Catalyst Centerのテンプレートについて

はじめに

このドキュメントでは、3層またはコラプストコアキャンパスアーキテクチャ用の設定テンプレートを使用したCisco Catalyst Centerと経験について説明します。

バックグラウンド情報

このドキュメントは、Cisco Catalyst Centerに関する基本的な知識と設定テンプレートの経験を持つエンタープライズプロフェッショナルを対象としています。これは、3層またはコラプストコアキャンパスアーキテクチャを扱ったことがある、または扱う予定がある方に特に適しています。

主な目的は、読者がCisco Catalyst Center内のテンプレートを使用して、設定および管理ソリューションを実装し、自動化するのを支援することです。このドキュメントは、高度な洞察、実用的な手法、および実際の例を示すことで、LANインフラストラクチャのスキルを強化し、自動化とテンプレートベースの管理を通じてワークフローを最適化したいと考えているユーザに実用的なリソースとして役立ちます。

要旨

企業ネットワークが進化を続けるにつれ、スケーラブルで一貫性のある自動管理のニーズが高まることはかつてありません。Cisco Catalyst Centerは、キャンパスネットワーク全体の設定、プロビジョニング、および保証を簡素化する、一元化されたインテントベースのプラットフォームを提供します。このホワイトペーパーでは、ネットワーク技術者がCisco Catalyst CenterのCLIテンプレートエディタと自動化機能を利用して、ネットワーク運用の合理化、設定エラーの削減、および3層アーキテクチャとコラプストコアアーキテクチャの導入の促進を行う方法について説明します。モジュール型のJinja2ベースのテンプレートを設計し、自動化を0日目とN日目のワークフローに統合し、コア、ディストリビューション、およびアクセスレイヤ全体で運用の一貫性を実現するためのベストプラクティスについて説明します。このドキュメントで説明されている戦略を採用することで、従来の手動によるネットワーク管理を、シスコのインテントベースのネットワーキングビジョンに沿った、俊敏性に優れ、標準化され、自動化が促進されるモデルに変えることができます。

キャンパスネットワークの課題

キャンパスネットワークが現代の組織のニーズに対応するために進化するにつれ、次のようないくつかの重要な課題に直面します。

2a.ネットワーク管理の複雑さ

ネットワーク機能の多くは依然として手動で管理されているため、人的エラーのリスクが高まります。これはメンテナンス作業を増やすだけでなく、特に静的または限られた予算では、ITリソースに負担をかけることになります。

2b.導入と自動化の課題

有線ネットワークとワイヤレスネットワークの両方に新しいデバイスを導入するには、多くの場合、時間と手間がかかり、導入の遅延や管理オーバーヘッドの増大につながります。

2c.ソフトウェア イメージの管理

ネットワーク全体で一貫した「ゴールデンイメージ」を維持することは困難です。多くのネットワークでは、有線デバイスとワイヤレスデバイスに対して複数のオペレーティングシステムが使用され、非効率性や管理上の問題が生じます。

2d一貫性のないネットワーク構成

ネットワーク構成のバリエーションが増えると、コンプライアンスの問題や運用の非効率性が発生し、信頼性の高いセキュアなネットワークの維持が難しくなります。
2e。ユーザの期待の高まり

ユーザは、場所やデバイスに関係なく、中断のない接続とシームレスなアプリケーションエクスペリエンスを求めています。このような期待に応えるには、復元力があり、インテリジェントで、リアルタイムの変化に適応できるネットワークが必要です。

このような課題に加えて、現代のLANインフラストラクチャは、その他のさまざまな複雑な問題に直面しています。

Cisco Catalyst Centerによるキャンパスネットワークの簡素化

Cisco Catalyst Centerは、本社、支社、有線およびワイヤレス接続、IT/OT環境をサポートする、キャンパスネットワーク向けの集中型ネットワーク管理ソリューションです。物理アプライアンス、VMware ESXiサーバ、AWSクラウドなど、柔軟な導入オプションを提供します。包括的な機能を備えたCatalyst Centerは、運用の簡素化、パフォーマンスの強化、セキュリティの強化を実現します。

