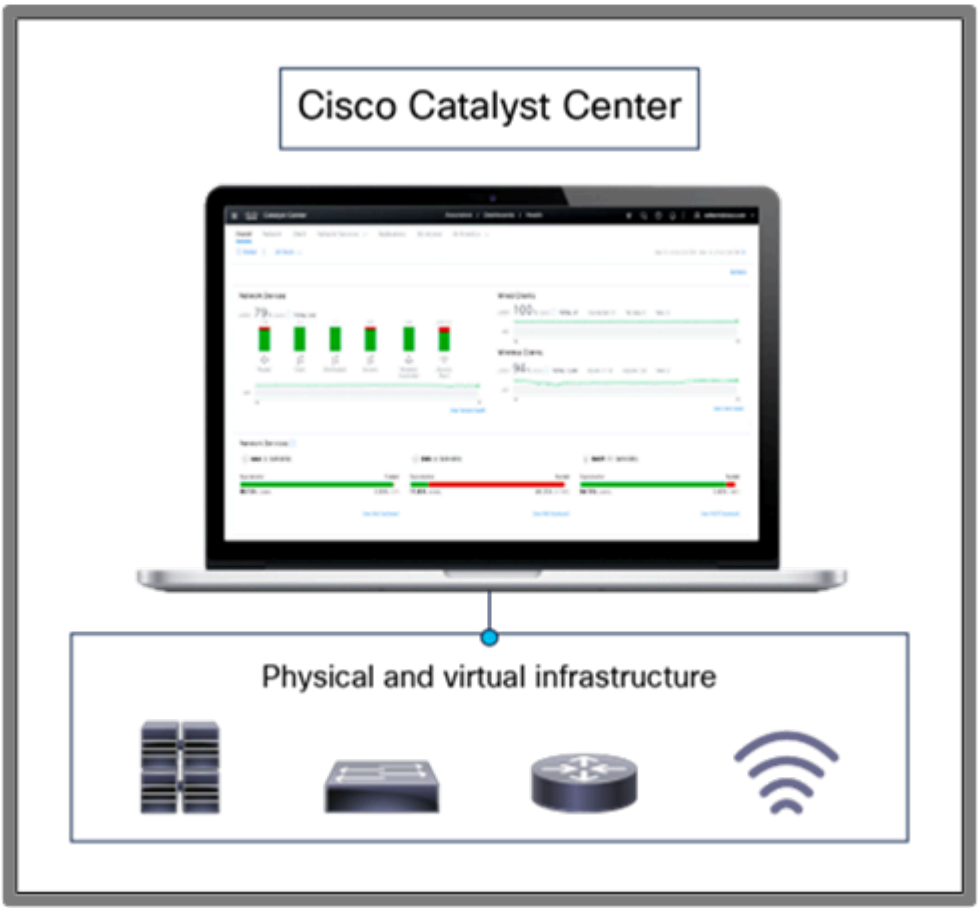


图 1：使用Cisco Catalyst Center管理基础设施

关键功能和优点

Cisco Catalyst Center(CC)提供简化网络管理和自动化的高级功能：

零接触调配(ZTP):自动注册设备，减少手动工作和部署时间。

软件映像管理(SWIM):通过升级前和升级后检查确保设备间软件版本一致，以防出现问题。

基于意图的自动化:通过将网络意图转换为有线和无线网络的设备配置来简化部署。

LAN自动化:自动执行第3层IP编址和路由，以创建端到端拓扑。

无线网络自动化:即插即用(PnP)等功能可实现无线接入点的快速调配。

分层网络管理:允许站点特定的配置文件（例如，SSID、RF参数、VLAN）跨位置进行一致的部署。

CLI模板：Catalyst Center模板编辑器使管理员可以轻松创建和管理基于CLI的配置模板，从而

跨设备实现一致且高效的部署。

保证:保证允许通过CC对受管设备进行集中监控。

除这些功能外，Cisco Catalyst Center还提供许多其它不属于本文档范围的功能。本文重点介绍如何使用Catalyst Center设计CLI模板。

采用Catalyst Center的LAN园区架构高级概述

传统LAN园区网络构成了企业连接的主干，确保有线和无线设备可靠且可扩展的通信。这些网络通常使用3层架构或折叠核心架构设计，具体取决于组织的规模和复杂性。

三层架构

三层架构是一种基础网络设计模型，由核心层、分布层和接入层组成。此架构提供可扩展性、高性能和高效的流量管理。请参阅每一层的概述。

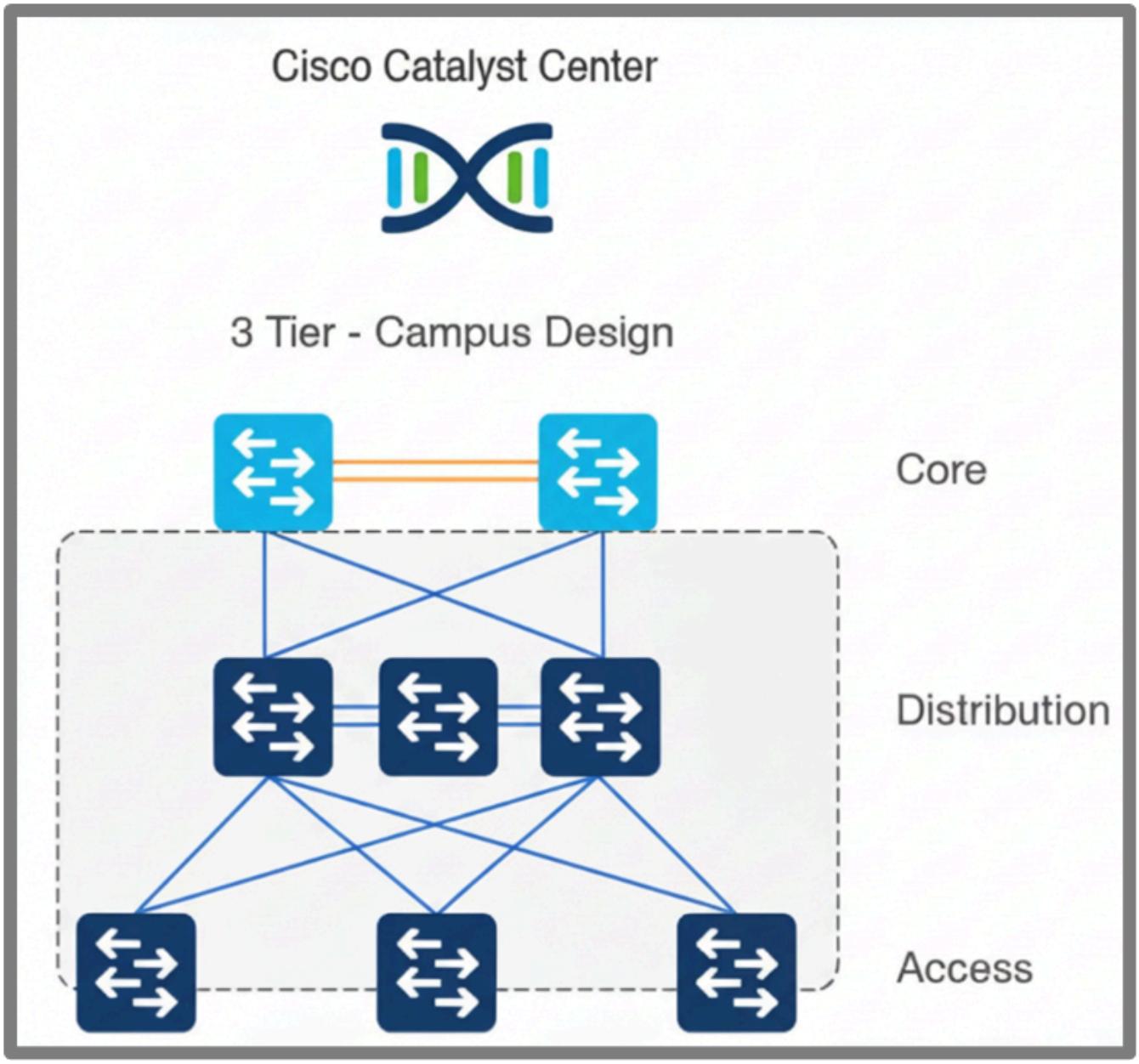


图 2：三层园区架构

核心层

核心层充当网络的主干，提供高速连接和可扩展性。关键配置包括北向和南向路由协议（例如OSPF和BGP）、路由策略、下行链路和上行链路接口配置、安全强化等

分布层

分布层桥接核心层和接入层，处理流量聚合、策略实施和冗余。关键配置包括用于冗余的HSRP/VRRP、用于环路预防的STP、第2层和第3层VLAN、上行链路和下行链路接口配置、用于安全的ACL以及安全强化。

接入层

接入层将终端连接到网络，从而实现安全可靠的访问。关键配置包括接入接口配置、上行链路接口配置、第2层VLAN、用于限制对设备访问的ACL以及安全强化。

折叠核心架构

折叠核心架构将核心层和分布层整合到单一层，在保持性能和可扩展性的同时降低了复杂性和成本。此方法非常适合不需要单独核心层的中小型网络。请参阅此架构中各层的概述。

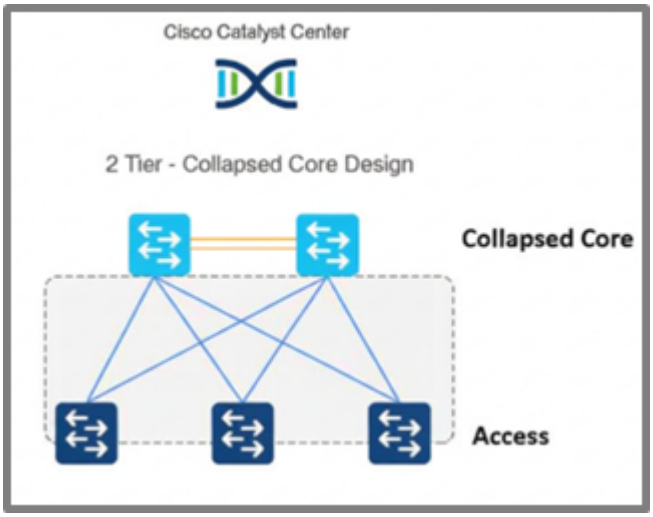


图 3：紧缩核心园区架构

折叠核心层

折叠核心层将核心层和分布层的功能相结合，提供主干连接、流量聚合和策略实施。关键配置包括北向和南向路由协议（如OSPF和BGP）、路由策略、下行链路和上行链路接口配置、用于故障检测的BFD、使用SVI的VLAN间路由、用于网关冗余的HSRP/VRRP、用于环路预防的STP以及安全强化。利用Cisco Catalyst Center中的模板，可以自动执行这些配置，从而确保一致且高效的部署。

