Kennenlernen von Vorlagen in Catalyst Center

Einleitung

In diesem Dokument werden Cisco Catalyst Center und die Verwendung von Konfigurationsvorlagen für dreistufige oder Collapsed Core-Campus-Architekturen beschrieben.

Hintergrundinformationen

Dieses Dokument ist für professionelle Anwender gedacht, die mit Cisco Catalyst Center vertraut sind und Erfahrung mit Konfigurationsvorlagen haben. Sie ist insbesondere für diejenigen relevant, die bereits mit dreistufigen oder Collapsed Core-Campus-Architekturen gearbeitet haben oder dies planen.

Das Hauptziel besteht darin, den Leser bei der Implementierung und Automatisierung von Konfigurations- und Managementlösungen mithilfe von Vorlagen in Cisco Catalyst Center zu unterstützen. Durch die Präsentation erweiterter Einblicke, praktischer Techniken und praktischer Beispiele dient dieses Dokument als praktische Ressource für alle, die ihre LAN-Infrastruktur-Kenntnisse erweitern und Workflows durch Automatisierung und vorlagenbasiertes Management optimieren möchten.

Zusammenfassung

Mit der Weiterentwicklung von Unternehmensnetzwerken steigt auch der Bedarf an skalierbarem, konsistentem und automatisiertem Management. Cisco Catalyst Center bietet eine zentralisierte, zielbasierte Plattform, die die Konfiguration, Bereitstellung und Absicherung in Campus-Netzwerken vereinfacht. In diesem Whitepaper wird untersucht, wie Netzwerkexperten den CLI Template Editor und die Automatisierungsfunktionen von Cisco Catalyst Center nutzen können, um den Netzwerkbetrieb zu optimieren, Konfigurationsfehler zu reduzieren und Bereitstellungen in dreistufigen und zusammengefassten Kernarchitekturen zu beschleunigen. Es beschreibt Best Practices für das Design modularer Jinja2-basierter Vorlagen, die Integration der Automatisierung in Day-0- und Day-N-Workflows und die Erzielung betrieblicher Konsistenz über Core-, Distribution- und Access-Layer hinweg. Durch die Umsetzung der in diesem Dokument beschriebenen Strategien können Sie das herkömmliche manuelle Netzwerkmanagement in ein flexibles, standardisiertes und automatisierungsorientiertes Modell verwandeln, das auf die Vision von Cisco für ein intelligentes Netzwerk abgestimmt ist.

Herausforderungen von Campus-Netzwerken

Campus-Netzwerke entwickeln sich ständig weiter, um die Anforderungen moderner Unternehmen zu erfüllen. Daher stehen sie vor einer Reihe wichtiger Herausforderungen:

(2a) Komplexität des Netzwerkmanagements

Viele Netzwerkfunktionen werden weiterhin manuell verwaltet, wodurch das Risiko menschlicher Fehler steigt. Dies erhöht nicht nur den Wartungsaufwand, sondern belastet auch die IT-Ressourcen, insbesondere bei statischen oder begrenzten Budgets.

2b) Herausforderungen bei Bereitstellung und Automatisierung

Das Onboarding neuer Geräte für kabelgebundene und Wireless-Netzwerke ist häufig zeitaufwendig und komplex, was zu Verzögerungen bei der Bereitstellung und einem erhöhten Verwaltungsaufwand führt.

2c) Management von Software-Images

Die Pflege eines konsistenten "golden image" im gesamten Netzwerk ist eine Herausforderung. In vielen Netzwerken kommen unterschiedliche Betriebssysteme für kabelgebundene und Wireless-Geräte zum Einsatz, was zu Ineffizienzen und Managementschwierigkeiten führt.

2d) Uneinheitliche Netzwerkkonfigurationen

Abweichungen in den Netzwerkkonfigurationen können zu Compliance-Problemen und Ineffizienzen im Betrieb führen und die Aufrechterhaltung eines zuverlässigen und sicheren Netzwerks erschweren.
2e. Steigende Benutzererwartungen

Benutzer fordern eine unterbrechungsfreie Anbindung und eine nahtlose Anwendungserfahrung, unabhängig von Standort oder Gerät. Um diese Erwartungen zu erfüllen, müssen Netzwerke ausfallsicher und intelligent sein und in der Lage sein, sich an Veränderungen in Echtzeit anzupassen.

Zusätzlich zu diesen Herausforderungen sind moderne LAN-Infrastrukturen mit einer Vielzahl weiterer Herausforderungen konfrontiert.

Vereinfachung von Campus-Netzwerken mit Cisco Catalyst Center

Cisco Catalyst Center ist eine zentralisierte Netzwerkmanagementlösung für Campus-Netzwerke, die Hauptsitz, Zweigstellen, kabelgebundene und Wireless-Verbindungen sowie IT/OT-Umgebungen unterstützt. Die Lösung bietet flexible Bereitstellungsoptionen, darunter physische Appliances, VMware ESXi-Server oder die AWS-Cloud. Mit umfassenden Funktionen vereinfacht Catalyst Center Betriebsabläufe, verbessert die Leistung und erhöht die Sicherheit.

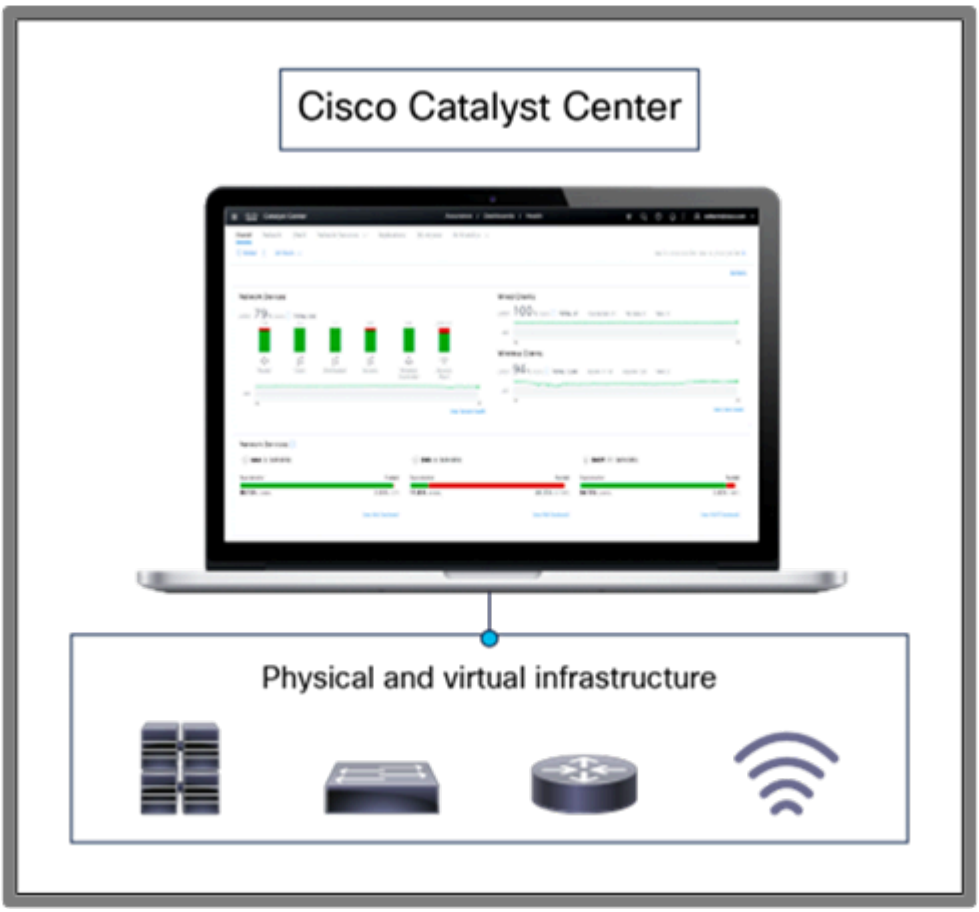


Abbildung 1: Verwaltung der Infrastruktur mit Cisco Catalyst Center

Wichtigste Funktionen und Vorteile

Cisco Catalyst Center (CC) bietet erweiterte Funktionen, die das Netzwerkmanagement und die Automatisierung optimieren:

Zero-Touch-Bereitstellung (ZTP): Automatisiert das Onboarding von Geräten und reduziert so den manuellen Aufwand und die Bereitstellungszeit.

Software-Image-Management (SWIM): Einheitliche Softwareversionen für alle Geräte durch Prüfungen vor und nach dem Upgrade, um Probleme zu vermeiden

Absichtsbasierte Automatisierung: Vereinfacht Bereitstellungen durch die Umwandlung von Netzwerkzielen in Gerätekonfigurationen für kabelgebundene und Wireless-Netzwerke.

LAN-Automatisierung: Automatisiert die Layer-3-IP-Adressierung und das Routing zur Erstellung von End-to-End-Topologien.

Wireless-Netzwerkautomatisierung: Funktionen wie Plug and Play (PnP) ermöglichen die schnelle Bereitstellung von Wireless Access Points.

Hierarchisches Netzwerkmanagement: Standortspezifische Profile (z. B. SSIDs, RF-Parameter, VLANs) für konsistente standortübergreifende Bereitstellungen

CLI-Vorlagen: Mit dem Catalyst Center Template Editor können Administratoren CLI-basierte Konfigurationsvorlagen erstellen und verwalten und so eine konsistente und effiziente Bereitstellung auf allen Geräten ermöglichen.

Gewährleistung : Assurance ermöglicht eine zentralisierte Überwachung der verwalteten Geräte über CC.

Zusätzlich zu diesen Funktionen bietet Cisco Catalyst Center viele weitere Funktionen, die den Rahmen dieses Dokuments sprengen. Dieses Whitepaper konzentriert sich hauptsächlich auf das Design von CLI-Vorlagen unter Verwendung von Catalyst Center.

Allgemeine Übersicht über die LAN Campus-Architektur mit Catalyst Center

Herkömmliche LAN-Campus-Netzwerke bilden das Rückgrat der Unternehmenskonnektivität und stellen eine zuverlässige und skalierbare Kommunikation für kabelgebundene und Wireless-Geräte sicher. Diese Netzwerke werden je nach Größe und Komplexität des Unternehmens in der Regel entweder mit der 3-Tier-Architektur oder mit der Collapsed Core-Architektur entwickelt.

Dreistufige Architektur

Die dreistufige Architektur ist ein grundlegendes Netzwerkdesignmodell, das aus Core Layer, Distribution Layer und Access Layer besteht. Diese Architektur bietet Skalierbarkeit, hohe Leistung und effizientes Datenverkehrsmanagement. Sehen Sie sich die Übersicht der einzelnen Ebenen an.