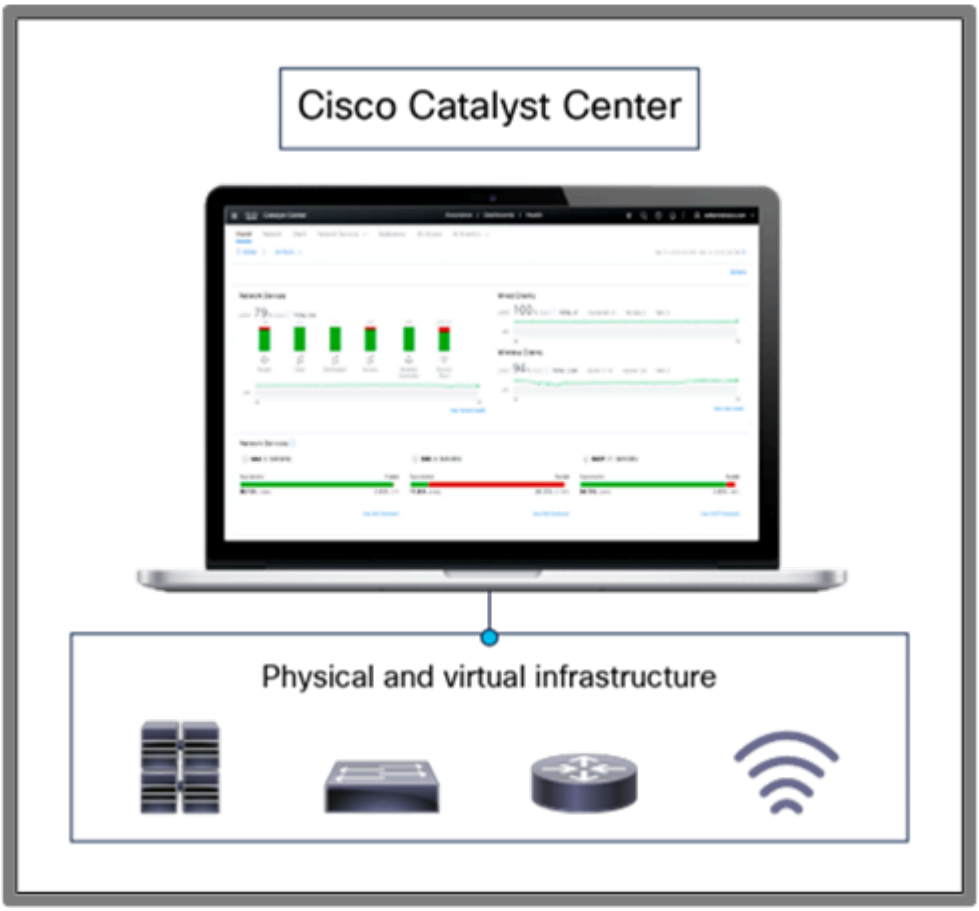


図1:Cisco Catalyst Centerによるインフラストラクチャの管理

重要な機能と利点

Cisco Catalyst Center(CC)は、ネットワーク管理と自動化を合理化する高度な機能を提供します。

ゼロタッチプロビジョニング(ZTP)：デバイスのオンボーディングを自動化し、手作業と導入時間を削減します。

Software Image Management(SWIM)：アップグレード前とアップグレード後のチェックによって問題を回避し、デバイス間でソフトウェアバージョンの一貫性を確保します。

インテントベースの自動化：ネットワークの意図を有線およびワイヤレスネットワークのデバイス設定に変換することで、導入を簡素化します。

LANの自動化：レイヤ3 IPアドレッシングとルーティングを自動化し、エンドツーエンドのトポロジを作成します。

ワイヤレスネットワークの自動化：プラグアンドプレイ(PnP)などの機能により、ワイヤレスアクセスポイントの迅速なプロビジョニングが可能です。

階層型ネットワーク管理：サイト固有のプロファイル（SSID、RFパラメータ、VLANなど）を使用して、ロケーション間で一貫した導入を実現します。

CLIテンプレート：Catalyst Centerテンプレートエディタを使用すると、管理者はCLIベースの設定テンプレートを簡単に作成して管理し、デバイス間で一貫性のある効率的な導入を実現できます。

アシュアランス：アシュアランスでは、CCを介して管理対象デバイスの集中型モニタリングを行うことができます。

これらの機能に加えて、Cisco Catalyst Centerには、このドキュメントでは説明しない多数の機能があります。このドキュメントでは、主にCatalyst Centerを使用したCLIテンプレートの設計に焦点を当てています。