接入层

如前所述，接入层将终端连接到网络，从而实现安全可靠的接入。关键配置包括接入接口配置、上行链路接口配置、第2层VLAN、用于限制对设备访问的ACL以及安全强化。

模板设计注意事项

本节概述如何在Cisco Catalyst Center中设计模板以生成设备配置。模板编辑器可创建可重复使用的CLI模板，并支持动态部署为您的网络定制的配置，从而简化调配。Catalyst Center支持两种模板语言：Jinja2和Velocity。这些语言有助于设备的配置管理。

Jinja是一种流行的、设计者友好的模板化语言，主要与Python配合使用，用于生成动态内容，如HTML、XML或其他基于文本的格式。它允许在模板内嵌入变量和控制结构（如循环和条件）以创建动态输出。

Apache Velocity是一个基于Java的模板引擎，它使用Velocity模板语言(VTL)在各种文档（包括网页、XML甚至源代码）中启用动态内容。它将Java对象的数据与模板合并，以生成最终输出。

本文档仅涉及Jinja2模板。

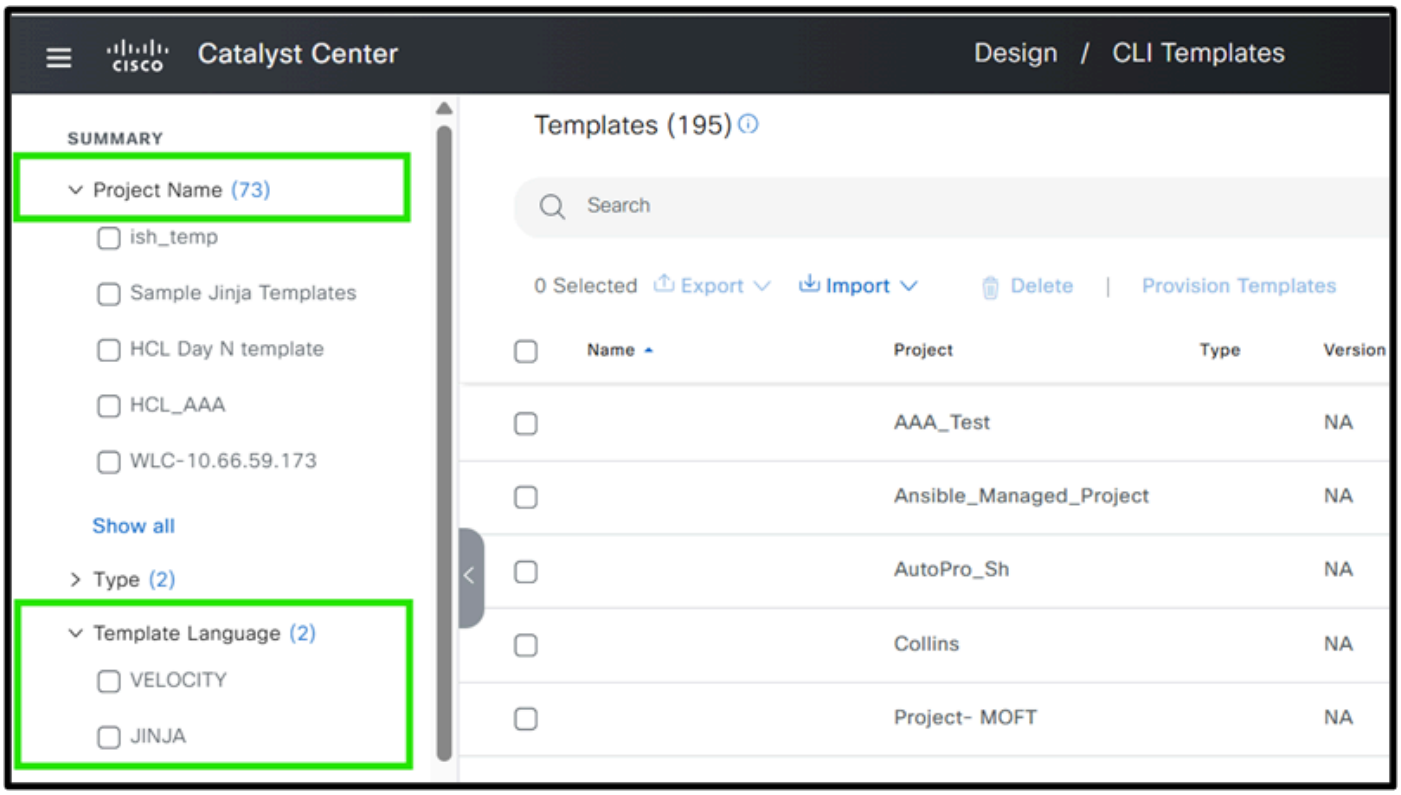


图 4：Cisco Catalyst Center模板编辑器

在本文档中，我们使用Jinja2是因为它具有灵活性。重点不是对Jinja2进行深入的探索，而是对模板设计的实际应用进行研究。有关Catalyst Center中Jinja2模板化的详细信息，请参阅以下链接：

https://ciscolearning.github.io/cisco-learning-codelabs/posts/cat-center-j2-part-1/#0

在深入了解思科园区网络的模板设计策略之前，必须利用关键的最佳实践，以确保使用模板时的效率和可管理性。

模板结构和最佳实践/最佳策略指南

使用Cisco Catalyst Center自动配置网络设备时，必须采用结构化策略和最佳实践。这些步骤有助于确保网络基础设施的一致性、可扩展性和易管理性。

按设备角色划分配置

首先根据设备在网络拓扑中的角色对设备进行分类。常见角色包括：

核心

分布

接入

示例：与接入交换机相比，作为核心交换机的设备必须具有不同的配置要求。

将配置划分为模块化模块

在每个设备角色中，通过将相似功能或配置组合在一起，将配置拆分为多个模块块。此模块化方法可简化自动化、故障排除和未来更新。

核心设备示例：

OSPF配置块

BGP配置块

QoS策略块

确定与角色无关的配置块

某些配置块适用于所有设备角色。识别和标准化这些模块可确保整个网络的最佳实践和一致性。

常见的角色无关配置块：

基本配置：主机名、登录标语

管理协议:DHCP、DNS、NTP、SNMP

访问策略：标准安全配置

这些模块可重新用于核心、分布层和接入设备，从而简化自动化流程。

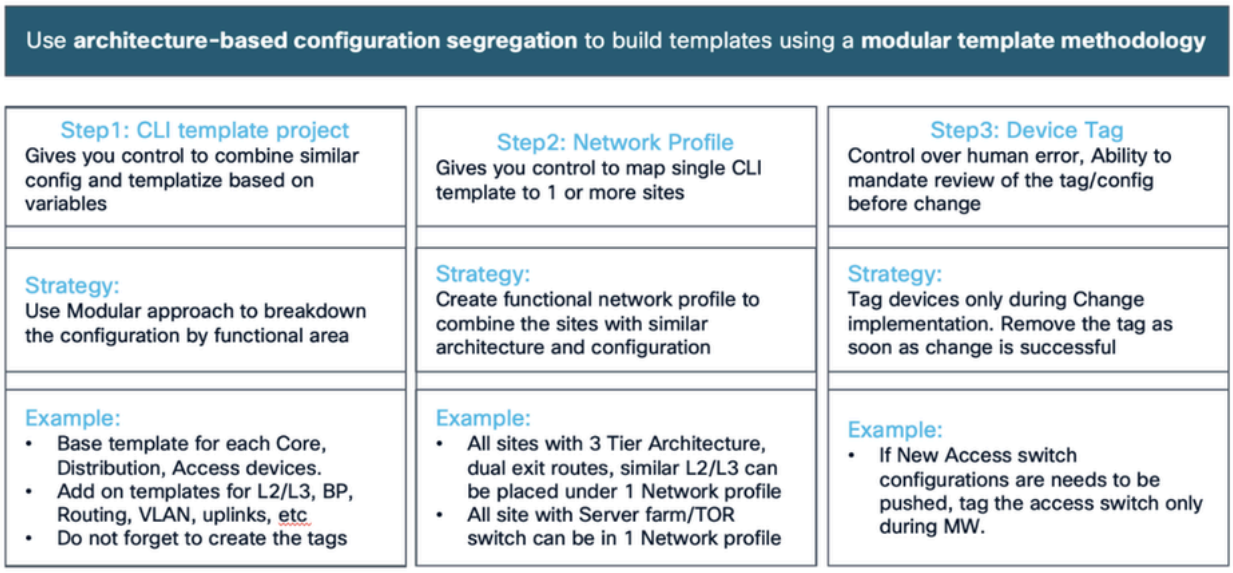


图 1：最佳实践与示例

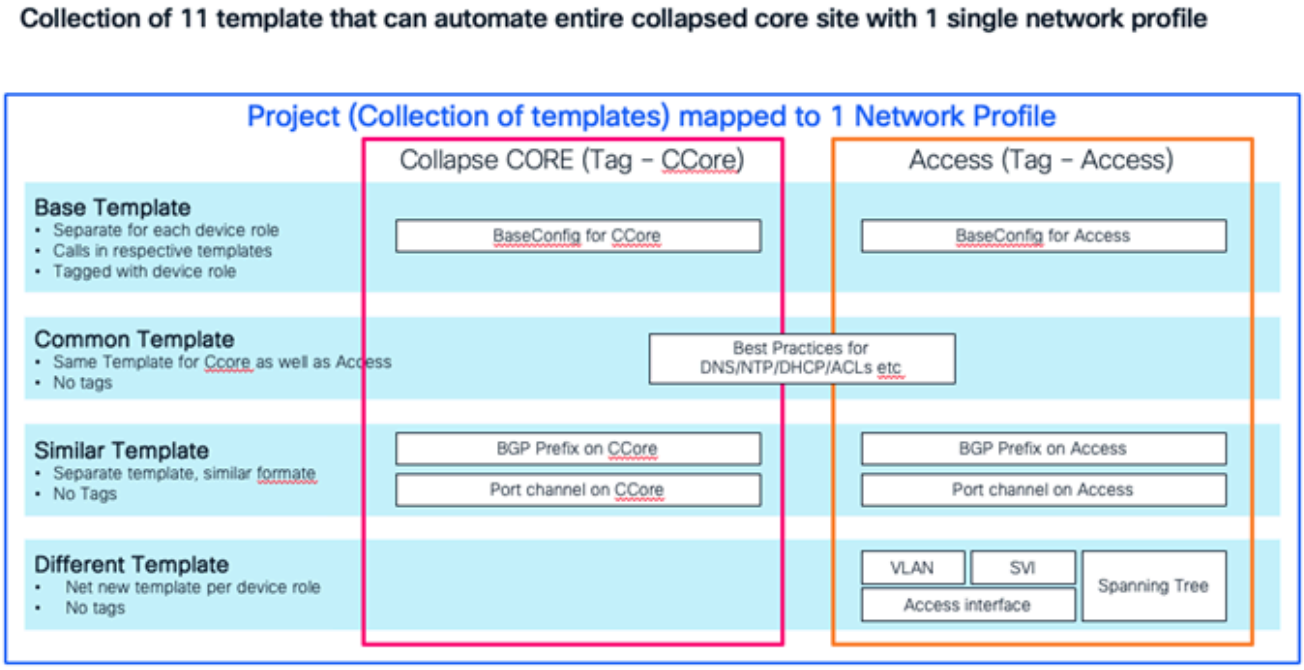


图 2：折叠核心模板示例

使用模板的最佳实践

用于自动配置的模块化模板设计

在Cisco Catalyst Center中自动化设备配置时，请避免将所有配置嵌入到单个整体模板中。相反，应采用模块化方法：

创建引用较小、特定用途的模板（模块）的基本模板:

将配置分为逻辑模块（例如，接口设置、路由协议、安全功能）。

此结构使更新更有效 — 对特定模块的更改会在使用该模块的任何位置自动反映出来，从而显着减少错误和复杂性。

示例：分支机构设备的模块化配置

假设您要自动配置分支设备。

基本模板：

包括对关键配置区域的模块模板的引用。

根据需要将变量传递到每个模块以进行自定义。

模块模板：

interface_settings:管理接口配置。

routing_protocols:包含OSPF、EIGRP或BGP设置。

security_features:定义ACL、防火墙规则或其他安全策略。

```
{% include "Branch/Interface Configuration" %}
{% include "Branch/Routing Protocol Configuration" %}
{% include "Branch/Security Configuration" %}

{{ Branch_Interface_Configuration(branch_id) }}
{{ Branch_Routing_Protocol_Configuration(branch_id, ospf_area) }}
{{ Branch_Security_Configuration(branch_id) }}
```

基本模板结构示例：

使用此结构，对路由或安全配置的任何更改只需在各自的模块中进行，无论使用基本模板的地方，这些更改都会立即得到反映。这使您的配置在所有分支路由器上更易于管理且更一致。

此处项目名称为Branch，并在project下定义了3个其他模块。所有这些都组合在基本模板中。

最小化模板中的变量

将模板中的变量数保持为最少，以降低复杂性和错误。更少的变量可简化部署，尤其是在大型网络中，从而使流程更高效且一致。

对模板使用设备标签

利用Cisco Catalyst Center中的设备标签（如位置、角色或站点）创建动态且可扩展的Jinja2模板。这些标记启用条件逻辑，确保将正确的配置应用到适当的设备。此方法可最大限度地减少错误并简化不同网络环境中的模板管理。

尽可能对静态值进行硬编码

对静态值进行硬编码可以简化模板并提高部署效率。常见示例包括DNS、NTP或系统日志服务器的IP地址，这些地址通常在设备之间保持一致。同样，在接入交换机上使用标准VLAN ID可以对这些值进行硬编码，从而降低可变性并加快部署。

采用两阶段方法：第0天和第N天模板

使用思科即插即用等服务自注册设备时，请使用两阶段模板策略：

第0天模板：推送基本配置，确保设备可以与Cisco Catalyst Center通信。

N日模板：一旦设备可访问，就部署高级功能和配置。

最佳实践支持高效且可扩展的模板，可简化思科园区网络部署。

Jinja模板宏中的空白控件

使用Jinja语言创建模板时，必须小心处理空白和新行，特别是在宏中呈现动态内容时。累积的空白或无意中的新行可能会导致生成的输出出现格式问题，这必定会导致下游处理出现误解或错误。为了解决这个问题，Jinja提供了控制空白的语法：将减号(-)直接放在分隔符（{{- ... -}}或

{%- ... -%}）中，它会删除表达式周围的任何前导或尾随空格。例如，将{{item[1]}}替换为{{-item[1] -}}可确保宏渲染时删除任何多余的空格或新行。这种做法在遍历列表或生成配置文件时特别有用，如模板代码片断所示。我们建议始终在此类场景中应用空白控制，以保持干净和可预测的输出。

示例（建议使用）：

```
wildcard_list %}中的项为{%
   {% if item[0] == prefix -%}
      {{ — 项目[1] -}}
   {%- endif %}
{% — 结束于%}
```

三层架构

本白皮书从开发接入交换机到核心交换机的模板开始，概述了每一层的要求。

接入层交换机

接入交换机使用即插即用登录，并且必须使用0天模板。有关Catalyst Center中的即插即用过程的详细信息，请参阅链接：

https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/catalyst-center/2-3-7/user_guide/b_cisco_catalyst_center_user_guide_237/m_onboard-and-provision-devices-with-plug-and-play.html

如前所述，Catalyst Center支持Velocity和Jinja2模板语言。本文档利用Jinja2来说明模板结构，因为它具有灵活性。接入层交换机配置可以使用Day-0和Day-N模板部署。

基本的第0天模板可以构建，请参阅第1步：

步骤 1：定义模板

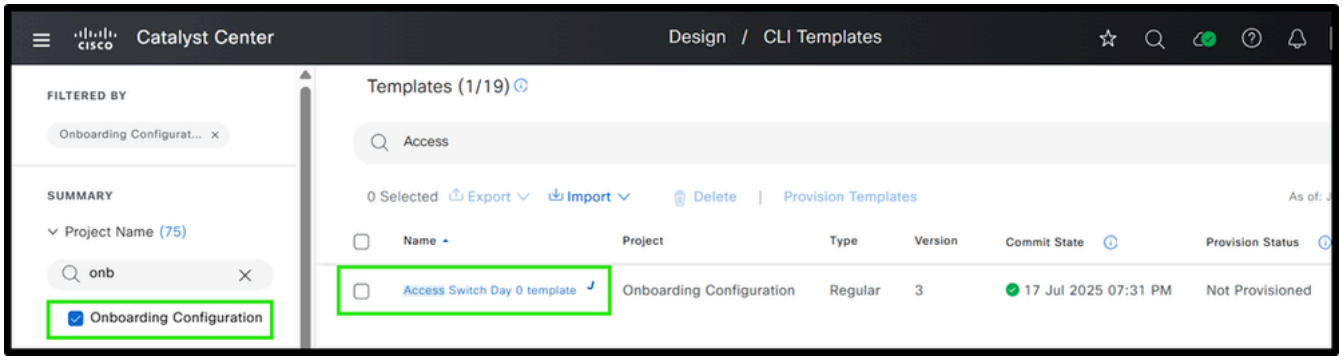
```
username admin privilege 15 password SamplePass123
!
enable secret EnableSecret123
!
ip routing
!
vlan {{ branch_number * 100 + 13 }}
 name SW_MGMT
!
interface vlan {{ branch_number * 100 + 13 }}
 ip address {{ ip_address }} 255.255.255.128
 no ip redirects
 no ip unreachables
 no ip proxy-arp
!
ip route 0.0.0.0 0.0.0.0 {{ nexthop }} name Default-Gateway
!
interface range Te1/1/1 - 2
 switchport
 switchport mode trunk
 no shutdown
!
```

步骤 1：定义模板

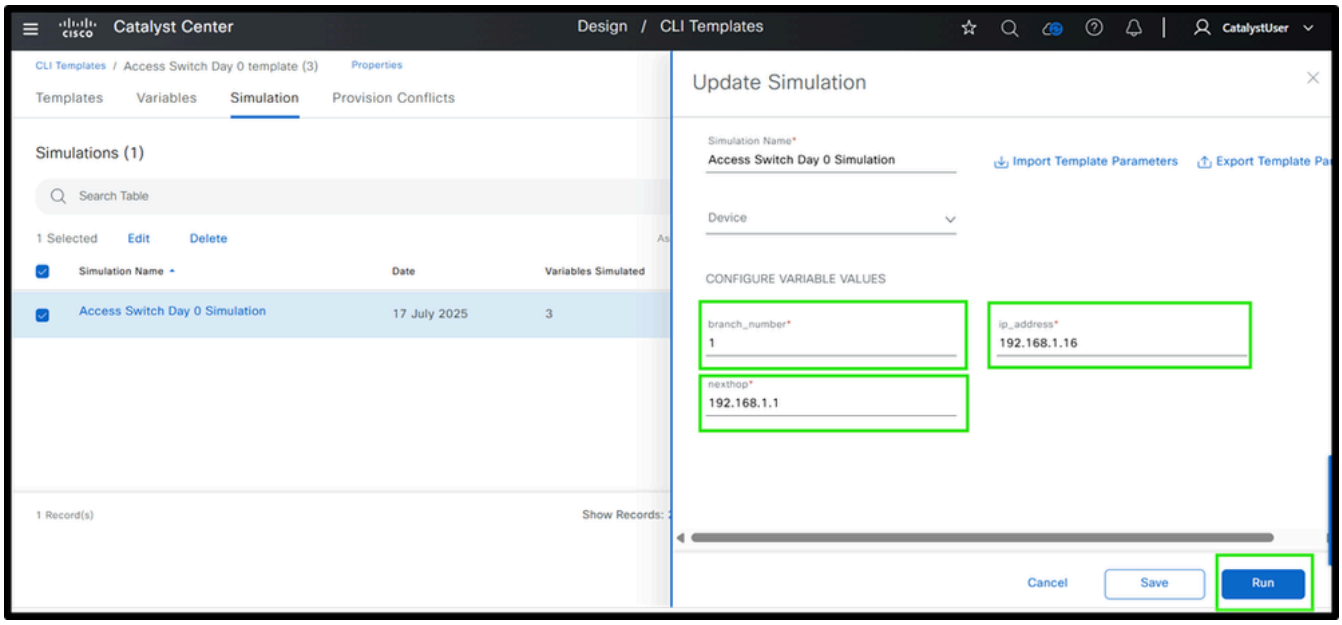
模板通过硬编码常量（如用户名、密码、使能加密和子网掩码）简化了配置，因为分支中的所有交换机共享相同的管理VLAN子网掩码。但是，管理IP地址对于每个交换机都是唯一的，并且被定义为变量。第N天模板中必须提供全面的模板结构，该模板利用此Day 0模板。在第N天模板中，接入交换机的每个功能都由专用模块管理，例如，一个模块处理第2层VLAN，独立的模块管理上行链路和下行链路接入接口，另一个模块侧重于安全强化，等等。

虽然首选一致的VLAN ID，但可以使用基于分支编号的公式动态生成不同的ID(例如，分支1 = VLAN 113，分支2 = VLAN 213)。 这使模板可在分支之间重复使用。同样，下一跳IP是一个变量，因为它必须因所连接的分布集群的不同而各分支机构有所不同。