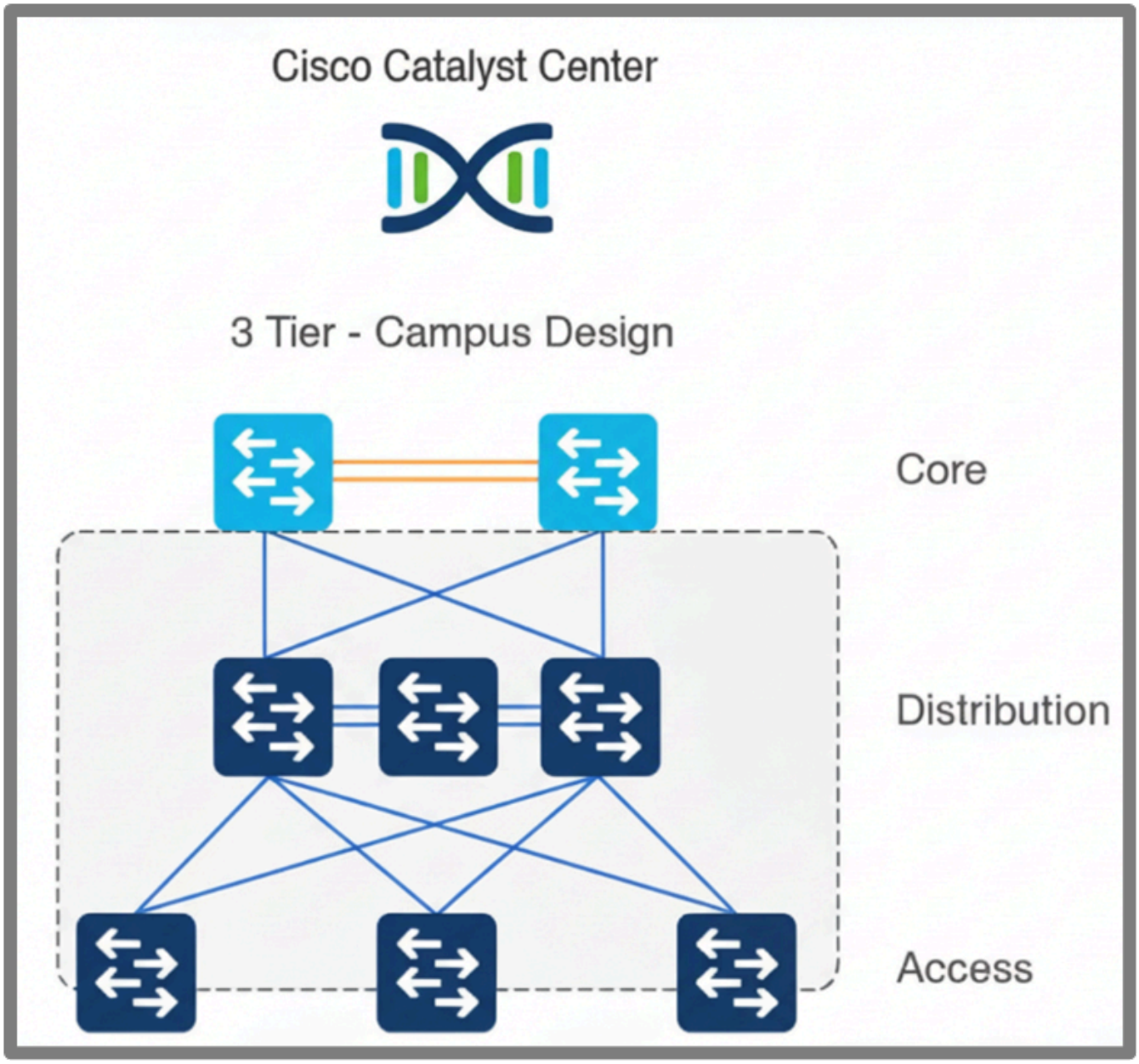


Abbildung 2: Dreistufige Campus-Architektur

Core Layer

Der Core Layer dient als Backbone des Netzwerks und bietet Hochgeschwindigkeitsverbindungen und Skalierbarkeit. Zu den wichtigsten Konfigurationen gehören Northbound- und Southbound-Routing-Protokolle (wie OSPF und BGP), Routing-Richtlinien, Downlink- und Uplink-Schnittstellenkonfigurationen, Sicherheitshärtung usw.

Distribution-Layer

Der Distribution-Layer verbindet die Core- und Access-Layer und ermöglicht so die Aggregation des Datenverkehrs, die Durchsetzung von Richtlinien und die Redundanz. Zu den wichtigsten Konfigurationen gehören HSRP/VRRP für Redundanz, STP für Loop

Prevention, Layer-2- und Layer-3-VLANs, Uplink- und Downlink-Schnittstellenkonfigurationen, ACLs für die Sicherheit und die Erhöhung der Sicherheit.

Access-Layer

Der Access-Layer verbindet Endgeräte mit dem Netzwerk und ermöglicht so einen sicheren und zuverlässigen Zugriff. Zu den wichtigsten Konfigurationen gehören die Konfiguration der Zugriffsschnittstelle, die Uplink-Schnittstellenkonfiguration, Layer-2-VLANs, ACLs zur Einschränkung des Zugriffs auf das Gerät und die Sicherheitsfunktionen.

Collapsed Core-Architektur

Die Collapsed Core-Architektur kombiniert die Core- und die Distribution-Layer in einem einzelnen Layer. Dadurch werden Komplexität und Kosten reduziert und gleichzeitig Leistung und Skalierbarkeit aufrechterhalten. Dieser Ansatz eignet sich gut für kleine bis mittelgroße Netzwerke, in denen kein separater Core-Layer erforderlich ist. Eine Übersicht über die Ebenen in dieser Architektur finden Sie hier.

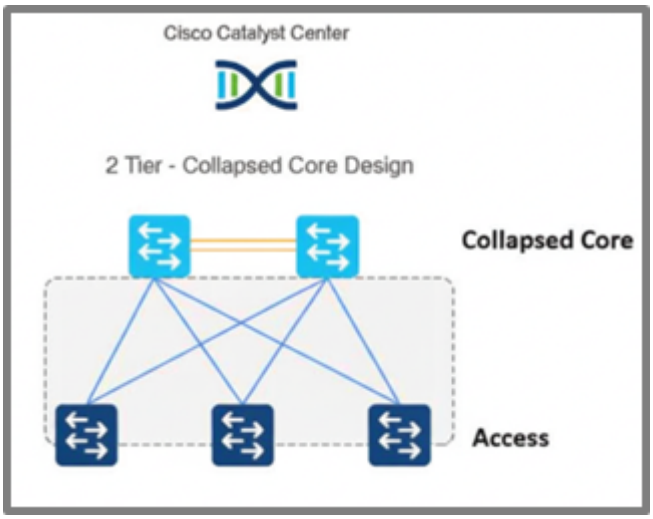


Abbildung 3: Collapsed Core Campus-Architektur

Collapsed-Core-Layer

Der Collapsed Core-Layer kombiniert die Funktionen des Core- und des Distribution-Layers und bietet so Backbone-Anbindung, Datenverkehrsaggregation und Richtliniendurchsetzung. Zu den wichtigsten Konfigurationen gehören Northbound- und Southbound-Routing-Protokolle (wie OSPF und BGP), Routing-Richtlinien, Downlink- und Uplink-Schnittstellenkonfigurationen, BFD für die Fehlererkennung, VLAN-weites Routing über SVIs, HSRP/VRRP für Gateway-Redundanz, STP für Loop Prevention und Sicherheitsmaßnahmen. Durch die Nutzung von Vorlagen in Cisco Catalyst Center können diese Konfigurationen automatisiert werden, um konsistente und effiziente Bereitstellungen zu gewährleisten.

Access-Layer

Wie zuvor beschrieben, verbindet der Access Layer die Endpunkte mit dem Netzwerk und ermöglicht so einen sicheren und zuverlässigen Zugriff. Zu den wichtigsten Konfigurationen gehören die Konfiguration der Zugriffsschnittstelle, die Uplink-Schnittstellenkonfiguration, Layer-2-VLANs, ACLs zur Einschränkung des Zugriffs auf das Gerät und die Sicherheitsfunktionen.

Überlegungen zum Vorlagendesign

In diesem Abschnitt wird beschrieben, wie Sie in Cisco Catalyst Center Vorlagen zum Generieren von Gerätekonfigurationen entwerfen. Der Vorlageneditor optimiert die Bereitstellung, indem er die Erstellung wiederverwendbarer CLI-Vorlagen ermöglicht und die dynamische Bereitstellung von Konfigurationen unterstützt, die auf Ihr Netzwerk zugeschnitten sind. Catalyst Center unterstützt zwei Vorlagensprachen: Jinja2 und Velocity Diese Sprachen unterstützen das Konfigurationsmanagement für Geräte.

Jinja ist eine beliebte, designerfreundliche Vorlagensprache, die hauptsächlich mit Python verwendet wird, um dynamische Inhalte wie HTML, XML oder andere textbasierte Formate zu generieren. Sie ermöglicht das Einbetten von Variablen und Kontrollstrukturen (wie Schleifen und Bedingungen) in Vorlagen, um dynamische Ausgaben zu erzeugen.

Apache Velocity ist eine Java-basierte Vorlagentechnologie, die mithilfe der Velocity Template Language (VTL) dynamische Inhalte in verschiedenen Dokumenten, einschließlich Webseiten, XML oder sogar Quellcode, ermöglicht. Es führt Daten aus Java-Objekten mit Vorlagen zusammen, um die endgültige Ausgabe zu erstellen.

In diesem Dokument werden nur Jinja2-Vorlagen behandelt.

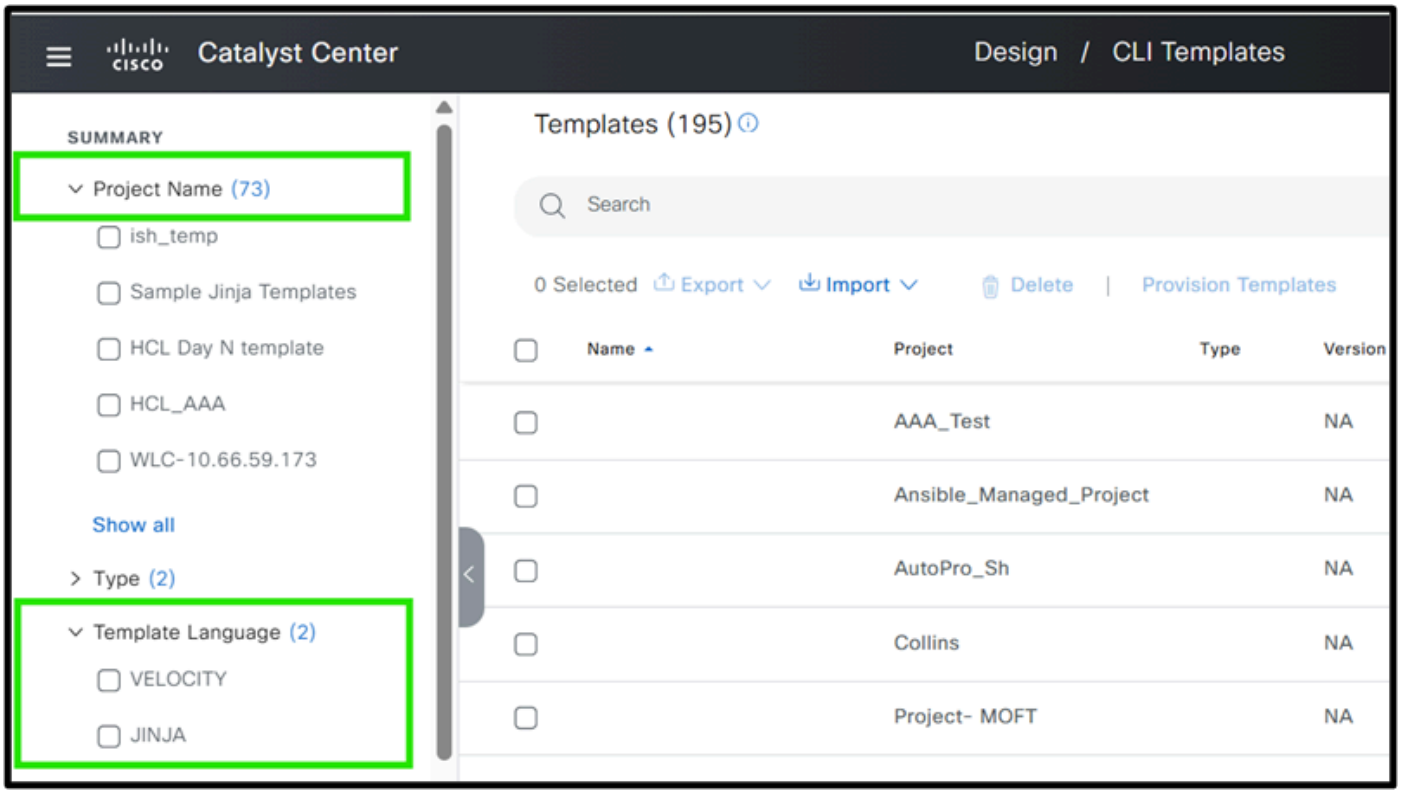


Abbildung 4: Cisco Catalyst Center Vorlageneditor

In diesem Dokument verwenden wir Jinja2 aufgrund seiner Flexibilität. Anstatt Jinja2 gründlich zu untersuchen, liegt der Schwerpunkt auf der praktischen Anwendung für das Template-Design. Weitere Informationen zur Jinja2-Vorlage im Catalyst Center finden Sie unter dem folgenden Link:

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

Bevor Sie sich näher mit Vorlagen für Designstrategien für Cisco Campus-Netzwerke befassen, sollten Sie die Best Practices nutzen, die für Effizienz und Verwaltbarkeit bei der Verwendung von Vorlagen erforderlich sind.