Catalyst CenterによるLANキャンパスアーキテクチャの概要

従来のLANキャンパスネットワークは、エンタープライズ接続のバックボーンを形成し、有線および無線デバイスに信頼性と拡張性の高い通信を提供します。これらのネットワークは通常、組織の規模と複雑さに応じて、3層アーキテクチャまたはコラプストコアアーキテクチャのいずれかを使用して設計されます。

3層アーキテクチャ

3層アーキテクチャは、コア層、ディストリビューション層、およびアクセス層で構成される基本ネットワーク設計モデルです。このアーキテクチャは、拡張性、高パフォーマンス、および効率的なトラフィック管理を提供します。各層の概要を参照してください。

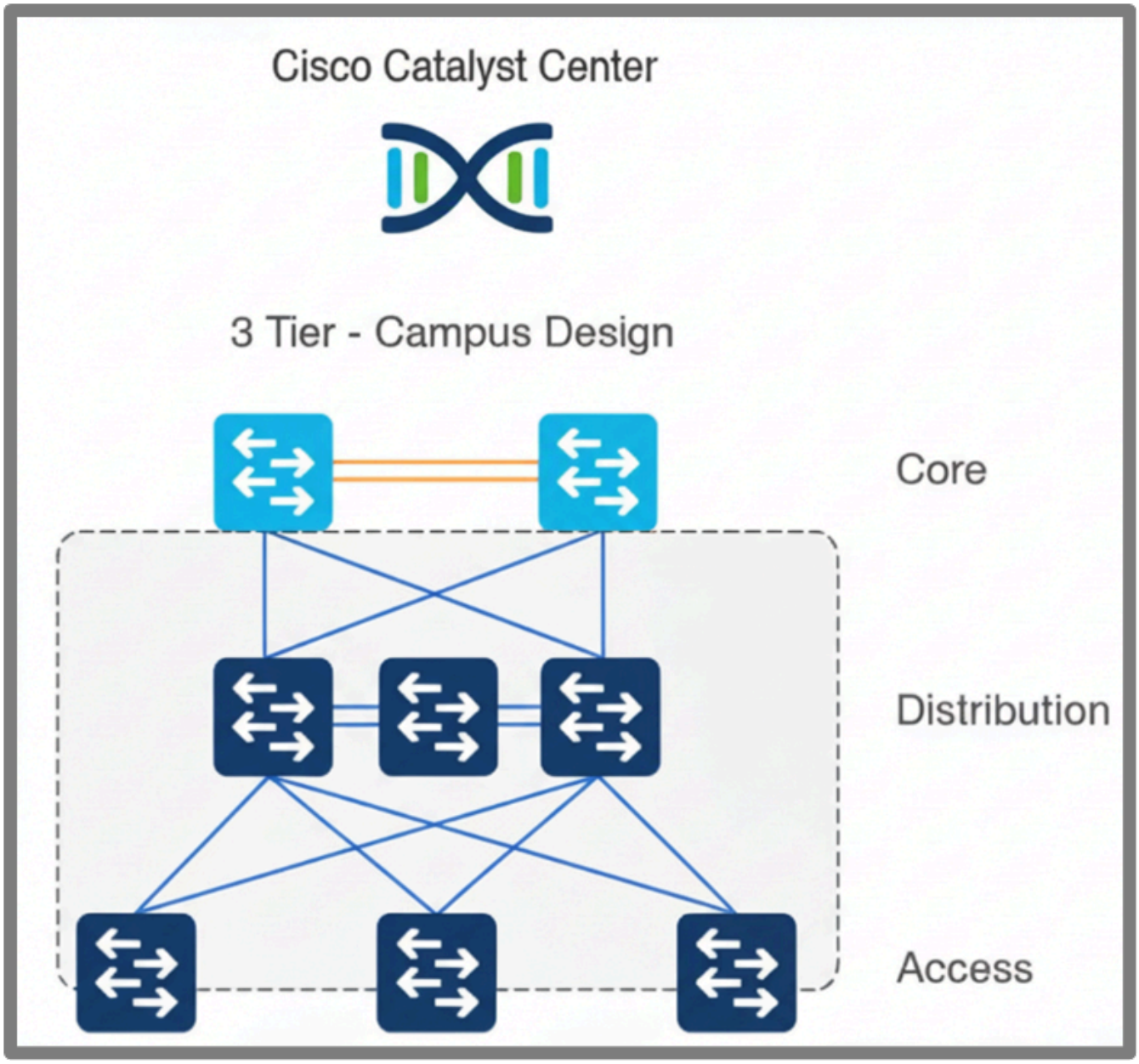


図2: 3層キャンパスアーキテクチャ

コア レイヤ

コア層はネットワークのバックボーンとして機能し、高速接続と拡張性を提供します。主要な設定には、ノースバウンドおよびサウスバウンドルーティングプロトコル（OSPFおよびBGPなど）、ルートポリシー、ダウンリンクおよびアップリンクインターフェイスの設定、セキュリティの強化などが含まれます

ディストリビューション レイヤ

ディストリビューションレイヤは、コアレイヤとアクセスレイヤをブリッジし、トラフィックの集約、ポリシーの適用、冗長性を処理します。主な設定には、冗長性のためのHSRP/VRRP、ループ防止のためのSTP、レイヤ2およびレイヤ3 VLAN、アップリンクおよ

びダウンリンクインターフェイスの設定、セキュリティのためのACL、およびセキュリティ強化のためのACLが含まれます。

アクセス レイヤ