步骤 2：执行模拟并提供变量



具有模拟输入和输出的接入交换机Day 0模板结构



例如模拟输入

始终建议在部署之前模拟模板。屏幕截图显示了输入变量后的最终配置。

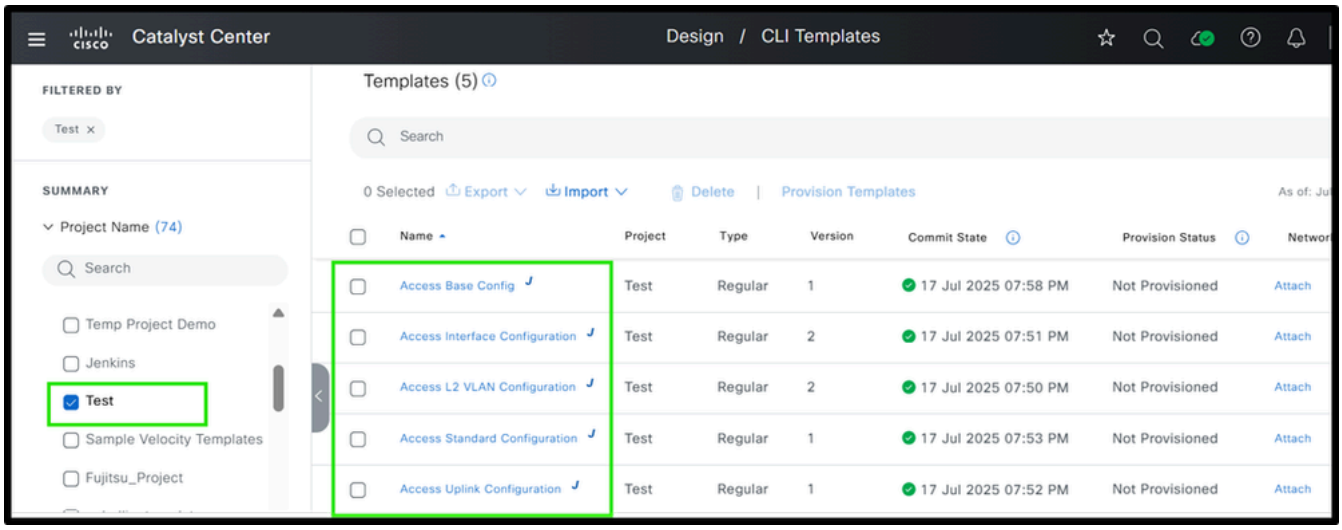
输入值后的内部配置

现在，我们来看看如何创建模块化N日模板。

接入交换机配置可以分为多个模块，所有这些模块都可以组合在一个基本模块中。接入交换机的基本模板结构如图所示。

基本模板及其模块均在Cisco Catalyst Center中名为“测试”的项目中创建。

步骤 1：定义各种模板（包括基本模板）



接入交换机第N天模板结构

步骤 2：定义各种模块

访问基本配置：

屏幕截图显示了基本配置的示例。

```
{% include "Test/Access L2 VLAN Configuration" %}
{% include "Test/Access Interface Configuration" %}
{% include "Test/Access Uplink Configuration" %}
{% include "Test/Access Standard Configuration" %}

{{ Access_L2_VLAN_Configuration(branch_number, is_poe) }}
{{ Access_Uplink_Configuration(branch_number, is_poe)}}
{{ Access_Interface_Configuration(branch_number,  is_poe) }}
{{ Access_Standard_Configuration(branch_number) }}
```

访问基本配置

此模块化配置模板包括四个部分：VLAN配置、上行链路接口配置、接入接口配置和标准配置。它仅使用两个变量：branch_number和is_poe，使其易于管理。

branch_number计算特定于分支机构的VLAN ID（如Day 0模板所示），is_poe确定接入交换机是PoE还是非PoE交换机。这些变量在调配过程中提供，并传递给模块以创建正确的配置，从而减少工作量，提高效率。

现在，让我们检查每个模块，了解它们对生成整体配置的特定部分的贡献。

访问L2 VLAN配置

```
{% macro Access_L2_VLAN_Configuration (branch_number, is_poe) %}
!
vlan {{ 100 * branch_number + 11 }}
 name DATA_VLAN
!
vlan {{ 100 * branch_number + 12 }}
 name VOICE_VLAN
!
{% if is_poe == 'Yes' %}
vlan {{ 100 * branch_number + 14 }}
 name AP_Mgmt
{% endif %}
!
{% endmacro %}
```

访问L2 VLAN配置

如前所述，本模块根据分支编号创建VLAN。数据和语音VLAN在所有交换机上创建，无论它们是否支持PoE。仅当将is_poe设置为“Yes”（表示交换机支持PoE）时，才会创建AP管理VLAN（例如，分支1的AP管理VLAN 114）。如果is_poe为“No”，则会跳过AP管理VLAN，因为非PoE交换机不支持接入点。这使用if条件进行管理。

```
{% macro common_access_settings() %}
 switchport port-security maximum 2
 switchport port-security
 switchport port-security violation shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
 storm-control broadcast level 2.00
 storm-control multicast level 2.00
 storm-control unknown-unicast 2.00
{% endmacro %}

{% macro Access_Interface_Configuration(branch_number, is_poe) %}
!
interface range Gi1/0/1 - 6
{% if is_poe == 'Yes' %}
 description *** AP ***
 switchport mode access
 switchport access vlan {{ 100 * branch_number + 14 }}
{% else %}
 description *** User Ports ***
 switchport mode access
 switchport access vlan {{ 100 * branch_number + 11 }}
 switchport voice vlan {{ 100 * branch_number + 12 }}
{% endif %}
{{ common_access_settings() }}
!
interface range Gi1/0/7 - 24
 description *** User Ports ***
 switchport mode access
 switchport access vlan {{ 100 * branch_number + 11 }}
 switchport voice vlan {{ 100 * branch_number + 12 }}
{{ common_access_settings() }}
!
{% endmacro %}
```

访问接口配置

此模块处理接入接口配置并使用与前面所述的PoE交换机相同的方法。如果is_poe变量为Yes，表示交换机是PoE交换机，则必须为前六个端口(1-6)配置AP管理VLAN。否则，前六个端口必须设置为用户接入端口。

假设交换机是24端口型号，则无论交换机是否为PoE，其余端口(7-24)始终配置为用户接入端口。

接口范围已标准化，不再作为输入变量，这被视为将模板中变量数量降至最低的最佳实践。此外，该模块包含一个名为common_access_settings的宏，该宏通过合并重复配置来最小化模板大小。此宏在接口设置内直接调用，无需多次指定。

注意：此模板对所有访问接口应用相同的说明。如果每个接口都需要唯一的描述，建议使

用单独的Python脚本或类似的自动化工具推送描述。

查看生成上行链路接口配置的模块。

```
{% macro Access_Uplink_Configuration(branch_number, is_poe) %}
{% if is_poe == 'Yes' %}
!
interface range Te 1/1/1 - 2
 switchport
 switchport mode trunk
 switchport trunk allowed vlan {{ branch_number * 100 + 11 }},{{ branch_number * 100 + 12 }},{{ branch_number * 100 + 13 }},{{
branch_number * 100 + 14 }}
 no shutdown
!
{% else %}
!
interface range Te 1/1/1 - 2
 switchport
 switchport mode trunk
 switchport trunk allowed vlan {{ branch_number * 100 + 11 }},{{ branch_number * 100 + 12 }},{{ branch_number * 100 + 13 }}
 no shutdown
!
{% endif %}
{% endmacro %}
```

接入上行链路配置

此模块生成上行链路接口的配置并处理VLAN修剪。如果交换机支持PoE，则AP管理VLAN会包含在允许的VLAN列表中；否则，它将被排除。如前所述，此逻辑使用代码中的if条件来管理。

查看最终模块，该模块演示标准配置，包括最佳实践和安全强化。

⚠ 警告：请注意，这仅用于说明目的，不能用作实际网络设置的参考，因为配置可能会因特定要求而异

```
{% macro Access_Standard_Configuration (branch_number) %}
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vtp mode off
no errdisable recovery cause all
crypto key generate rsa modulus 2048
!
ip ssh version 2
ip ssh time-out 120
ip ssh source-interface vlan {{ branch_number * 100 + 13 }}
no ip http server
no ip http secure-server
ip http client source-interface vlan {{ branch_number * 100 + 13 }}
!
logging buffered informational
logging host 192.168.1.10
logging host 192.168.2.20
logging source-interface vlan {{ branch_number * 100 + 13 }}
!
ntp authentication
ntp authentication-key 10 md5 Netw0rkAuthKey
ntp source vlan {{ branch_number * 100 + 13 }}
ntp server 192.168.3.1 key 10
ntp server 192.168.3.2 key 10
!
snmp-server enable traps
snmp-server trap-source vlan {{ branch_number * 100 + 13 }}
snmp-server group NMSNWDEVICE v3 priv access SNMPHOST
snmp-server user netadmin NMSNWDEVICE v3 auth sha AuthKey123 priv aes 128 PrivKey123
!
ip access-list standard SNMPHOST
permit 192.168.4.0 0.0.0.255
!
ip access-list standard VTYACL
permit 192.168.5.10
```

第一部分：访问标准配置

```
permit 192.168.5.11
!
aaa new-model
ip tacacs source-interface vlan {{ branch_number * 100 + 13 }}
tacacs server TACACS_1
 address ipv4 192.168.6.1
 key TACACSKey123
 timeout 4
tacacs server TACACS_2
 address ipv4 192.168.6.2
 key TACACSKey123
 timeout 4
aaa group server tacacs+ TACACS-SERVER
 server name TACACS_1
 server name TACACS_2
!
aaa authentication login default group TACACS-SERVER local
aaa authorization exec default group TACACS-SERVER local
aaa accounting exec default start-stop group TACACS-SERVER
!
line console 0
login authentication default
exec-timeout 5 0
!
line vty 0 15
login authentication default
access-class VTYACL in
exec-timeout 5 0
!
banner login ^
********************************** WARNING ************************************
All systems/network should be used/accessed by authorized persons only
  If you are not authorized to do so, you should log off immediately
    Access to and usage of this system /network may be monitored
    All users must comply with information security policies
        Any Violation may lead to disciplinary action.
*******************************************************************************^
{% endmacro %}
```

第 2 部分：访问标准配置

此模块生成一个标准配置，该配置融入了最佳实践、安全加固和安全设备管理的主要功能。除branch_number外，大多数值都经过硬编码，以确保各分支机构的一致性。branch_number用于计算每个分支机构中交换机的管理VLAN，并且用作多个配置的源接口。