Vorlagenstruktur und Best Practices/Leitfaden für Best Strategy

Bei der Automatisierung der Konfiguration von Netzwerkgeräten mit Cisco Catalyst Center müssen unbedingt strukturierte Strategien und Best Practices umgesetzt werden. Diese Schritte gewährleisten Konsistenz, Skalierbarkeit und einfaches Management in der gesamten Netzwerkinfrastruktur.

- Aufteilen der Konfiguration nach Geräterolle

Zunächst müssen Geräte entsprechend ihrer Rolle in der Netzwerktopologie kategorisiert werden. Zu den allgemeinen Rollen gehören:

- Kern
- Vertrieb
- Zugriff

Beispiel: Ein Gerät, das als Core-Switch fungiert, muss andere Konfigurationsanforderungen als ein Access Switch haben.

Aufteilen der Konfiguration in Modulblöcke

Unterteilen Sie die Konfiguration innerhalb jeder Geräterolle in modulare Blöcke, indem Sie ähnliche Funktionen oder Konfigurationen gruppieren. Dieser modulare Ansatz vereinfacht Automatisierung, Fehlerbehebung und zukünftige Aktualisierungen.

Beispiele für Core-Geräte:

- OSPF-Konfigurationsblock
- BGP-Konfigurationsblock
- QoS-Richtlinienblock

Identifizieren rollenbasierter Konfigurationsbausteine

Einige Konfigurationsblöcke gelten allgemein für alle Geräterollen. Durch die Identifizierung und Standardisierung dieser Bausteine werden Best Practices und Konsistenz im gesamten Netzwerk sichergestellt.

Allgemeine rollenunabhängige Konfigurationsbausteine:

- Basiskonfiguration: Hostname, Anmeldebanner
- Management-Protokolle: DHCP, DNS, NTP, SNMP

Zugriffsrichtlinien: Standardsicherheitskonfigurationen

Diese Bausteine können für Core-, Distribution- und Access-Geräte wiederverwendet werden, wodurch der Automatisierungsprozess optimiert wird.

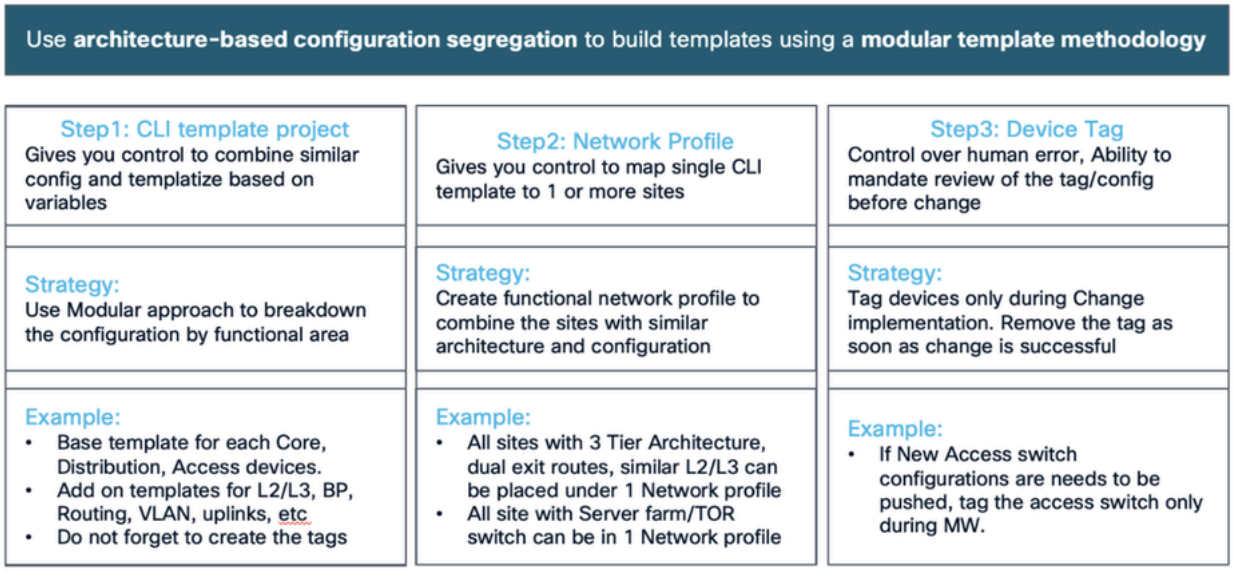


Abbildung 1: Best Practice mit Beispiel

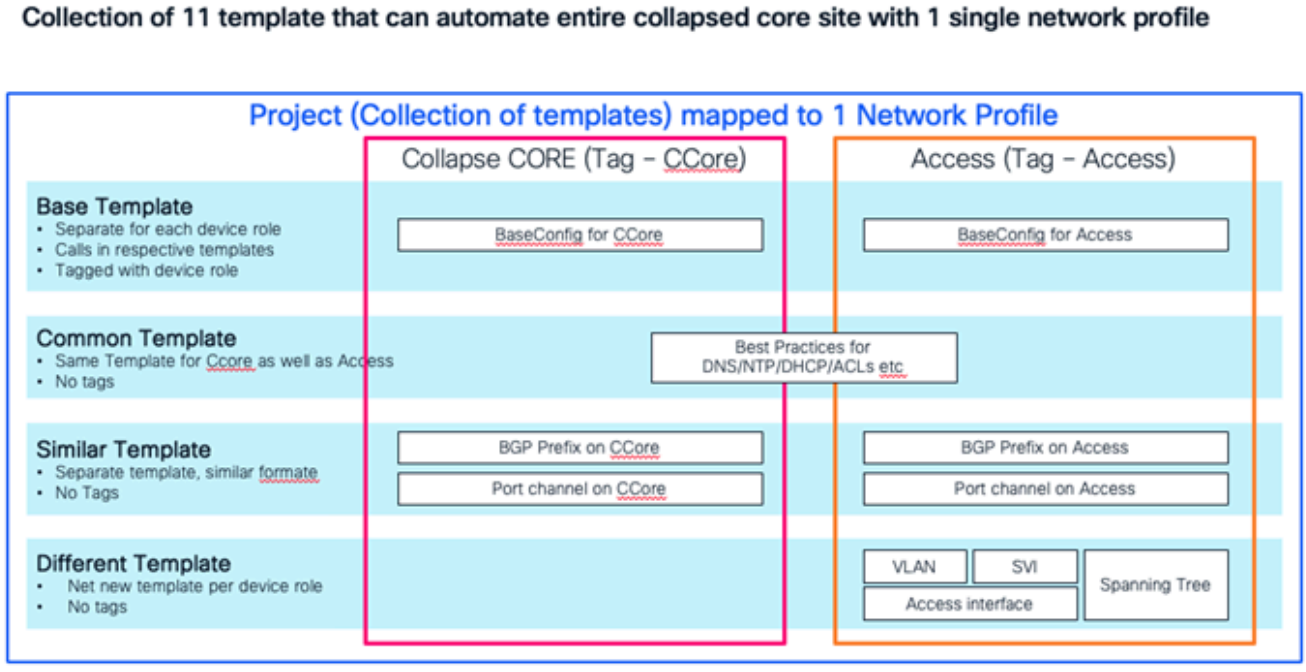


Abbildung 2: Beispiel einer Collapsed-Core-Vorlage

Best Practices für das Arbeiten mit Vorlagen

Modulares Vorlagendesign für automatisierte Konfiguration

Vermeiden Sie bei der Automatisierung von Gerätekonfigurationen in Cisco Catalyst Center die Einbettung aller Konfigurationen in eine einzige, monolithische Vorlage. Stattdessen sollte ein modularer Ansatz gewählt werden:

Erstellen Sie eine Basisvorlage, die auf kleinere, zweckspezifische Vorlagen (Module) verweist:

Unterteilen der Konfiguration in logische Module (z. B. Schnittstelleneinstellungen, Routing-Protokolle, Sicherheitsfunktionen).

Durch diese Struktur werden Aktualisierungen effizienter. Änderungen an einem bestimmten Modul werden automatisch übernommen, wo auch immer dieses Modul verwendet wird. So werden Fehler und Komplexität erheblich reduziert.

Beispiel: Modulare Konfiguration für Zweigstellengeräte

Angenommen, Sie automatisieren die Konfiguration für ein Zweigstellengerät.

Basisvorlage:

Enthält Verweise auf Modulvorlagen für wichtige Konfigurationsbereiche.

Übergibt Variablen nach Bedarf an jedes Modul zur Anpassung.

Modulvorlagen:

Schnittstelleneinstellungen: Verwaltung von Schnittstellenkonfigurationen

Routing-Protokolle: Enthält OSPF-, EIGRP- oder BGP-Einstellungen.

Sicherheitsfunktionen: Definiert ACLs, Firewall-Regeln oder andere Sicherheitsrichtlinien.

```
{% 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) }}
```

Beispiel für eine Basisvorlagenstruktur:

Bei dieser Struktur müssen alle Änderungen an Routing- oder Sicherheitskonfigurationen nur in den jeweiligen Modulen vorgenommen werden. Diese Änderungen werden sofort übernommen, wenn die Basisvorlage verwendet wird. Dadurch sind Ihre Konfigurationen einfacher zu verwalten und auf allen Zweigstellen-Routern konsistent.

Hier lautet der Projektname Zweigstelle, und unter Projekt sind drei weitere unterschiedliche Module definiert. Alle diese Funktionen werden in einer Basisvorlage kombiniert.

Minimieren von Variablen in der Vorlage

Halten Sie die Anzahl der Variablen in Ihrer Vorlage auf ein Minimum, um Komplexität und Fehler zu reduzieren. Weniger Variablen vereinfachen die Bereitstellung, insbesondere in großen Netzwerken, wodurch der Prozess effizienter und konsistenter wird.

Verwenden von Gerätetags für Vorlagen

Erstellung dynamischer und skalierbarer Jinja2-Vorlagen mithilfe von Device-Tags im Cisco Catalyst Center, z. B. für Standort, Rolle oder Standort Diese Tags ermöglichen eine bedingte Logik, die sicherstellt, dass die richtigen Konfigurationen auf die entsprechenden Geräte angewendet werden. Dieser Ansatz minimiert Fehler und vereinfacht das Vorlagenmanagement in verschiedenen Netzwerkumgebungen.

Statische Hardcode-Werte, sofern möglich

Statische Hardcoding-Werte können Vorlagen vereinfachen und die Bereitstellungseffizienz verbessern. Gängige Beispiele sind IP-Adressen für DNS-, NTP- oder Syslog-Server, die in der Regel auf allen Geräten gleich bleiben. Die Verwendung von Standard-VLAN-IDs auf Access Switches ermöglicht ebenfalls eine feste Codierung dieser Werte, wodurch die Variabilität reduziert und die Bereitstellung beschleunigt wird.