アクセス層はエンドポイントをネットワークに接続し、安全で信頼性の高いアクセスを可能にします。主な設定には、アクセスインターフェイス設定、アップリンクインターフェイス設定、レイヤ2 VLAN、デバイスへのアクセスを制限するACL、およびセキュリティの強化が含まれます。

コラプストコアアーキテクチャ

コラプストコアアーキテクチャは、コアレイヤとディストリビューションレイヤを1つのレイヤに統合し、パフォーマンスと拡張性を維持しながら、複雑さとコストを軽減します。このアプローチは、個別のコア層が不要な小規模から中規模のネットワークに適しています。このアーキテクチャのレイヤの概要を参照してください。

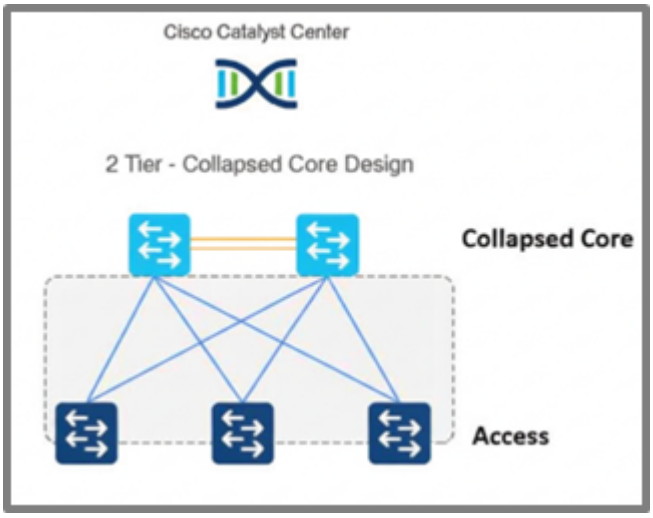


図3：コラプストコアキャンパスアーキテクチャ

コラプストコア層

コラプストコア層は、コア層とディストリビューション層の機能を組み合わせたもので、バックボーン接続、トラフィック集約、およびポリシー適用を提供します。主要な設定には、ノースバウンドルーティングプロトコルとサウスバウンドルーティングプロトコル（OSPFやBGPなど）、ルートポリシー、ダウンリンクおよびアップリンクインターフェイスの設定、障害検出用のBFD、SVIを使用したVLAN間ルーティング、ゲートウェイ冗長性のためのHSRP/VRRP、ループ防止用のSTP、およびセキュリティ強化が含まれます。Cisco Catalyst Centerのテンプレートを活用することで、これらの設定を自動化し、一貫性のある効率的な導入を実現できます。

アクセス レイヤ

前述したように、アクセス層はエンドポイントをネットワークに接続し、安全で信頼性の高いアクセスを可能にします。主な設定には、アクセスインターフェイス設定、アップリンクインターフェイス設定、レイヤ2 VLAN、デバイスへのアクセスを制限するACL、およびセキュリティの強化が含まれます。