步骤 3：在配置交换机之前执行模拟。仅必须模拟基本配置。

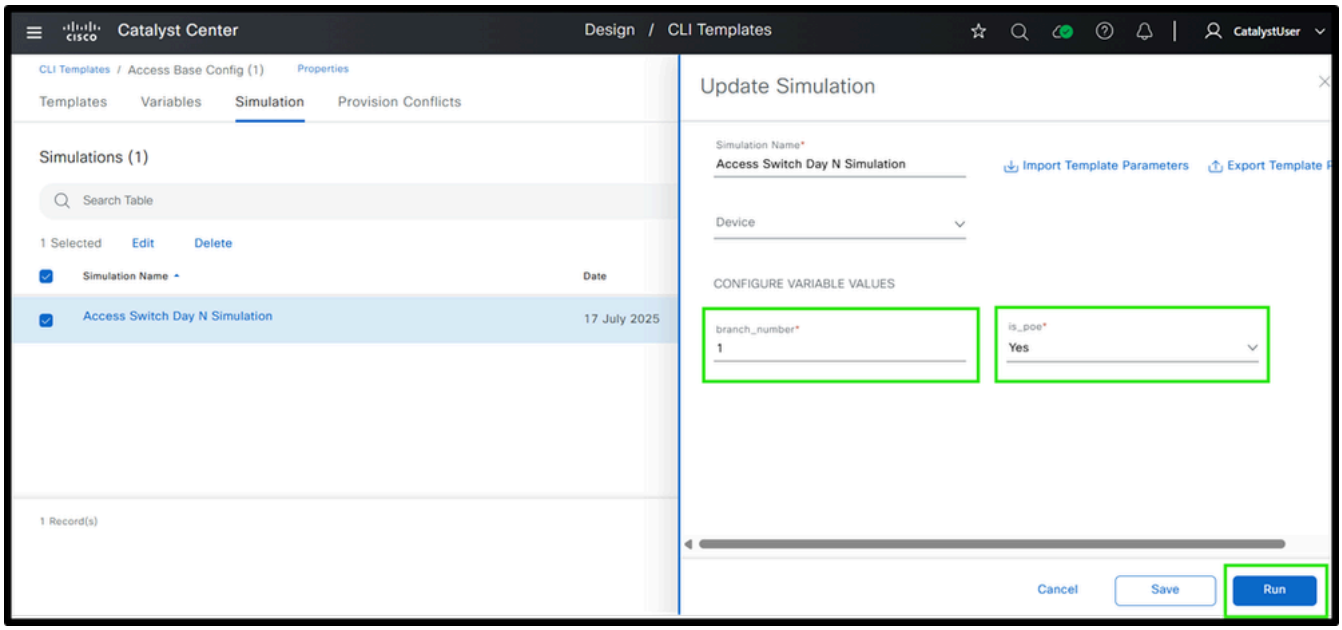
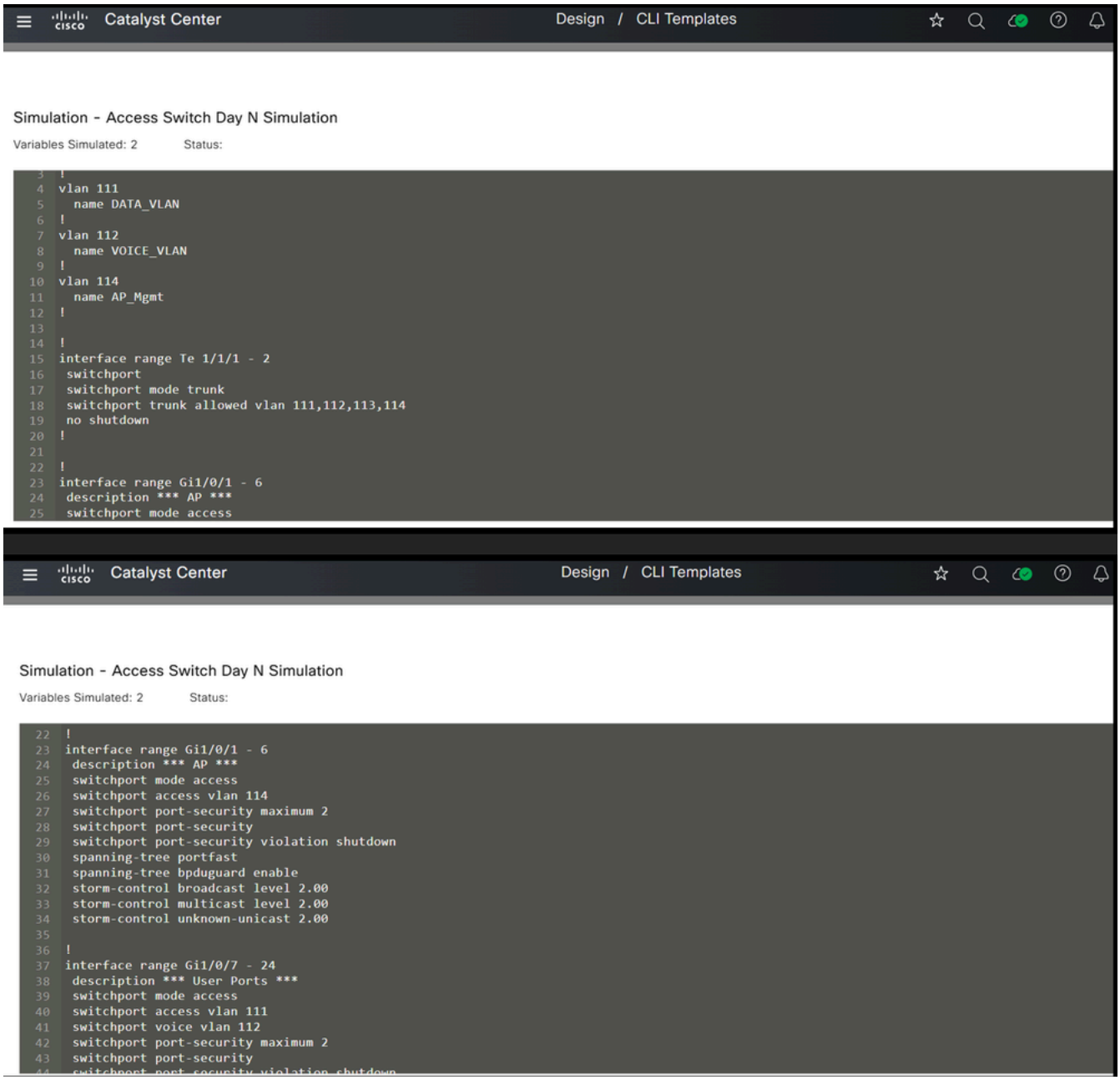


图 7：接入交换机第N天模板模拟输入和输出



模拟

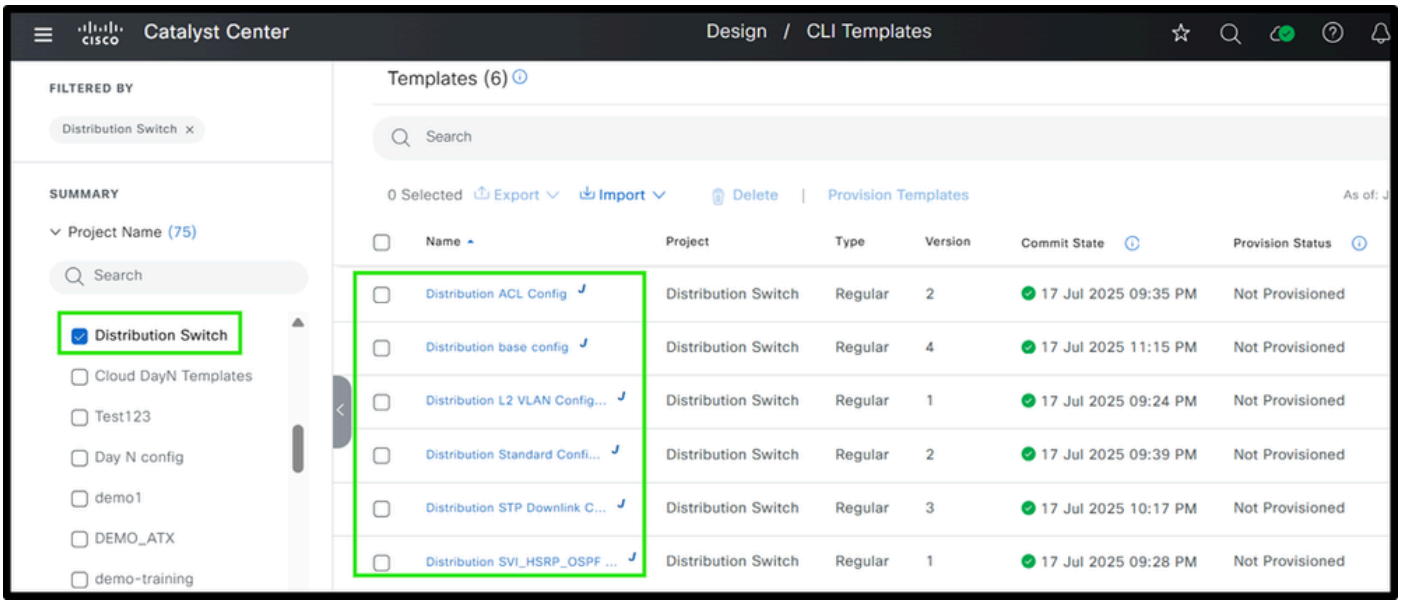
这就是在接入层使用模板生成配置的方法。

现在，让我们来看看分布层设备，看看如何对它们应用模板。

分布层交换机

现在设计一个用于分布交换机的模块化模板。基本模板及其模块是Cisco Catalyst Center中“分布交换机”项目的一部分。

步骤 1： 分布层交换机模板结构



例如分发模板

第2步： 定义每个模块

提供的基本配置定义了每个模块，并引用了所有模块。

```
{% include "Distribution Switch/Distribution L2 VLAN Configuration" %}
{% include "Distribution Switch/Distribution STP Downlink Config" %}
{% include "Distribution Switch/Distribution SVI_HSRP_OSPF Config" %}
{% include "Distribution Switch/Distribution ACL Config" %}
{% include "Distribution Switch/Distribution Standard Configuration" %}

{{ Distribution_L2_VLAN_Configuration(branch_number, is_poe) }}}}
{{ Distribution_STP_Downlink_Config(branch_number, is_poe, distribution_number) }}
{{ Distribution_SVI_HSRP_OSPF_Config(branch_number, is_poe, distribution_number) }}
{{ Distribution_ACL_Config(branch_number) }}
{{ Distribution_Standard_Config() }}
```

例如分布库模板模块

与接入交换机类似，所有模板都创建在“分布交换机”项目中，并在基本模板中引用。虽然有些模板与用于接入交换机的模板相同，但本部分解释特定于分布交换机的差异。模块“分布层L2 VLAN配置”与前面介绍的接入交换机模块相同。请检查提供此信息的Access L2 VLAN Configuration模块。它根据为变量提供的输入值生成所需的VLAN。

现在请查看“分布STP下行链路配置”模块，该模块处理分布交换机的生成树和上行链路配置的生成。

```
{% macro Distribution_STP_Downlink_Config (branch_number, is_poe, distribution_number) %}
!
spanning-tree mode rapid-pvst

{% set base_vlan = branch_number * 100 %}
{% set vlans = [base_vlan + 11, base_vlan + 12, base_vlan + 13] %}

{% if is_poe == 'Yes' %}
  {% set vlans = vlans + [base_vlan + 14] %}
{% endif %}

{% if distribution_number == 1 %}
   {% set stp_priority = 4096 %}
{% else %}
   {% set stp_priority = 8192 %}
{% endif %}

spanning-tree vlan {{ vlans | join(',') }} priority {{ stp_priority }}
!
interface range TWE 1/0/1 - 2
 switchport
 switchport mode trunk
 switchport trunk allowed vlan {{ vlans | join(',') }}
 no shutdown
!
{% endmacro %}
```