Einführung eines zweistufigen Ansatzes: Vorlagen für Tag 0 und Tag N

Bei der Einbindung von Geräten mit Services wie Cisco Plug and Play sollten Sie eine zweistufige Vorlagenstrategie anwenden:

Vorlagen für Tag 0: Push-Basiskonfigurationen, um sicherzustellen, dass das Gerät mit dem Cisco Catalyst Center kommunizieren kann

Vorlagen für Tag N: Bereitstellung erweiterter Funktionen und Konfigurationen, sobald das Gerät erreichbar ist

Best Practices ermöglichen effiziente und skalierbare Vorlagen, die die Bereitstellung von Cisco Campus-Netzwerken vereinfachen.

Leerraum-Steuerelement in Jinja-Vorlagenmakros

Beim Erstellen von Vorlagen in der Sprache Jinja ist es wichtig, Leerzeichen und Zeilenumbrüche sorgfältig zu handhaben, besonders wenn dynamische Inhalte in Makros wiedergegeben werden. Akkumulierte Leerzeichen oder unbeabsichtigte Zeilenumbrüche können zu Formatierungsproblemen in der generierten Ausgabe führen, die zu Fehlinterpretationen oder Fehlern in der Downstream-Verarbeitung führen müssen. Um dies zu erreichen, stellt Jinja Syntax zum Steuern von Leerzeichen bereit: Platzieren eines Minuszeichens (-) direkt in den Trennzeichen ({{- ... -}} oder {%- ... -%}), entfernen Sie alle führenden oder nachgestellten Leerzeichen um den Ausdruck. Wenn Sie beispielsweise {{item[1]}} durch {{- item[1] -}} ersetzen, wird sichergestellt, dass alle zusätzlichen Leerzeichen oder Zeilenumbrüche entfernt werden, wenn das Makro gerendert wird. Diese Vorgehensweise ist besonders hilfreich, wenn Sie Listen durchlaufen oder Konfigurationsdateien generieren, wie im Vorlagenausschnitt gezeigt. Wir empfehlen, in solchen Szenarien immer die Leerzeichen-Steuerung anzuwenden, um eine saubere und vorhersagbare Ausgabe zu gewährleisten.

Beispiel (empfohlene Verwendung):

```
{% für Element in Platzhalterliste %}
   {% bei Element[0] == Präfix -%}
      {{- Element[1] -}}
   {%- endif %}
{%- endfor %}
```

Dreistufige Architektur

Dieses Whitepaper beginnt mit der Entwicklung von Vorlagen für Access Switches bis hin zu Core Switches und beschreibt die Anforderungen für die einzelnen Layer.

Access-Layer-Switches

Access-Switches werden per Plug-and-Play integriert und müssen eine Day-0-Vorlage aufweisen. Weitere Informationen zum Plug-and-Play-Prozess in Catalyst Center finden Sie unter:

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

Wie bereits erwähnt, unterstützt Catalyst Center sowohl Velocity als auch Jinja2-Vorlagensprachen. Dieses Dokument verwendet Jinja2, um die Struktur der Vorlage zu veranschaulichen, aufgrund seiner Flexibilität. Die Access Layer-Switch-Konfiguration kann mithilfe der Day-0- und Day-N-Vorlage bereitgestellt werden.

Eine einfache Day 0-Vorlage kann strukturiert werden, siehe Schritt 1:

Schritt 1: Vorlage definieren

```
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
!
```

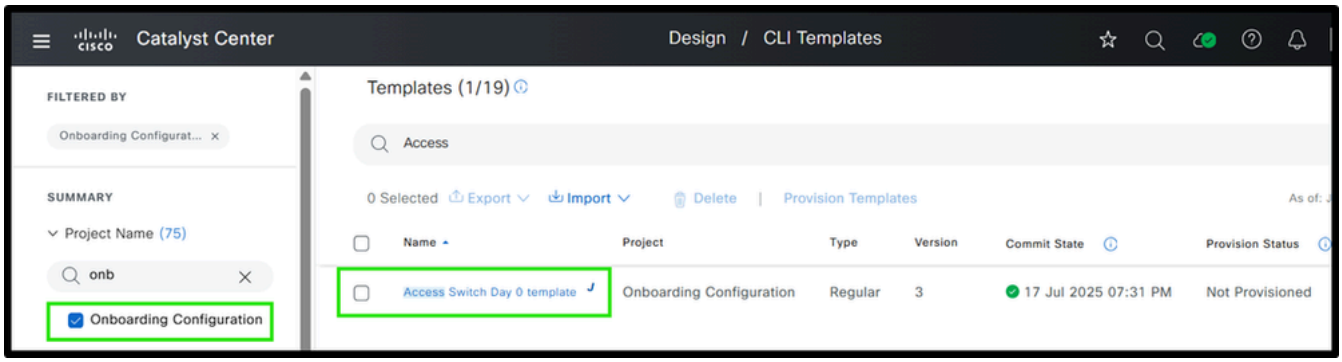
Schritt 1: Vorlage definieren

Die Vorlage vereinfacht die Konfiguration durch hardcodierte Konstanten wie Benutzername, Kennwort, enable secret und Subnetzmaske, da alle Switches in einer Außenstelle die gleiche

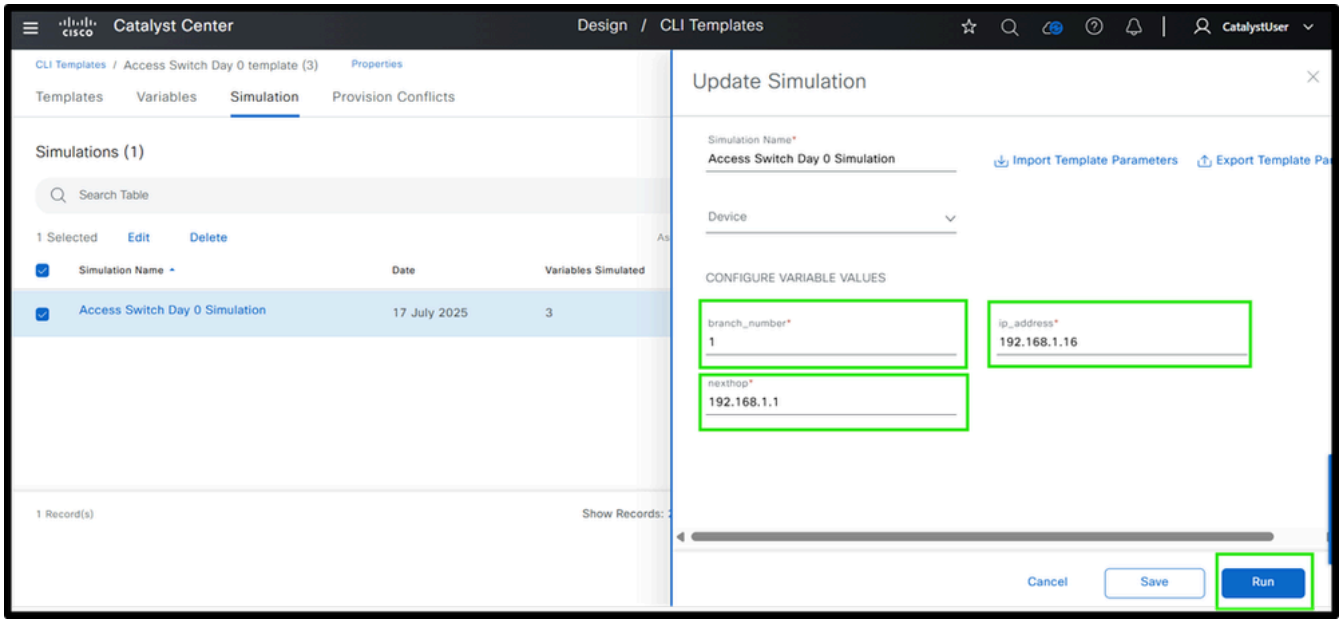
Management-VLAN-Subnetzmaske verwenden. Die Management-IP-Adresse ist jedoch für jeden Switch eindeutig und wird als Variable definiert. Eine umfassende Vorlagenstruktur muss in der Vorlage Tag N bereitgestellt werden, die diese Vorlage Tag 0 verwendet. In der Day N-Vorlage wird jede Funktion des Access Switches durch ein dediziertes Modul verwaltet. Ein Modul übernimmt beispielsweise Layer-2-VLANs, separate Module verwalten Uplink- und Downlink-Zugriffsschnittstellen, ein anderes Modul konzentriert sich auf die Sicherheitssicherung usw.

Obwohl konsistente VLAN-IDs bevorzugt werden, können unterschiedliche IDs dynamisch mithilfe einer Formel generiert werden, die auf der Zweigstellennummer basiert (Beispiel: Zweig 1 = VLAN 113, Zweig 2 = VLAN 213). Dadurch kann die Vorlage über Zweigstellen hinweg wiederverwendet werden. Ebenso ist die Next-Hop-IP eine Variable, da sie sich je nach verbundenem Verteilungs-Cluster unterscheiden muss.

Phase 2: Simulation durchführen und Variable bereitstellen

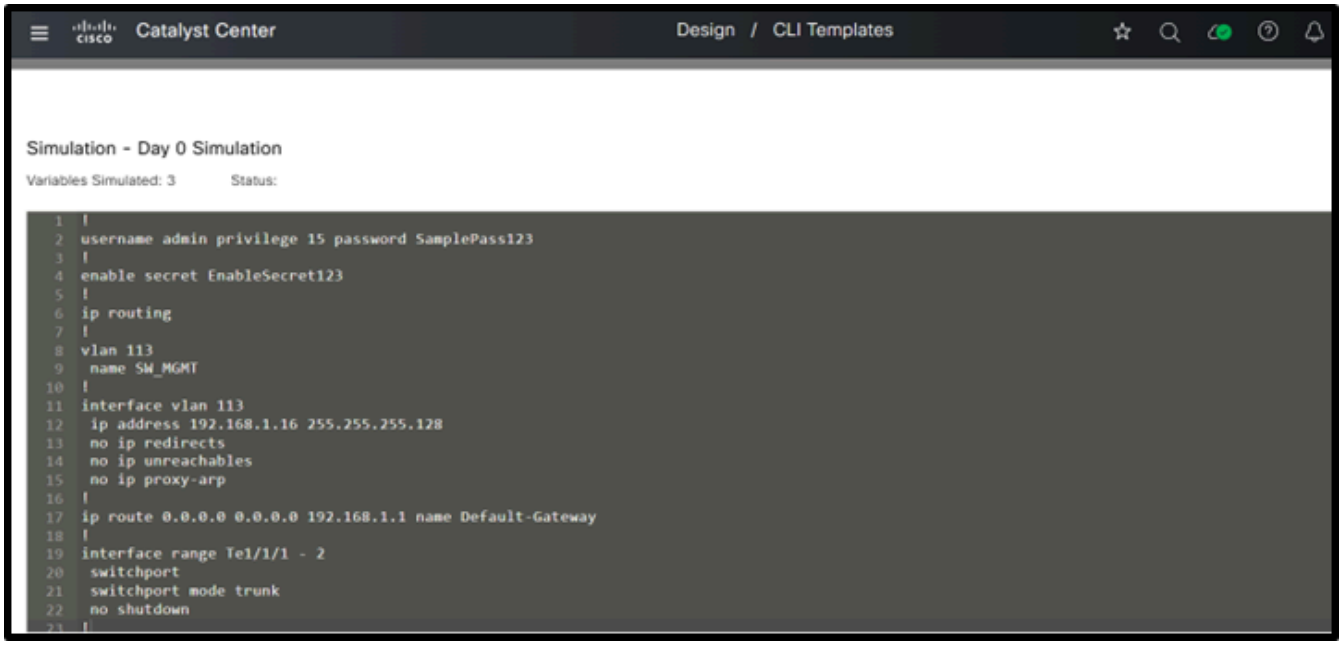


Template-Struktur des Access Switches Tag 0 mit Simulationseingängen und -ausgängen



Beispiel: Simulationseingänge

Es wird immer empfohlen, die Vorlage vor der Bereitstellung zu simulieren. Der Screenshot zeigt die endgültige Konfiguration nach Eingabe der Variablen.