テンプレート設計の考慮事項

このセクションでは、デバイス設定を生成するためのCisco Catalyst Centerでのテンプレートの設計方法について説明します。テンプレートエディタは、再利用可能なCLIテンプレートの作成を可能にし、ネットワークに合わせた設定の動的な導入をサポートすることで、プロビジョニングを合理化します。Catalyst Centerでは、Jinja2とVelocityの2つのテンプレート言語がサポートされています。これらの言語は、デバイスの構成管理に役立ちます。

Jinjaは、主にPythonで使用される人気のあるデザイナー向けのテンプレート言語で、HTML、XML、その他のテキストベースのフォーマットなどの動的なコンテンツを生成するために使用されます。テンプレート内に変数や制御構造（ループや条件文など）を埋め込み、動的な出力を作成できます。

Apache Velocityは、Javaベースのテンプレートエンジンで、Velocity Template Language(VTL)を使用して、Webページ、XML、ソースコードなどのさまざまなドキュメントで動的なコンテンツを有効にします。Javaオブジェクトのデータをテンプレートにマージして、最終出力を生成します。

本書では、Jinja2テンプレートのみを扱います。

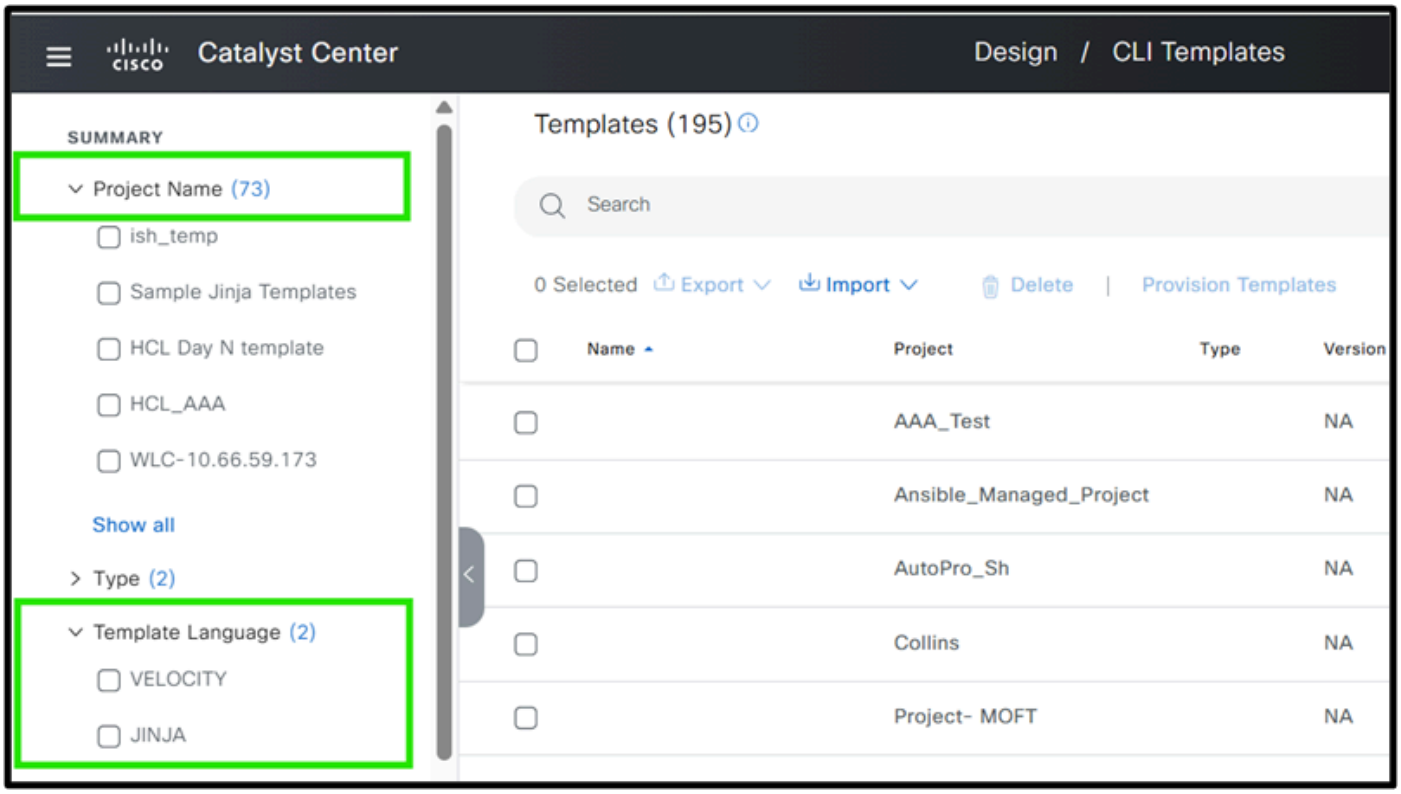


図4: Cisco Catalyst Centerテンプレートエディタ

本書では、柔軟性の高いJinja2を使用します。Shinja2を深く掘り下げるのではなく、テンプレートデザインの実用化に焦点を当てています。Catalyst CenterでのJinja2テンプレートの詳細については、次のリンクを参照してください。