分布STP下行链路配置

此处使用Jinja2宏功能，在基于模块中引用。因此，这种结构有助于构建模块化方法。

此模块根据branch_number和交换机是否启用PoE来配置生成树协议(STP)和下行链路接口。“branch_number”变量用于为每个分支机构生成唯一的基本VLAN，从而确保不同的VLAN，类似于已为接入交换机突出显示的方法。如果交换机支持PoE("is_poe" == 'Yes')，则会向列表中添加一个额外的VLAN，例如AP管理VLAN。“distribution_number”变量确定STP优先级，为分布1设置4096（使其成为首选根网桥），为辅助分布交换机设置8192。最后，将适当的VLAN应用到中继接口，确保根据交换机是否启用PoE，仅允许相关VLAN。

现在请查看分布SVI_HSRP_OSPF配置模块，该模块重点介绍SVI、HSRP和OSPF的设置，以实现高效的网络路由和冗余。

```
{% macro Distribution_SVI_HSRP_OSPF_Config (branch_number, is_poe, distribution_number) %}
!
interface loopback0
ip address {{ loopback_ip }} 255.255.255.255
!
router ospf 1
router-id {{ loopback_ip }}
!
key chain HSRP_KEY
 key 0
  key-string cisco@7875
!
interface vlan {{ 100 * branch_number + 11 }}
description Data_Endpoints
ip address 172.17.{{ (branch_number - 1) * 16 }}.{{ distribution_number + 1 }} 255.255.240.0
standby bfd
standby version 2
standby {{ 100 * branch_number + 11 }} ip 172.17.{{ (branch_number - 1) * 16 }}.1
{% if distribution_number == 1 %}
standby {{ 100 * branch_number + 11 }} priority 255
{% else %}
standby {{ 100 * branch_number + 11 }} priority 250
{% endif %}
standby {{ 100 * branch_number + 11 }} authentication md5 key-chain HSRP_KEY
standby {{ 100 * branch_number + 11 }} preempt delay minimum 120
no ip redirects
no ip unreachables
no ip proxy-arp
ip ospf 1 area 0
bfd interval 100 min_rx 100 multiplier 3
!
! uplink interfaces
interface TWE1/1/1
 no switchport
 ip address {{ twe1_1_1_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
interface TWE1/1/2
 no switchport
 ip address {{ twe1_1_2_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
{% endmacro %}
```

分布SVI_HSRP_OSPF配置

此模块Distribution_SVI_HSRP_OSPF_Config可帮助配置分布交换机的SVI、HSRP、OSPF和上行链路接口。在本例中，我们重点介绍用于数据子网的SVI，但此方法可用于其他SVI，如语音或管理。

如果已完成数据子网的IP地址规划，则可以根据branch_number和distribution_number变量为每个SVI自动计算IP地址。例如，如果Branch 1的子网为172.17.0.0/20,Branch 2的子网为172.17.16.0/20，而Branch 3的子网为172.17.32.0/20，则网关IP为172.17.x.1（其中x是分支机构的编号）。 第一台分布层交换机的第二个IP地址是172.17.x.2，第二台分布层交换机的第三个IP地址是172.17.x.3。这样可以自动计算IP地址，从而减少错误并简化过程。

为环回接口分配了变量loopback_ip的IP，该变量用作OSPF路由器ID，以确保网络中路由稳定一致。在OSPF配置中，此环回IP用作路由器ID，相关接口将添加到OSPF区域0。对于HSRP，优先级值设置：第一台分布交换机为255，第二台为250，确保正确的故障转移。此外，使用密钥链(HSRP_KEY)配置HSRP身份验证以增强安全性。

为了保持配置干净且可管理，需要对某些值进行硬编码。例如，所有分支的子网掩码(255.255.240.0)和某些HSRP设置（如版本和BFD）相同，从而减少了变量数量。这使得配置更简单、更易于应用，并且出错的可能性更小。最后，上行链路接口配置了IP并添加到OSPF区域0，以便在交换机之间正确路由。此方法使配置过程更易于管理且不易出错，同时对于不同的分支机构也是灵活的。

现在复习“分布ACL配置”模块，该模块提供分布层分段。

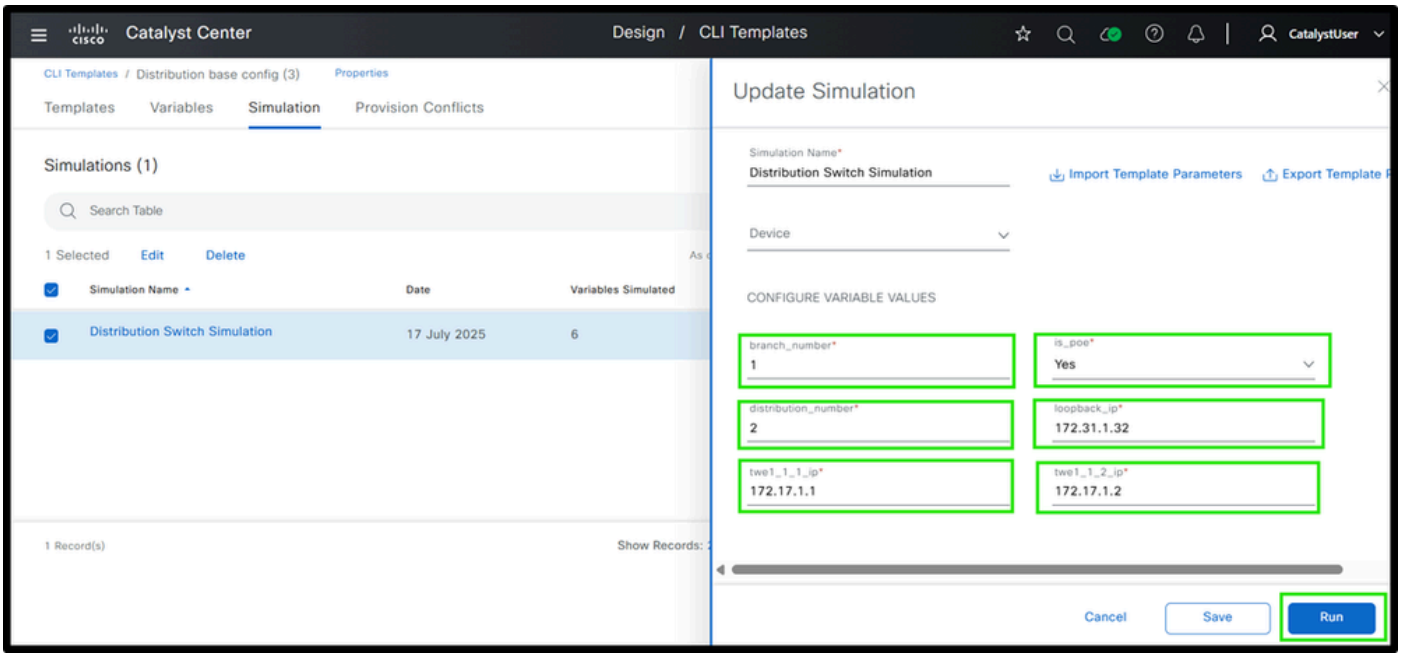
```
{% macro Distribution_ACL_Config (branch_number) %}
!
ip access-list extended BLOCK_BRANCH
 deny ip 172.17.{{ 16 * (branch_number - 1) }}.0 0.0.15.255 172.16.{{ 16 * (branch_number - 1) }}.0 0.0.15.255
 deny ip any host 239.255.255.250
 permit ip any any
!
interface vlan {{ 100 * branch_number + 11 }}
 ip access-group BLOCK_BRANCH in
!
{% endmacro %}
```

分布ACL配置

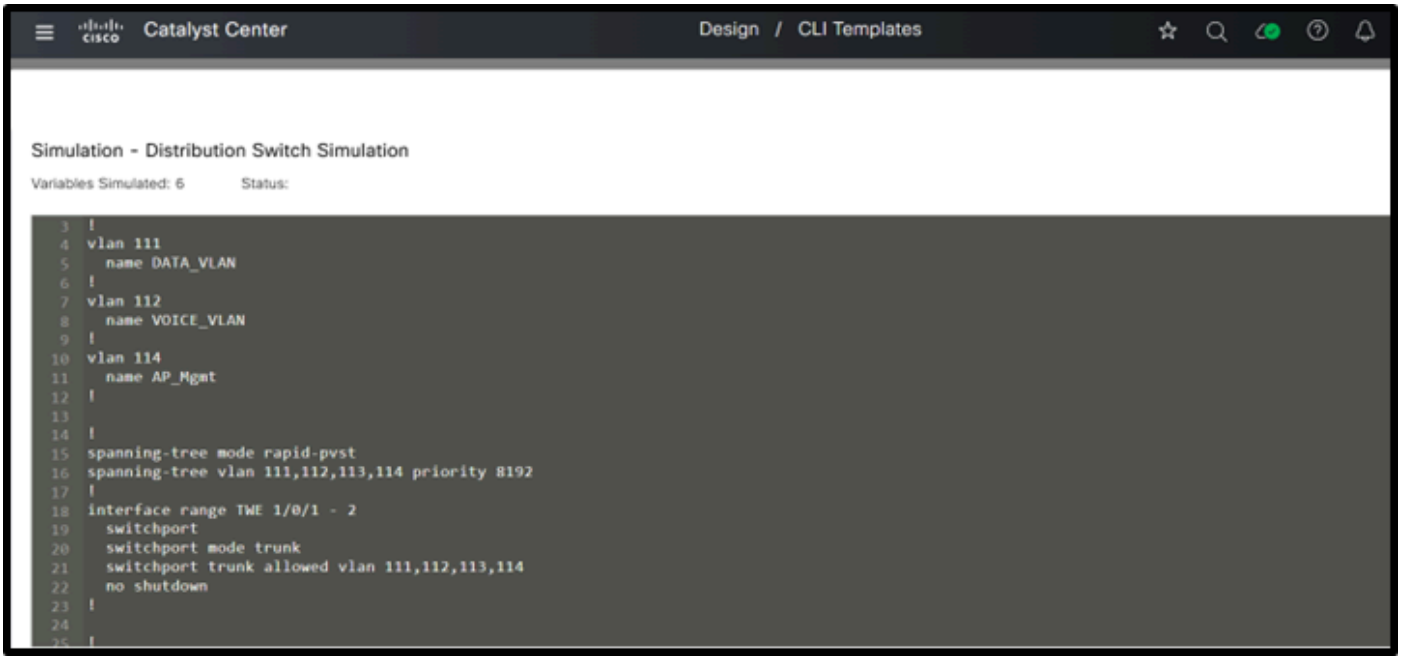
本单元演示使用Jinja2模板在分布层进行分段。它利用branch_number变量动态计算子网地址，从而实现自动化且可扩展的ACL配置。对于每个分支，ACL通过拒绝这些范围之间的IP流量来阻止子网1(172.17.X.0)和子网2(172.16.X.0)之间的通信。它还拒绝流向组播地址239.255.255.250的流量，同时允许所有其他流量。VLAN接口根据分支编号动态分配，ACL应用于该接口的入站流量。这种自动化方法可确保每个分支机构的有效分段，减少手动配置错误，并简化网络策略实施。