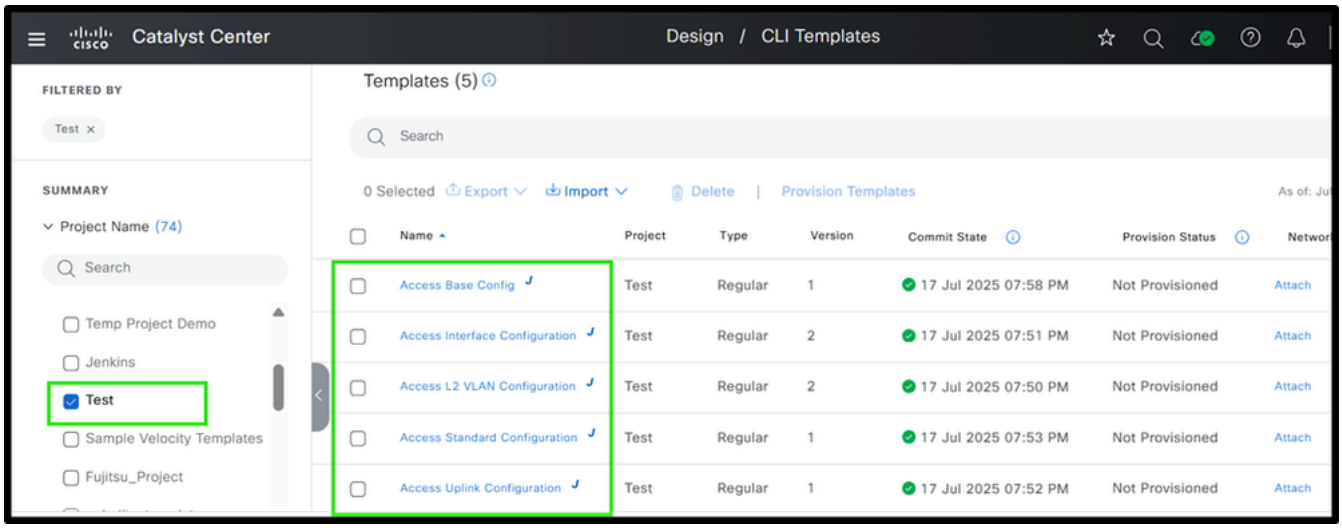
Endkonfig. nach Eingabe der Werte

Sehen wir uns nun an, wie eine modulare Day N-Vorlage erstellt wird.

Die Access Switch-Konfiguration kann in verschiedene Module aufgeteilt werden, die alle in einem Basismodul zusammengefasst werden können. Die Basisvorlage für Access Switches ist wie dargestellt aufgebaut.

Sowohl die Basisvorlage als auch die zugehörigen Module werden im Rahmen des Projekts "Test" in Cisco Catalyst Center erstellt.

Schritt 1: Verschiedene Vorlagen einschließlich Basisvorlage definieren



Tag-N-Vorlagenstruktur des Access Switches

Phase 2: Definieren verschiedener Module

Basiskonfiguration des Zugriffs:

Der Screenshot zeigt ein Beispiel für die Basiskonfiguration.

```
{% 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) }}
```

Basiskonfiguration des Zugriffs

Diese modulare Konfigurationsvorlage besteht aus vier Teilen: VLAN-Konfiguration, Uplink-Schnittstellenkonfiguration, Konfiguration der Zugangsschnittstelle und Standardkonfiguration. Es werden nur zwei Variablen verwendet: branch_number und is_poe, damit sie einfach und leicht zu verwalten sind.

Die branch_number berechnet verzweigungsspezifische VLAN-IDs, wie in der Vorlage Day 0 (Tag 0) dargestellt. is_poe bestimmt, ob es sich bei dem Access Switch um einen PoE- oder einen nicht-PoE-basierten Switch handelt. Diese Variablen werden bei der Bereitstellung bereitgestellt und an die Module übergeben, um die richtigen Konfigurationen zu erstellen. Dadurch wird der Aufwand verringert und die Effizienz verbessert.

Sehen wir uns nun die einzelnen Module an, um festzustellen, wie sie zur Generierung bestimmter Teile der Gesamtkonfiguration beitragen.

Access-L2-VLAN-Konfiguration

```
{% 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 %}
```

Access-L2-VLAN-Konfiguration

Dieses Modul erstellt VLANs basierend auf der Zweigstellennummer, wie zuvor erläutert. Daten- und Sprach-VLANs werden auf allen Switches erstellt, unabhängig davon, ob sie PoE unterstützen oder nicht. Das AP-Management-VLAN (z. B. 114 für Zweig 1) wird nur erstellt, wenn is_poe auf "Yes" (Ja) gesetzt ist, d. h. der Switch unterstützt PoE. Wenn is_poe "Nein" ist, wird das AP-Management-VLAN übersprungen, da Access Points von Nicht-PoE-Switches nicht unterstützt werden können. Dies wird mithilfe einer if-Bedingung verwaltet.

```
{% 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 %}
```

Konfiguration der Zugriffsschnittstelle

Dieses Modul übernimmt die Konfiguration der Zugriffsschnittstelle und verwendet den gleichen Ansatz wie der zuvor beschriebene PoE-Switch. Wenn die Variable is_poe "Yes" (Ja) lautet, d. h. der Switch ist ein PoE-Switch, müssen die ersten sechs Ports (1-6) mit dem AP-Management-VLAN konfiguriert werden. Andernfalls müssen die ersten sechs Ports als Benutzerzugriffsports festgelegt werden.

Wenn es sich um einen Switch mit 24 Ports handelt, werden die verbleibenden Ports (7-24) immer als Benutzerzugriffsports konfiguriert, unabhängig davon, ob es sich um einen PoE-Switch handelt oder nicht.

Der Schnittstellenbereich wurde standardisiert und wird nicht mehr als Eingangsvariable betrachtet. Dies gilt als Best Practice zur Minimierung der Anzahl von Variablen in der Vorlage. Darüber hinaus enthält das Modul ein Makro mit dem Namen common_access_settings, das die Vorlagengröße minimiert, indem wiederholte Konfigurationen konsolidiert werden. Dieses Makro wird einfach innerhalb der

Schnittstelleneinstellungen aufgerufen, sodass es nicht erforderlich ist, sie mehrmals anzugeben.

Anmerkung: Diese Vorlage wendet dieselbe Beschreibung auf alle Zugriffsschnittstellen an. Wenn für jede Schnittstelle eindeutige Beschreibungen erforderlich sind, wird empfohlen, diese mithilfe separater Python-Skripts oder ähnlicher Automatisierungstools per Push bereitzustellen.

Überprüfen Sie das Modul, das Konfigurationen für Uplink-Schnittstellen generiert.

```
{% 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 %}
```

Access Uplink-Konfiguration

Dieses Modul generiert die Konfiguration für Uplink-Schnittstellen und behandelt VLAN-Bereinigung. Wenn der Switch PoE unterstützt, ist das AP-Management-VLAN in der Liste der zulässigen VLANs enthalten. ansonsten ist sie ausgeschlossen. Diese Logik wird, wie bereits beschrieben, mithilfe der if-Bedingung im Code verwaltet.

Sehen Sie sich das letzte Modul an, das die Standardkonfigurationen, einschließlich Best Practices und Sicherheitsmaßnahmen, demonstriert.

Vorsicht: Beachten Sie, dass dies nur zur Veranschaulichung dient und nicht als Referenz für die tatsächliche Netzwerkeinrichtung verwendet werden darf, da die Konfigurationen je nach den spezifischen Anforderungen variieren können.

```
{% 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
```

Teil 1: Zugriff auf die Standardkonfiguration

```
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 %}
```

Teil 2: Zugriff auf die Standardkonfiguration

Dieses Modul generiert eine Standardkonfiguration, die Best Practices, Sicherheitsfunktionen und wichtige Funktionen für die sichere Geräteverwaltung umfasst. Die meisten Werte sind aus Gründen der Konsistenz über mehrere Zweigstellen hinweg fest codiert, mit Ausnahme von branch_number, der zur Berechnung des Management-VLAN für Switches in den einzelnen Zweigstellen verwendet wird und als Quellschnittstelle für verschiedene Konfigurationen dient.

Schritt 3: Führen Sie vor der Konfiguration der Switches eine Simulation durch. Es muss nur die Basiskonfiguration simuliert werden.