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

シスコのキャンパスネットワークのテンプレート設計戦略を詳しく説明する前に、テンプレートを使用する際の効率性と管理性を確保するための主要なベストプラクティスを利用することが重要です。

テンプレートの構造とベストプラクティス/ベストプラクティスのガイドライン

Cisco Catalyst Centerを使用してネットワークデバイスの設定を自動化する際には、構造化された戦略とベストプラクティスを採用することが不可欠です。これらの手順により、ネットワークインフラストラクチャ全体の一貫性、拡張性、および管理の容易性が確保されます。

デバイスロールによる構成の分割

最初に、ネットワークトポロジでの役割に従ってデバイスを分類します。一般的な役割は次のとおりです。

- コア
- 配信
- Access

例：コアスイッチとして機能するデバイスは、アクセススイッチとは異なる設定要件を持つ必要があります。

設定をモジュール型ブロックに区分する

各デバイスロール内で、類似した機能または設定をグループ化して、設定をモジュラブロックに分割します。このモジュラ式アプローチにより、自動化、トラブルシューティング、および将来のアップデートが簡素化されます。

コアデバイスの例：

- OSPF設定ブロック
- BGP設定ブロック
- QoSポリシーブロック

ロールに依存しない設定ブロックの特定

一部の設定ブロックは、すべてのデバイスロールに共通して適用されます。これらのブロックを特定して標準化することで、ネットワーク全体のベストプラクティスと一貫性が確保されます。