最后，最后一个模块“分布标准配置”与访问标准配置模块中描述的几乎相同（有关详细信息，请参阅该部分）。 它包括最佳实践、安全加固和安全设备管理的主要功能。唯一的区别在于源接口:在接入交换机模板中，它被定义为VLAN {{ branch_number * 100 + 13 }}，而在分布交换机配置中，它可以硬编码为Loopback0。

步骤 3：在部署配置之前执行模拟。



(1)配电开关模版模拟输入和输出



(2)配电开关模版模拟输入与输出

(3)配电开关模版模拟输入和输出

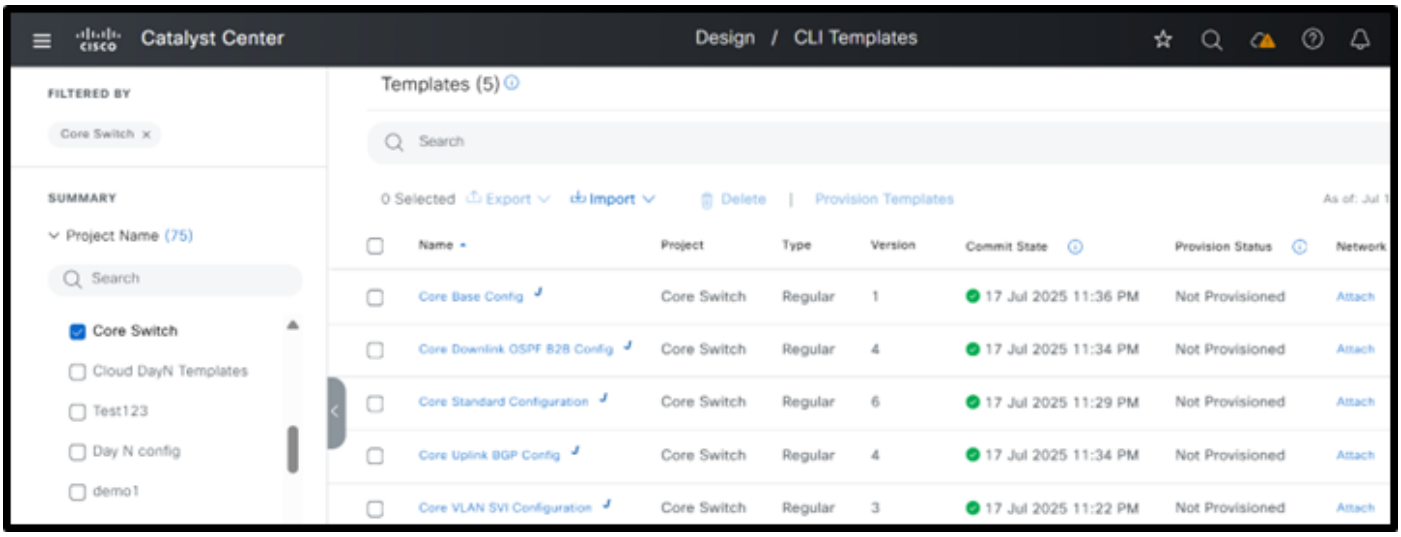
(4)配电开关模版模拟输入和输出

这是模板在分布层用于生成配置的方式。现在，让我们来看看核心层设备，看看如何在这里应用模板。

核心层交换机

现在，请为核心交换机设计一个模块化模板。基础模板及其模块是Cisco Catalyst Center“核心交换机”项目的一部分。请参阅步骤1中的基本模板。

步骤 1：定义各种核心交换机结构



核心交换机模板结构

步骤 2：定义各种模块

```
{% include "Core Switch/Core VLAN SVI Configuration" %}
{% include "Core Switch/Core Downlink OSPF B2B Config" %}
{% include "Core Switch/Core Uplink BGP Config" %}
{% include "Core Switch/Core Standard Configuration" %}

{{ Core_VLAN_SVI_Configuration () }}
{{ Core_Downlink_OSPF_B2B_Config () }}
{{ Core_Uplink_BGP_Config () }}
{{ Core_Standard_Config () }}
```

核心基本配置

所有分支中的大多数核心交换机配置都类似，因此可以对通用值进行硬编码。通常，只有IP地址会更改，这些地址可以使用变量进行设置。由于每个分支机构通常只有两台核心交换机，因此对这些变量的管理非常简单。即使某些分支机构拥有更多的核心交换机，其数量仍少于接入或分布交换机数量。因此，最佳做法是将接入和分布交换机的变量降至最低更为重要，因为它们数量较多，而且变量过多会增加配置时间。

现在从第一个模块开始："核心VLAN SVI配置。" 在本示例中，核心交换机位于防火墙后面，必须与其建立eBGP对等。此模块负责生成eBGP对等和OSPF邻居关系所需的VLAN和相应的SVI。假设防火墙在主用/备用配置中运行。

```
{% macro Core_VLAN_SVI_Configuration () %}
!
vlan 2001
 name eBGP_peering_to_FW
!
vlan 2002
 name OSPF_neighborship
!
interface vlan 2001
 description eBGP Peering to Firewall
 ip address {{ VLAN2001_IP }} 255.255.255.248
 bfd interval 100 min_rx 100 multiplier 3
 no ip redirects
 no ip unreachables
 no ip proxy-arp
!
interface vlan 2002
 description OSPF neighborship to Core SW 2
 ip address {{ VLAN2002_IP }} 255.255.255.248
 bfd interval 100 min_rx 100 multiplier 3
 ip ospf 1 a 0
 no ip redirects
 no ip unreachables
 no ip proxy-arp
!
{% endmacro %}
```

核心VLAN SVI配置

如前所述，此模块创建建立OSPF和BGP邻居关系所需的VLAN和关联的SVI。除SVI IP地址以外的所有参数均采用硬编码 — 如果与IP编址计划一致，则包括子网掩码。此方法有助于限制变量并减少配置错误的可能性。

现在，我们来了解一下“核心下行链路OSPF B2B配置”模块，该模块为核心交换机1和核心交换机2之间的下行链路接口、OSPF和背对背链路生成配置。

```
{% macro Core_Downlink_OSPF_B2B_Config () %}
!
interface loopback0
ip address {{ loopback_ip }} 255.255.255.255
!
router ospf 1
router-id {{ loopback_ip }}
default-information originate
!
! downlink interfaces
interface TWE1/0/1
 ip address {{ twe1_0_1_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
interface TWE1/0/2
 ip address {{ twe1_0_2_ip }} 255.255.255.0
 ip ospf 1 area 0
 no shutdown
!
interface TWE1/0/24
 description Towards_Core_SW
 switchport mode trunk
 switchport trunk allowed vlan 2001,2002
 channel-group 10 mode active
 spanning-tree portfast trunk
 no shutdown
!
interface TWE1/0/48
 description Towards_Core_SW
 switchport mode trunk
 switchport trunk allowed vlan 2001,2002
 channel-group 10 mode active
 spanning-tree portfast trunk
 no shutdown
!
interface Port-channel10
 description Towards_Core_SW
 switchport mode trunk
 switchport trunk allowed vlan 2001,2002
 spanning-tree portfast trunk
 no shutdown
!
{% endmacro %}
```

核心下行链路OSPF B2B配置

与上一个模块类似，此模块中的大多数值都经过硬编码，以尽量减少变量的数量。只有环回接口和下行链路接口的IP地址是可变的。此外，背靠背端口通道和VLAN在不同分支机构之间实现标准化。

现在，我们来了解一下“核心上行链路BGP配置”模块，它生成BGP配置并管理连接到防火墙的上行链路。

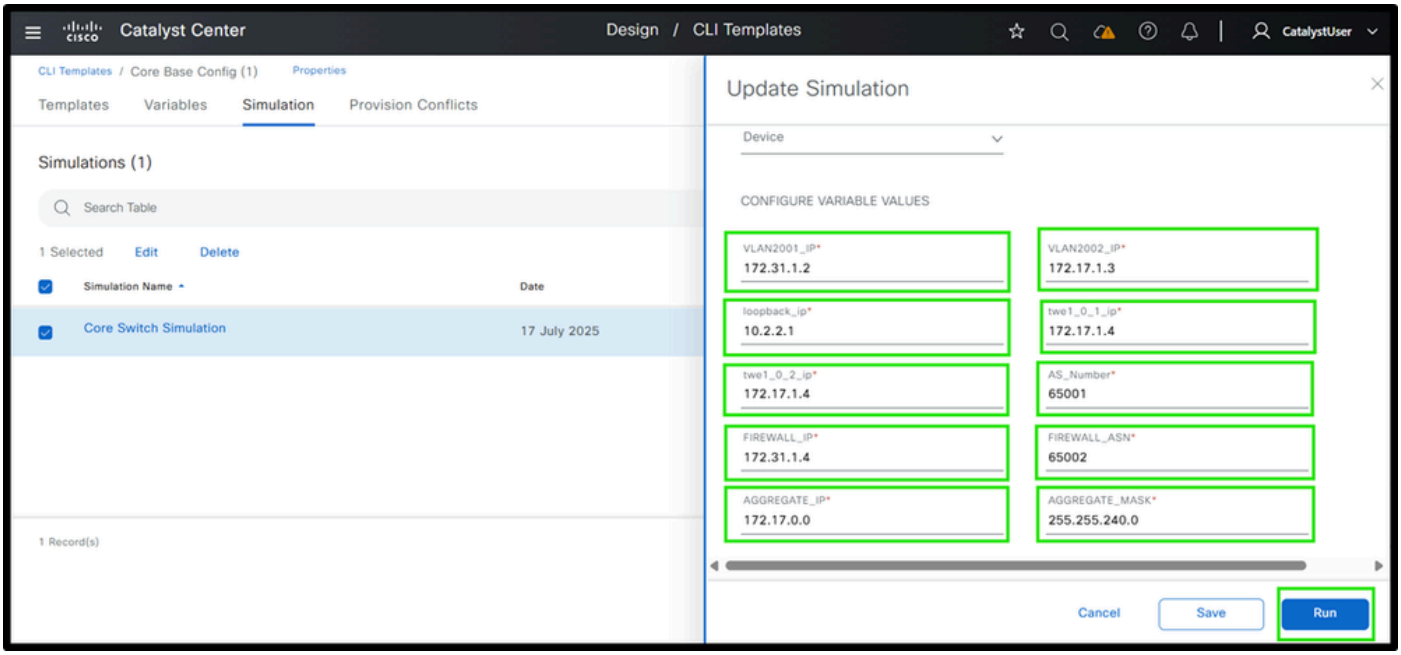
```
{% macro Core_Uplink_BGP_Config () %}
!
router bgp {{ AS_Number }}
bgp log-neighbor-changes
bgp router-id interface Loopback0
bgp graceful-restart
!
! eBGP Peering with Firewall
neighbor {{ BGP_NEIGHBOR }} remote-as {{ REMOTE_AS }}
neighbor {{ BGP_NEIGHBOR }} description eBGP Peering with Firewall
neighbor {{ BGP_NEIGHBOR }} update-source vlan 2001
neighbor {{ BGP_NEIGHBOR }} fall-over bfd
aggregate-address {{ AGGREGATE_IP }} {{ AGGREGATE_MASK }} summary-only
!
address-family ipv4
network {{ loopback_ip }} mask 255.255.255.255
neighbor {{ BGP_NEIGHBOR }} activate
exit-address-family
!
! Redistribute OSPF into BGP
redistribute ospf
!
! Uplink intefaces
interface TWE1/0/23
 description Towards_Firewall
 switchport mode access
 switchport access vlan 2001
 channel-group 10 mode active
 spanning-tree portfast
 no shutdown
!
interface TWE1/0/47
 description Towards_Firewall
 switchport mode access
 switchport access vlan 2001
 channel-group 10 mode active
 spanning-tree portfast
 no shutdown
!
interface Port-channel10
 description Towards_Firewall
 switchport mode access
 switchport access vlan 2001
 spanning-tree portfast
 no shutdown
!
{% endmacro %}
```