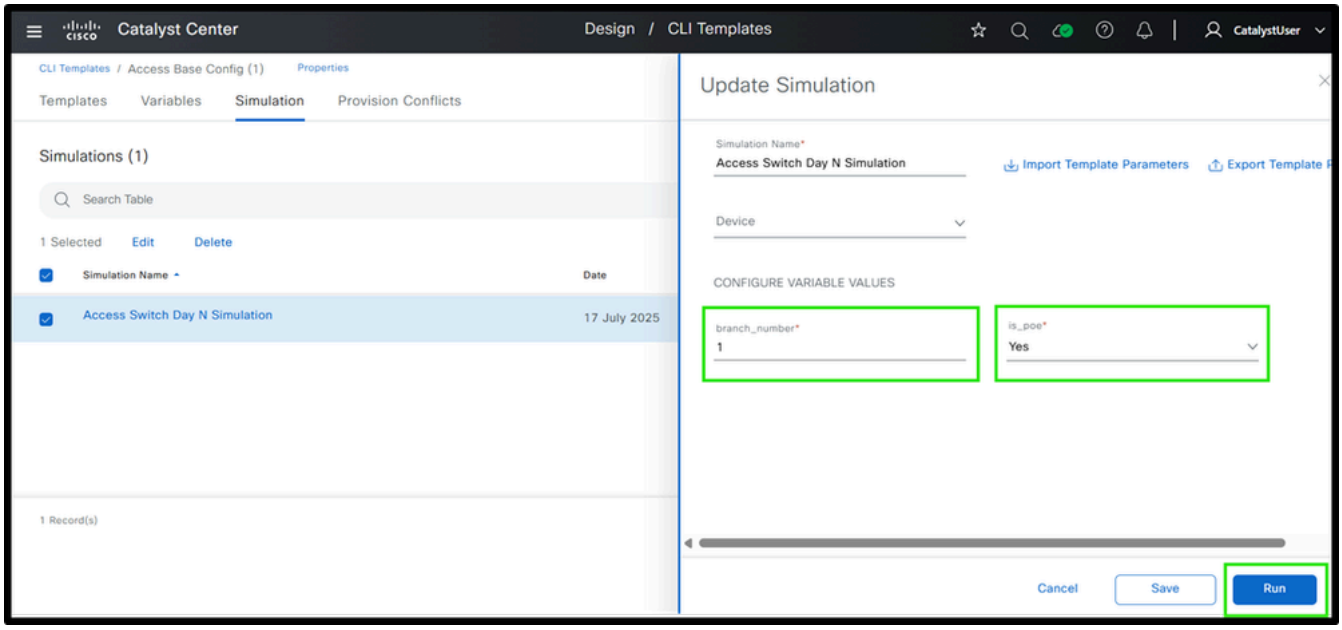
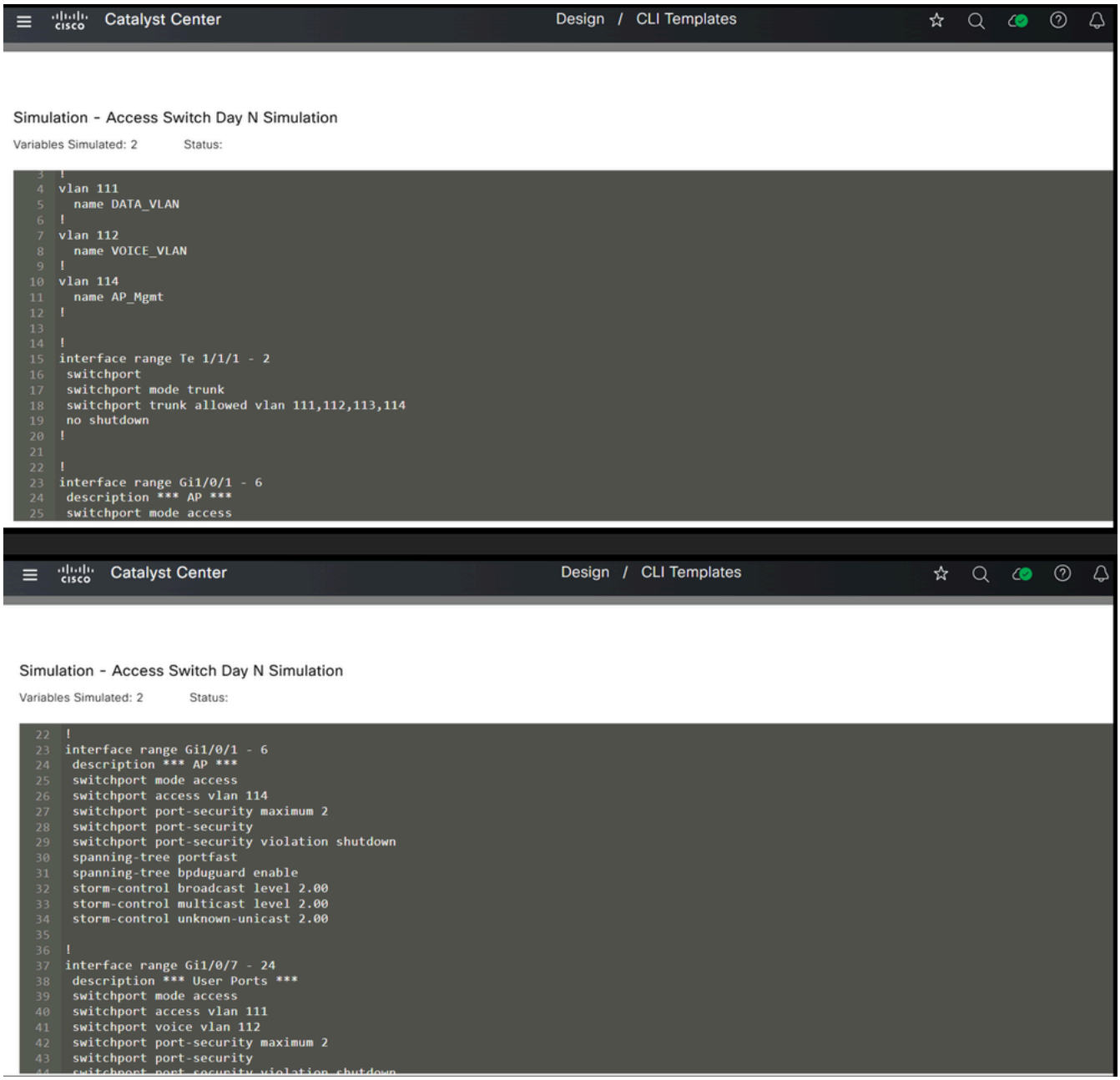


Abbildung 7: Access Switch Day N Template Simulation Eingänge und Ausgänge



Simulation

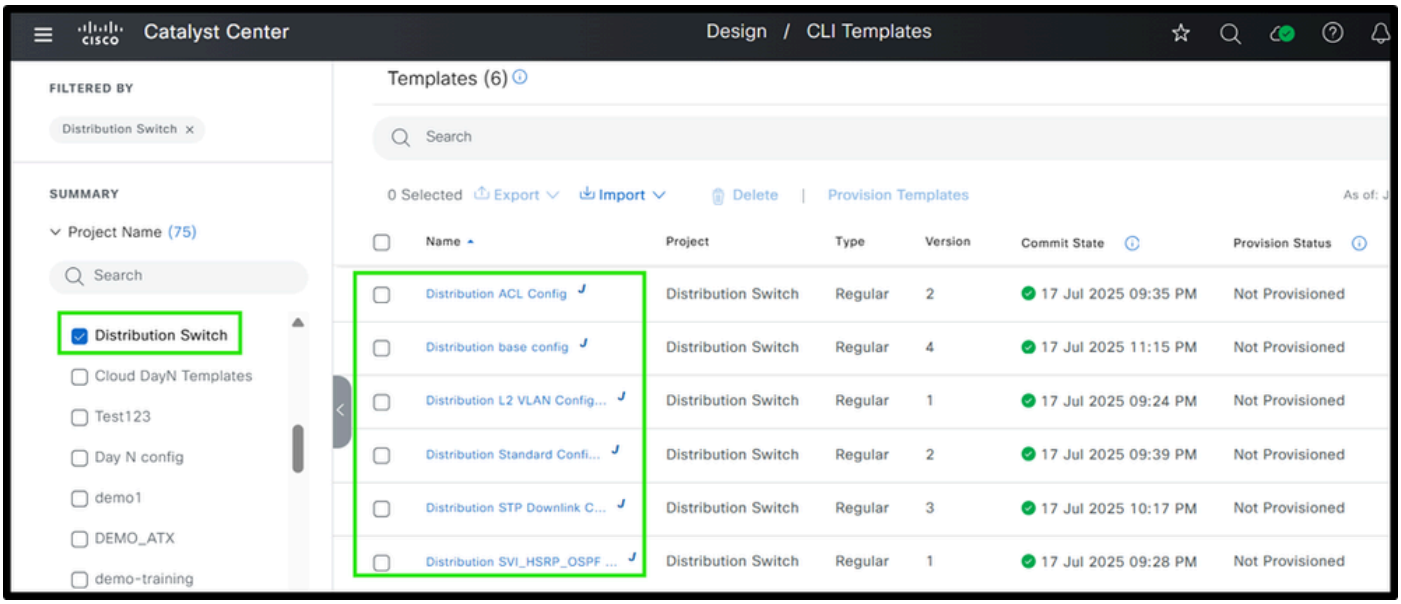
Auf diese Weise können Vorlagen auf dem Access Layer zum Generieren von Konfigurationen verwendet werden.

Werfen wir nun einen Blick auf die Distribution-Layer-Geräte, um zu sehen, wie Vorlagen darauf angewendet werden können.

Distribution-Layer-Switches

Nun zur Entwicklung einer modularen Vorlage für Distribution-Switches. Die Basisvorlage und die zugehörigen Module sind Teil des Distribution Switch-Projekts im Cisco Catalyst Center.

Schritt 1: Struktur der Distribution Switch-Vorlage



Beispiel: Verteilungsvorlagen

Schritt 2: Definieren der einzelnen Module

Die bereitgestellte Basiskonfiguration definiert, auf welche Module jeweils Bezug genommen wird.

```
{% 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() }}
```

Beispiel: Distribution Base-Vorlagenmodule

Ähnlich wie bei Access Switches werden alle Vorlagen im Projekt "Distribution Switch" erstellt und in der Basisvorlage referenziert. Während einige Vorlagen mit denen für Access Switches identisch sind, werden in diesem Abschnitt die spezifischen Unterschiede zu Distribution Switches erläutert. Das Modul "Distribution L2 VLAN Configuration" ist identisch mit dem zuvor für Access Switches beschriebenen Modul. Überprüfen Sie das Modul Access L2 VLAN Configuration, das diese Informationen bereitstellt. Er generiert die erforderlichen VLANs auf Grundlage der für die Variablen bereitgestellten Eingabewerte.

Sehen Sie sich jetzt das Modul "Distribution STP Downlink Config" an, das die Generierung von Spanning Tree- und Uplink-Konfigurationen für Distribution Switches übernimmt.

```
{% 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 %}
```

Distribution STP Downlink-Konfiguration

Hier wird die Makrofunktionalität Jinja2 verwendet, auf die im Basismodul verwiesen wird. Diese Struktur hilft beim Aufbau eines modularen Ansatzes.

Dieses Modul konfiguriert Spanning Tree Protocol (STP)- und Downlink-Schnittstellen basierend auf der "branch_number" und darauf, ob der Switch PoE-fähig ist. Die Variable "branch_number" wird verwendet, um eindeutige Basis-VLANs für jede Außenstelle zu generieren und so, ähnlich wie bei den bereits genannten Access Switches, verschiedene VLANs sicherzustellen. Wenn der Switch PoE-fähig ist ("is_poe" == 'Yes'), wird der Liste ein zusätzliches VLAN, z. B. das AP-Management-VLAN, hinzugefügt. Die Variable "distribution_number" bestimmt die STP-Priorität und legt 4096 für Distribution 1 (wodurch sie zur bevorzugten Root-Bridge wird) und 8192 für sekundäre Distribution Switches fest. Schließlich werden die entsprechenden VLANs auf die Trunk-Schnittstelle angewendet, um sicherzustellen, dass nur die relevanten VLANs zulässig sind, je nachdem, ob der Switch PoE-fähig ist.

Sehen Sie sich jetzt das Modul "Distribution SVI_HSRP_OSPF Config" an, dessen Schwerpunkt auf der Einrichtung von SVIs, HSRP und OSPF für effizientes Netzwerk-Routing und effiziente

Redundanz liegt.

```
{% 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 %}
```

Distribution - SVI_HSRP_OSPF-Konfiguration

Dieses Modul, Distribution_SVI_HSRP_OSPF_Config, unterstützt die Konfiguration von SVIs, HSRP, OSPF und Uplink-Schnittstellen für Distribution Switches. In diesem Beispiel konzentrieren

wir uns auf die SVI für Daten-Subnetze. Die gleiche Methode kann jedoch auch für andere SVIs wie Sprache oder Verwaltung verwendet werden.

Wenn die IP-Adressplanung für Datensubnetze bereits abgeschlossen ist, können die IP-Adressen für jede SVI automatisch anhand der Variablen branch_number und distribution_number berechnet werden. Wenn z. B. Verzweigung 1 das Subnetz 172.17.0.0/20, Verzweigung 2 172.17.16.0/20 und Verzweigung 3 172.17.32.0/20 hat, lautet die Gateway-IP 172.17.x.1 (wobei x die Verzweigungsnummer ist). Die zweite IP-Adresse für den ersten Distribution Switch lautet 172.17.x.2, die dritte IP-Adresse für den zweiten Distribution Switch lautet 172.17.x.3. Auf diese Weise werden die IP-Adressen automatisch berechnet, wodurch Fehler reduziert und der Prozess vereinfacht werden.

Der Loopback-Schnittstelle wird eine IP aus der Variablen loopback_ip zugewiesen, die als OSPF-Router-ID dient, um ein stabiles und konsistentes Routing im Netzwerk sicherzustellen. In der OSPF-Konfiguration wird diese Loopback-IP-Adresse als Router-ID verwendet, und die relevanten Schnittstellen werden dem OSPF-Bereich 0 hinzugefügt. Für HSRP werden Prioritätswerte festgelegt: 255 für den ersten Distribution Switch und 250 für den zweiten, um ein ordnungsgemäßes Failover sicherzustellen. Darüber hinaus wird die HSRP-Authentifizierung zur Erhöhung der Sicherheit mithilfe einer Schlüsselkette (HSRP_KEY) konfiguriert.