一般的なロール非依存の設定ブロック：

- 基本設定：ホスト名、ログインバナー
- 管理プロトコル：DHCP、DNS、NTP、SNMP

アクセスポリシー：標準セキュリティ設定

これらのブロックは、コア、ディストリビューション、およびアクセスデバイスに再利用でき、自動化プロセスを合理化できます。

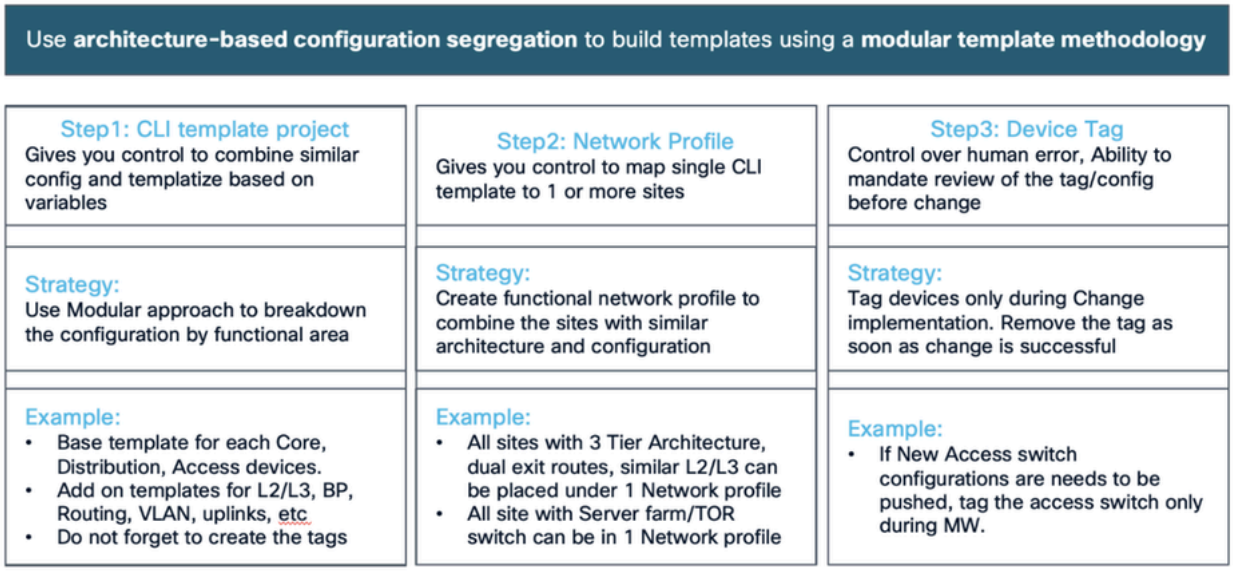


図1：例を使用したベストプラクティス

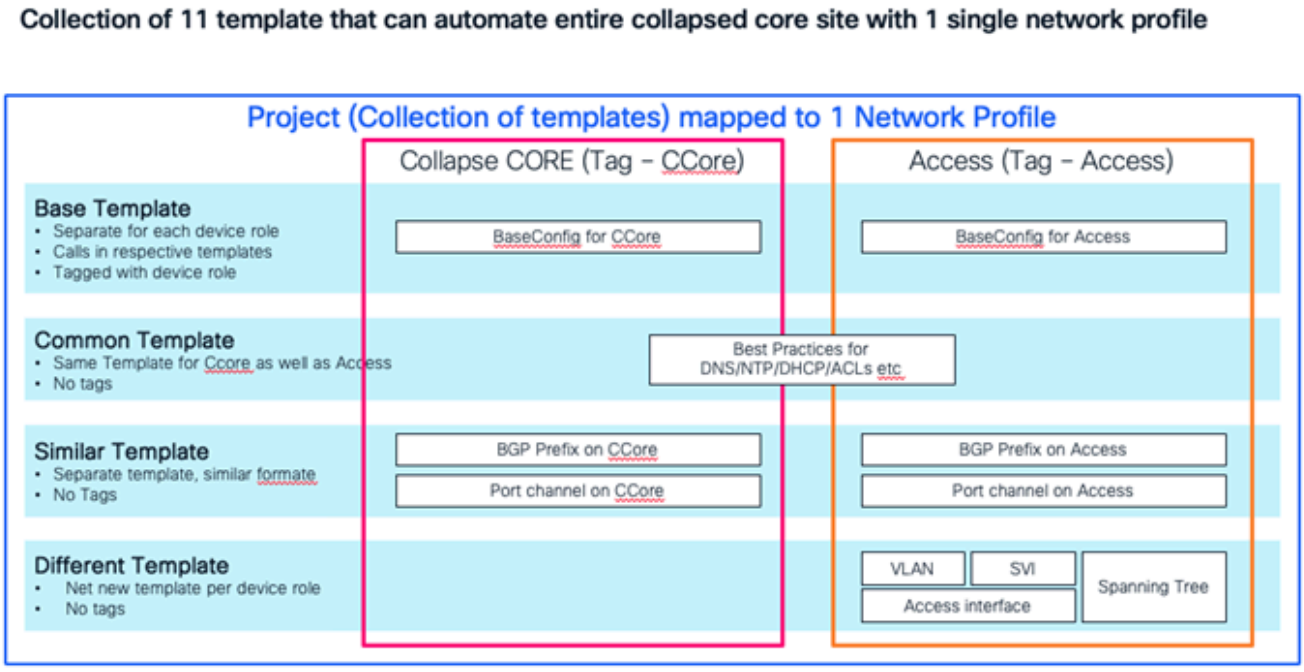


図2：コラプストコアテンプレートの例

テンプレートの操作に関するベストプラクティス

自動設定用のモジュラテンプレート設計

Cisco Catalyst Centerでデバイス設定を自動化する場合は、すべての設定を単一のモノリシックテンプレートに埋め込まないでください。代わりに、次のモジュール型アプローチを採用します。

小規模で目的に応じたテンプレート（モジュール）を参照する基本テンプレートを作成します。

設定を論理モジュール（インターフェイス設定、ルーティングプロトコル、セキュリティ機能など）に分割します。

この構造によってアップデートの効率が向上します。特定のモジュールに対する変更は、そのモジュールが使用される場所に自動的に反映され、エラーと複雑さが大幅に軽減されます。