核心上行链路BGP配置

此模块生成与防火墙建立eBGP邻居关系所需的BGP配置。如上所示，大多数值都是硬编码的，因为它们在不同分支之间保持一致。只有IP地址和AS编号（因每个分支而异）会作为输入变量并用于生成必要的配置。大多数其他设置已标准化，以尽量减少变量的数量。连接到防火墙的上行链路接口与用于eBGP邻居的VLAN一起指定，后者由上一个模块生成。

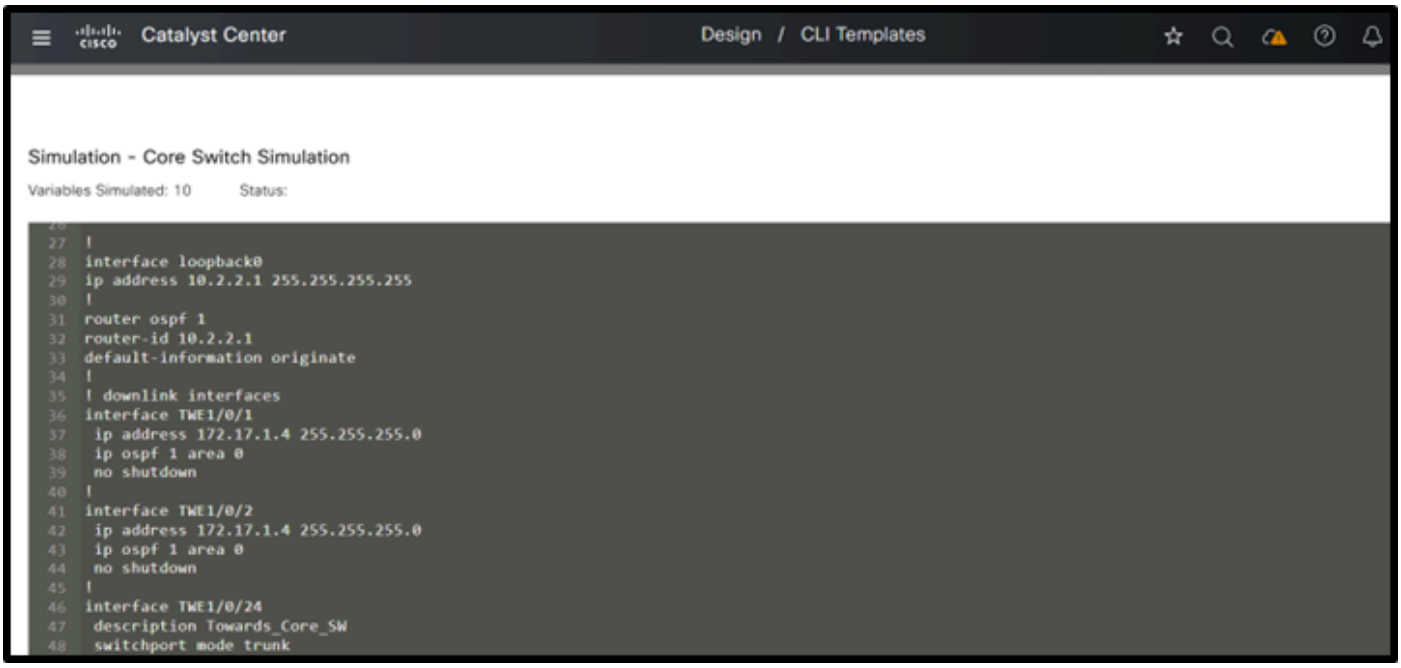
最后，最后一个模块“Core Standard Configuration”与Access Standard Configuration中描述的模块几乎相同（有关详细信息，请参阅该部分）。 它包括最佳实践、安全加固和安全设备管理的主要功能。与分布层交换机的配置一致，本模块中的源接口也可以设置为loopback0，并且此值可以硬编码。

步骤 3：执行模拟



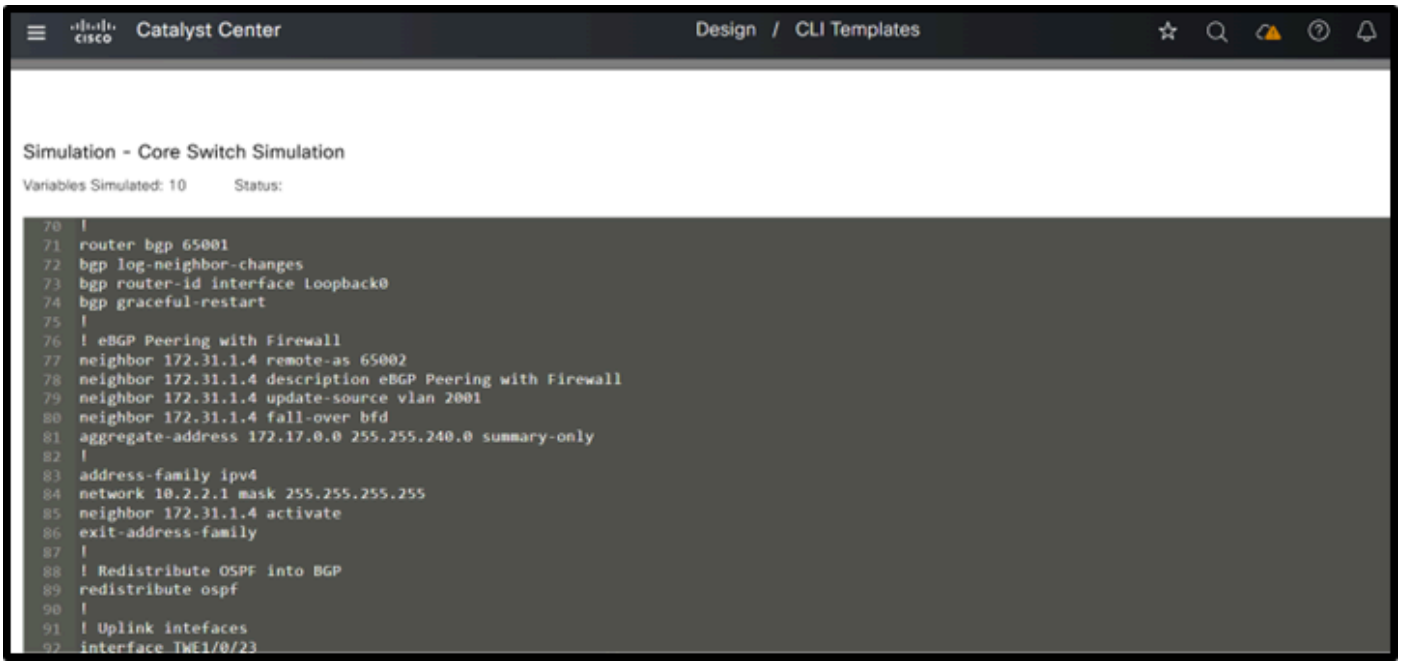
(1)核心交换机模板模拟输入和输出

(2)核心交换机模板模拟输入和输出



(3)核心交换机模板模拟输入和输出

(4)核心交换机模板模拟输入和输出



(5)核心交换机模板模拟输入与输出

这样就完成了设计三层架构模板的详细说明，概述了每个模块的结构和配置。

所有这些模块都采用了前面介绍的最佳做法。

注意：在设计折叠核心体系结构的模板时，请参阅为三层体系结构提供的说明。模板结构保持不变；但是，以前在核心层和分布层分别实施的功能现在在折叠的核心层合并。此处也可以使用相同的模块化模板方法，方法是创建一个基础模板并引用其中的相关模块。

摘要

传统的三层园区架构通常依赖于跨核心层、分布层和接入层的广泛手动配置。这种方法不仅耗时，而且容易发生人为错误。缺乏自动化和集中管理会显着增加运营开销，导致难以有效扩展和管理现代动态园区网络。通过Catalyst Center CLI模板功能配置可以自动用于传统LAN网络。在调配设备时，必须使用模块化方法。模块可以基于不同层使用的各种功能。最后将这些模块绑定到基础模块。

行动号召

我们邀请组织采用本白皮书介绍的模块化模板方法，将其作为在三层核心架构和折叠核心架构中标准化交换机配置和优化网络运营的最佳实践。

- 通过实施模块化模板，网络团队可以：
- 通过一致、可重复的配置实践提高运营效率。
- 最大程度地减少人为错误并减少故障排除时间。
- 实现更高的可扩展性，以支持增长和不断发展的业务需求。
- 确保不同环境的配置一致性。

此方法不仅简化了日常管理，而且能够加快部署速度、简化更新周期，并增强与安全和合规性要求的协调。采用模块化模板可使您的网络在不断变化的IT环境中实现灵活性、恢复能力以及长期成功。

有关实际演示，请了解有关模板的更多信息（请参阅YouTube系列）

1如何在Catalyst Center中创建和管理模板

https://youtu.be/SyUqEEcwye0

2如何在Catalyst Center的CLI模板中使用系统绑定变量

https://youtu.be/gV1QBuHYJdo

作者

Naveen Kumar，思科客户体验客户交付架构师

Risabh Mishra，思科客户体验咨询工程师

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。