Um eine saubere und verwaltbare Konfiguration zu gewährleisten, sind einige Werte fest codiert. So sind beispielsweise die Subnetzmaske (255.255.240.0) und bestimmte HSRP-Einstellungen (wie Version und BFD) in allen Zweigstellen gleich, wodurch sich die Anzahl der Variablen verringert. Dadurch wird die Konfiguration einfacher, einfacher anzuwenden und die Wahrscheinlichkeit von Fehlern verringert. Schließlich werden die Uplink-Schnittstellen mit IPs konfiguriert und dem OSPF-Bereich 0 hinzugefügt, um das ordnungsgemäße Routing zwischen Switches zu gewährleisten. Dieser Ansatz vereinfacht das Management des Konfigurationsprozesses, ist weniger fehleranfällig und kann flexibel auf die Anforderungen der verschiedenen Zweigstellen angewendet werden.

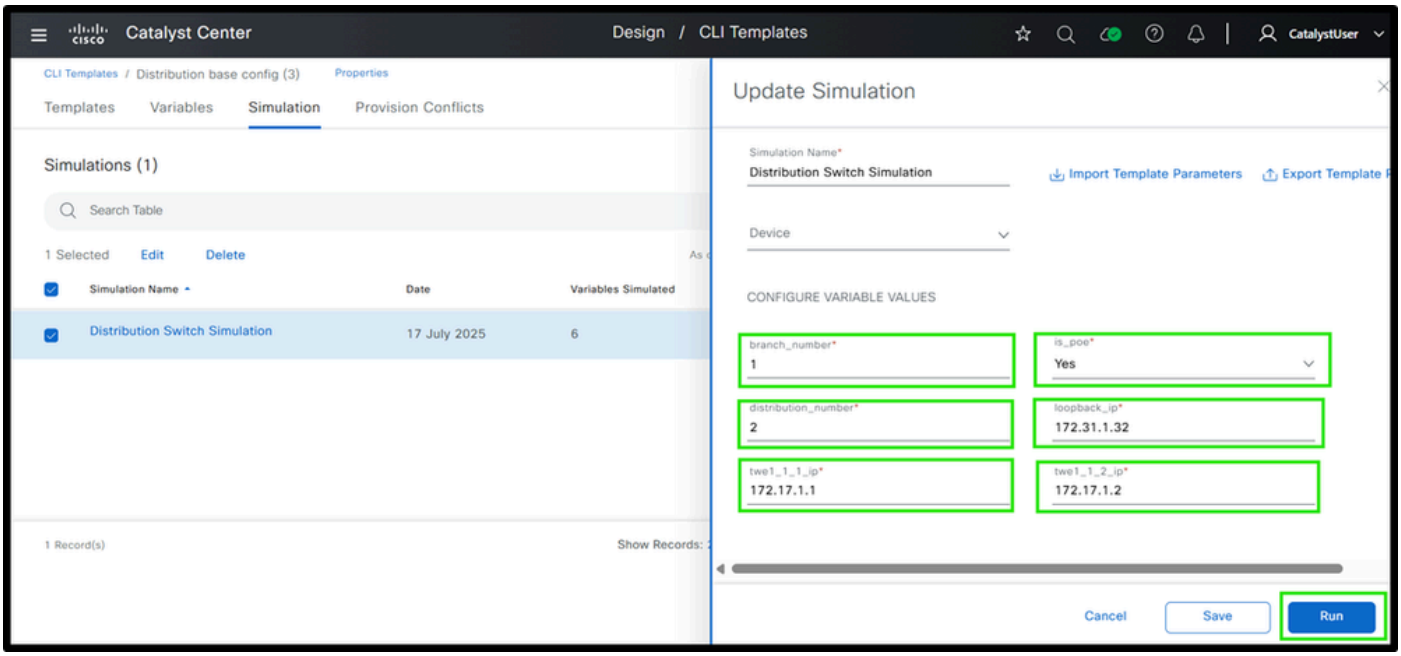
Sehen Sie sich nun das Modul "Distribution ACL Config" an, das eine Segmentierung auf dem Distribution Layer ermöglicht.

```
{% 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 %}
```

Dieses Modul demonstriert die Segmentierung auf dem Distribution Layer mithilfe einer Jinja2-Vorlage. Die Variable branch_number dient zur dynamischen Berechnung der Subnetzadressen und ermöglicht so automatisierte und skalierbare ACL-Konfigurationen. Die ACL blockiert für jede Außenstelle die Kommunikation zwischen Subnetz 1 (172.17.X.0) und Subnetz 2 (172.16.X.0), indem sie den IP-Datenverkehr zwischen diesen Bereichen verweigert. Außerdem wird der Multicast-Verkehr an die Adresse 239.255.255.250 abgelehnt, während der gesamte andere Verkehr zugelassen wird. Die VLAN-Schnittstelle wird dynamisch basierend auf der Zweigstellennummer zugewiesen, und die ACL wird eingehend auf diese Schnittstelle angewendet. Dieser automatisierte Ansatz sorgt für eine effektive Segmentierung nach Zweigstellen, reduziert manuelle Konfigurationsfehler und vereinfacht die Durchsetzung von Netzwerkrichtlinien.

Schließlich ist das letzte Modul "Distribution Standard Configuration" nahezu identisch mit dem im Modul "Access Standard Configuration" beschriebenen Modul (Details finden Sie in diesem Abschnitt). Enthalten sind Best Practices, eine robuste Sicherheitslösung und wichtige Funktionen für ein sicheres Gerätemanagement. Der einzige Unterschied liegt in der Quellschnittstelle: In der Access Switch-Vorlage ist sie als VLAN {{ Zweigstellennummer * 100 + 13 }} definiert, während sie in der Distribution Switch-Konfiguration als Loopback0 hardcodiert werden kann.

Schritt 3: Führen Sie vor der Bereitstellung der Konfiguration eine Simulation durch.



(1) Distribution Switch Template Simulation Ein- und Ausgänge

(2) Distribution Switch Template Simulation Ein- und Ausgänge

(3) Distribution Switch Template Simulation Ein- und Ausgänge

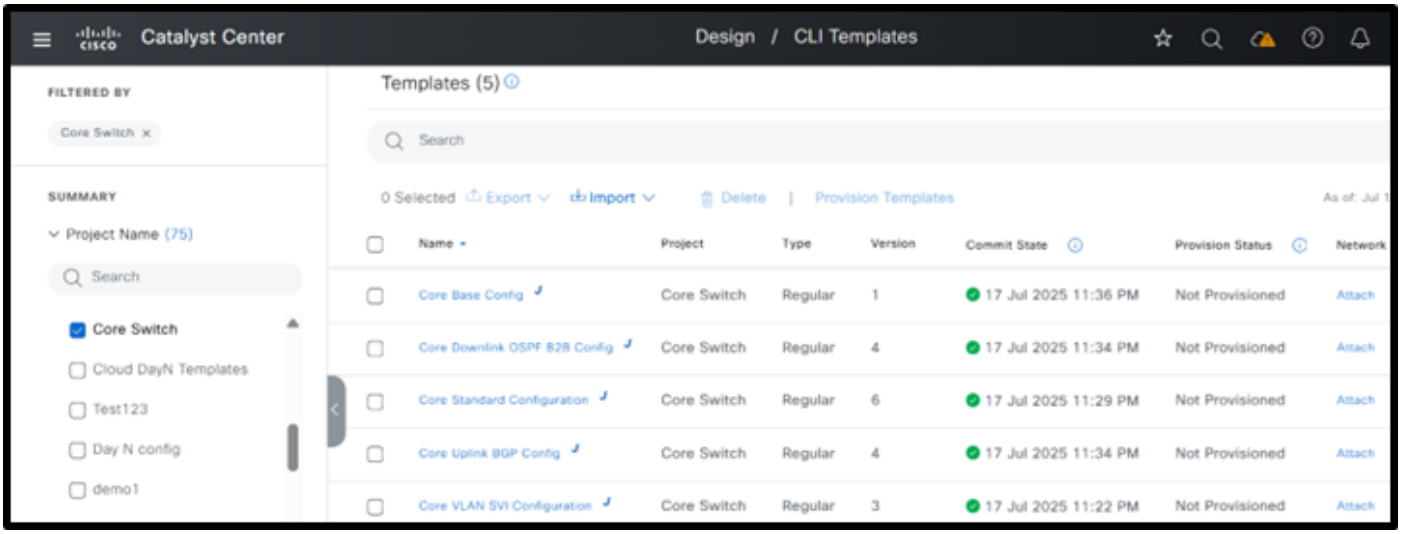
(4) Distribution Switch Template Simulation Ein- und Ausgänge

So können Vorlagen auf dem Distribution Layer verwendet werden, um Konfigurationen zu generieren. Werfen wir nun einen Blick auf die Core-Layer-Geräte, um zu sehen, wie Vorlagen dort angewendet werden können.

Core-Layer-Switches

Entwickeln Sie jetzt eine modulare Vorlage für Core-Switches. Die Basisvorlage und die zugehörigen Module sind Teil des Projekts "Core Switch" im Cisco Catalyst Center. Siehe Basisvorlage in Schritt 1.

Schritt 1: Verschiedene Core-Switch-Strukturen definieren



Struktur der Core Switch-Vorlage

Phase 2: Definieren verschiedener Module

```
{% 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 () }}
```

Core-Basiskonfiguration

Die meisten Core-Switch-Konfigurationen sind in allen Zweigstellen ähnlich, sodass gemeinsame Werte fest codiert werden können. In der Regel ändern sich nur die IP-Adressen, und diese können mithilfe von Variablen festgelegt werden. Da jede Außenstelle in der Regel nur über zwei Core-Switches verfügt, ist die Verwaltung dieser Variablen einfach. Selbst wenn einige Außenstellen über mehr Core-Switches verfügen, ist ihre Anzahl immer noch geringer als die Anzahl der Access- oder Distribution-Switches. Aus diesem Grund ist es als Best Practice wichtiger, die Variablen für Access und Distribution Switches zu minimieren, da sie in größerer Anzahl verwendet werden und zu viele Variablen die Konfiguration zeitaufwendiger machen können.

Beginnen Sie nun mit dem ersten Modul: "Core VLAN SVI Configuration." In diesem Beispiel befinden sich die Core-Switches hinter einer Firewall und müssen eBGP-Peering mit dieser einrichten. Dieses Modul ist für die Generierung der VLANs und der entsprechenden SVIs verantwortlich, die für das eBGP-Peering und die OSPF-Nachbarschaft erforderlich sind. Es wird angenommen, dass die Firewall in einer Aktiv/Standby-Konfiguration funktioniert.

```
{% 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 %}
```

Core VLAN SVI-Konfiguration

Wie bereits erläutert, erstellt dieses Modul die erforderlichen VLANs und die zugehörigen SVIs zum Aufbau von OSPF- und BGP-Nachbarbeziehungen. Alle Parameter mit Ausnahme der SVI-IP-Adressen sind fest codiert - einschließlich der Subnetzmaske, wenn diese mit dem IP-Adressierungsplan übereinstimmt. Mit dieser Methode können Sie Variablen begrenzen und das Risiko von Konfigurationsfehlern verringern.

Sehen wir uns nun das Modul "Core Downlink OSPF B2B Config" an, das Konfigurationen für Downlink-Schnittstellen, OSPF und Back-to-Back-Verbindungen zwischen Core-Switch 1 und Core-Switch 2 generiert.

```
{% 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 %}
```

Core Downlink OSPF B2B-Konfiguration

Ähnlich wie im vorherigen Modul sind die meisten Werte in diesem Modul fest codiert, um die Anzahl der Variablen zu minimieren. Nur die IP-Adressen für Loopback- und Downlink-Schnittstellen sind variabel. Darüber hinaus sind die Back-to-Back-Port-Kanäle und VLANs über verschiedene Zweigstellen hinweg standardisiert.

Sehen wir uns nun das "Core Uplink BGP Config"-Modul an, das BGP-Konfigurationen generiert und mit den Firewalls verbundene Uplinks verwaltet.

```
{% 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 %}
```

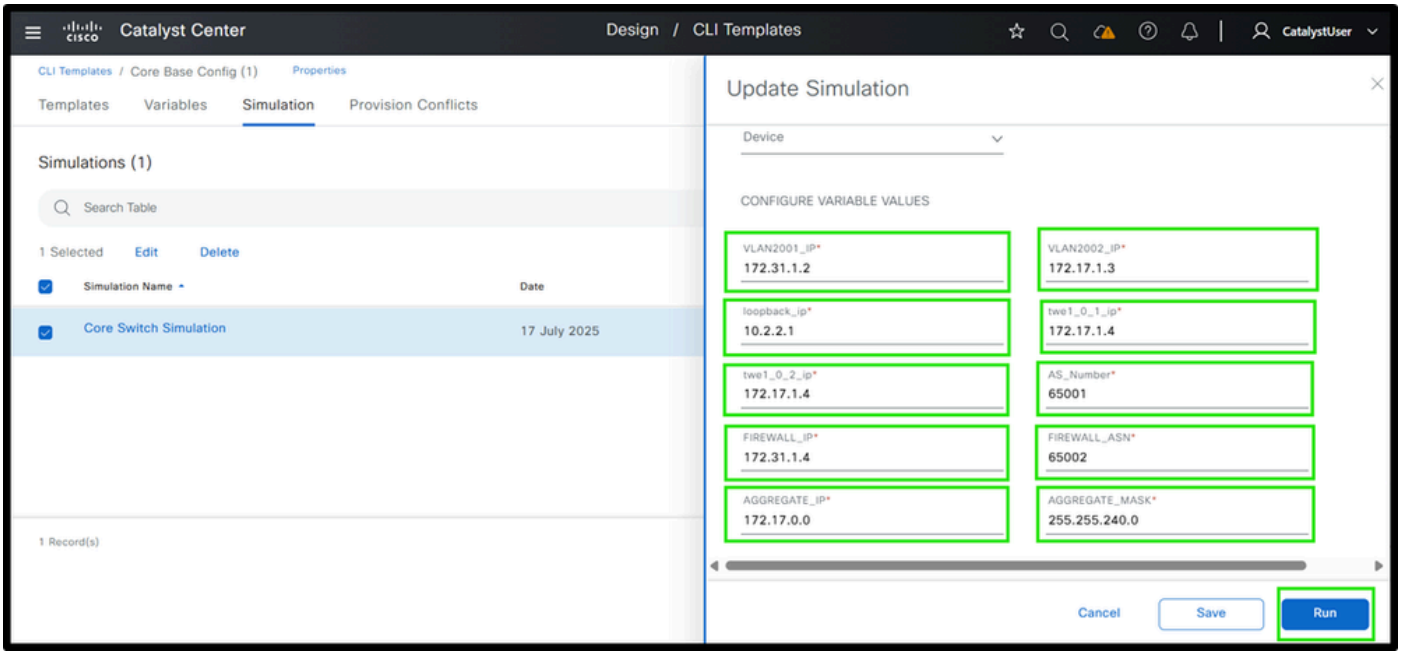
Core-Uplink BGP-Konfiguration

Dieses Modul generiert die BGP-Konfiguration, die zum Einrichten einer eBGP-Nachbarbeziehung mit der Firewall erforderlich ist. Wie oben gezeigt, sind die meisten Werte fest codiert, da sie über

verschiedene Verzweigungen hinweg konsistent bleiben. Als Eingangsvariablen werden nur die IP-Adressen und AS-Nummern verwendet, die für jede Außenstelle unterschiedlich sein können. Die meisten anderen Einstellungen wurden standardisiert, um die Anzahl der Variablen zu minimieren. Die mit der Firewall verbundenen Uplink-Schnittstellen werden zusammen mit dem VLAN für die eBGP-Nachbarschaft angegeben, das vom vorherigen Modul generiert wurde.

Schließlich ist das letzte Modul "Core Standard Configuration" fast identisch mit dem in der Access Standard Configuration beschriebenen Modul (weitere Informationen finden Sie in diesem Abschnitt). Enthalten sind Best Practices, eine robuste Sicherheitslösung und wichtige Funktionen für ein sicheres Gerätemanagement. In Übereinstimmung mit der Distribution Switch-Konfiguration kann die Quellschnittstelle auch in diesem Modul auf loopback0 gesetzt werden, und dieser Wert kann fest codiert werden.

Schritt 3: Simulation durchführen

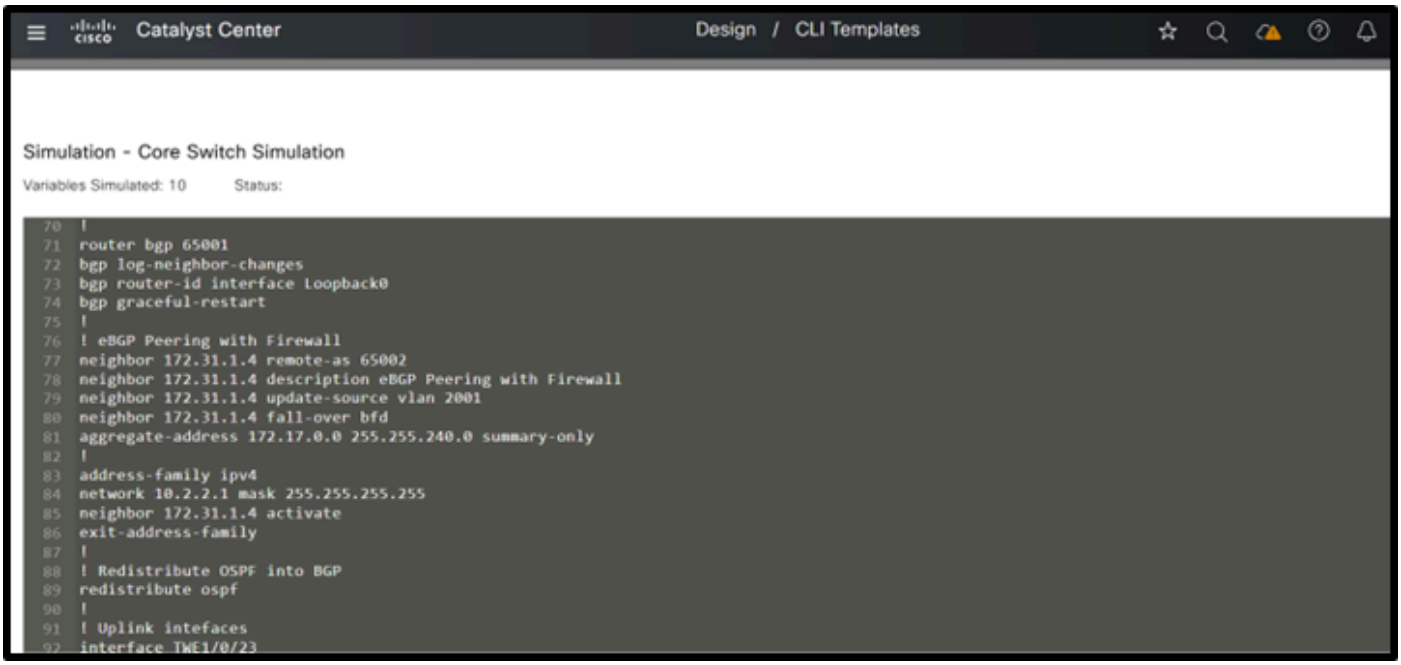


(1) Core Switch Template Simulation Eingänge und Ausgänge

(2) Core Switch Template Simulation Eingänge und Ausgänge

(3) Core Switch Template Simulation Eingänge und Ausgänge

(4) Core Switch Template Simulation Eingänge und Ausgänge



(5) Core Switch Template Simulation Eingänge und Ausgänge

Hiermit ist die detaillierte Erläuterung der Designvorlagen für die dreistufige Architektur abgeschlossen, in der sowohl die Struktur als auch die Konfiguration der einzelnen Module beschrieben werden.

Alle diese Module nutzen die zuvor erläuterten Best Practices.

Anmerkung: Wenn Sie Vorlagen für eine Collapsed-Core-Architektur entwerfen, beachten

---

Sie bitte die Erläuterungen für die dreistufige Architektur. Die Vorlagenstruktur bleibt unverändert. Funktionen, die bisher getrennt an der Core- und Distribution-Layer implementiert wurden, werden nun jedoch an der Collapsed-Core-Layer zusammengefasst. Dieselbe Methode für modulare Vorlagen kann auch hier verwendet werden, indem eine Basisvorlage erstellt wird und die entsprechenden Module darin referenziert werden.

Zusammenfassung

Die herkömmliche dreistufige Campus-Architektur basiert häufig auf einer umfassenden manuellen Konfiguration über den Core-, Distribution- und Access Layer hinweg. Dieser Ansatz ist nicht nur zeitaufwendig, sondern auch anfällig für menschliches Versagen. Durch den Mangel an Automatisierung und zentralisiertem Management steigt der betriebliche Aufwand erheblich, was die effektive Skalierung und Verwaltung moderner, dynamischer Campus-Netzwerke erschwert. Mithilfe von Catalyst Center CLI können Funktionskonfigurationen für herkömmliche LAN-Netzwerke automatisiert werden. Bei der Bereitstellung der Geräte muss der modulare Ansatz verwendet werden. Module können auf den verschiedenen Funktionen basieren, die auf verschiedenen Ebenen verwendet werden. Und schließlich binden Sie alle diese Module an das Basismodul.

Aktionsplan

Organisationen sind eingeladen, die in diesem Whitepaper vorgestellte Methodik modularer Vorlagen als Best Practice für die Standardisierung von Switch-Konfigurationen und die Optimierung des Netzwerkbetriebs über dreistufige und Collapsed-Core-Architekturen hinweg zu übernehmen.

- Die Implementierung modularer Vorlagen bietet Netzwerkteams folgende Vorteile:
- Verbesserung der Betriebseffizienz durch konsistente, wiederholbare Konfigurationsverfahren
- Minimieren Sie menschliche Fehler, und verkürzen Sie die Fehlerbehebungszeit.
- Höhere Skalierbarkeit zur Unterstützung von Wachstum und wachsenden geschäftlichen Anforderungen
- Gewährleistung konsistenter Konfigurationen in unterschiedlichen Umgebungen

Dieser Ansatz optimiert nicht nur die tägliche Verwaltung, sondern ermöglicht auch schnellere Bereitstellungen, vereinfacht Aktualisierungszyklen und optimiert die Abstimmung auf Sicherheits- und Compliance-Anforderungen. Die Verwendung modularer Vorlagen erhöht die Flexibilität und Ausfallsicherheit Ihres Netzwerks und sorgt für langfristigen Erfolg in der dynamischen IT-Landschaft.

Für praktische Demonstrationen, erfahren Sie mehr über Vorlagen , siehe YouTube-Serie

1 Erstellung und Management von Vorlagen in Catalyst Center

https://youtu.be/SyUqEEcwye0

2 Verwendung von System-Bind-Variablen in CLI-Vorlagen in Catalyst Center

https://youtu.be/gV1QBuHYJdo

Autoren

Naveen Kumar, Customer Delivery Architect, Cisco Customer Experience

Risabh Mishra, Consulting Engineer, Cisco Customer Experience

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.