例：ブランチデバイスのモジュラ設定

ブランチデバイスの設定を自動化していると仮定します。

基本テンプレート：

主な設定領域用のモジュールテンプレートへの参照が含まれます。

必要に応じて変数を各モジュールに渡してカスタマイズします。

モジュールテンプレート：

interface_settings：インターフェイス設定を管理します。

routing_protocols:OSPF、EIGRP、またはBGPの設定が含まれます。

security_features:ACL、ファイアウォールルール、またはその他のセキュリティポリシーを定義します。

```
{% 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) }}
```

基本テンプレート構造の例：

この構造では、ルーティングまたはセキュリティ設定に対する変更はそれぞれのモジュールで行うだけで、変更は基本テンプレートが使用されている場所に即座に反映されます。これにより、すべてのブランチルータで設定の管理が容易になり、一貫性が向上します。

ここでは、プロジェクト名はBranchで、他の3つの異なるモジュールがprojectの下で定義されています。これらはすべて基本テンプレートに組み合わされています。

テンプレート内の変数の最小化

複雑さとエラーを減らすために、テンプレート内の変数の数を最小限に抑えます。特に大規模なネットワーク全体では、導入を簡素化する変数が少なくなるため、プロセスの効率と一貫性が向上します。

テンプレートのデバイスタグの使用

Cisco Catalyst Centerのデバイスタグ（場所、ロール、サイトなど）を活用して、動的でスケーラブルなJinja2テンプレートを作成します。これらのタグによって条件付きロジックが有効になり、適切なデバイスに正しい設定が適用されます。このアプローチにより、さまざまなネットワーク環境でエラーを最小限に抑え、テンプレート管理を簡素化できます。

可能な場合は固定値をハードコードする

静的な値をハードコードすると、テンプレートが簡素化され、導入の効率が向上します。一般的な例としては、DNS、NTP、またはSyslogサーバのIPアドレスがあります。これらは通常、デバイス間で一貫性を維持します。同様に、アクセススイッチで標準のVLAN IDを使用すると、これらの値をハードコードでき、ばらつきを減らして導入を迅速化できます。

2段階アプローチの採用：Day 0およびDay Nテンプレート

Cisco Plug and Playなどのサービスを使用してデバイスをオンボーディングする際には、次

の2段階のテンプレート戦略を使用します。

Day 0テンプレート：基本設定をプッシュして、デバイスがCisco Catalyst Centerと通信できるようにします。

N日目のテンプレート：デバイスが到達可能になったら、高度な機能と設定を導入します。

ベストプラクティスにより、効率的でスケーラブルなテンプレートが可能になり、シスコのキャンパスネットワークの導入が簡素化されます。

Jinjaテンプレートマクロの空白コントロール

神社言語を使用してテンプレートを作成する場合、特にマクロ内で動的コンテンツをレンダリングする場合は、空白と改行を慎重に処理することが重要です。蓄積された空白または意図しない改行は、生成される出力でフォーマットの問題を引き起こし、ダウンストリーム処理で誤った解釈や工ラーを引き起こす可能性があります。これに対処するために、Jinjaは空白を制御するための構文を提供しています。デリミタ（{{- ... -}}または{%- ... -%}）の中に直接マイナス記号(-)を入れると、式の前後の空白が取り除かれます。たとえば、{{item[1]}}を{{- item[1] -}}に置き換えると、マクロの実行時に余分なスペースや改行が削除されます。この方法は、テンプレートのス二ペットに示すように、リストを繰り返したり、設定ファイルを生成したりする際に特に役立ちます。このようなシナリオでは、クリーンで予測可能な出力を維持するために、常に空白の制御を適用することをお勧めします。

例（推奨される使用方法）:

```
{% for item in wildcard_list %}
   {% if item[0] == prefix -%}
      {{- item[1] -}}
   {%- endif %}
{%- %}に対して終了
```

3層アーキテクチャ

このホワイトペーパーでは、最初にアクセススイッチからコアスイッチに至るテンプレートを開発し、各層の要件を概説します。

アクセスレイヤスイッチ

アクセススイッチはプラグアンドプレイを使用してオンボーディングされるため、Day 0テンプレートが必要です。Catalyst Centerでのプラグアンドプレイプロセスの詳細については、次のリンクを参照してください。

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

前述したように、Catalyst CenterはVelocityとJinja2の両方のテンプレート言語をサポートしています。本書では、柔軟性の高いJinja2を使用してテンプレート構造を説明します。アクセスレイヤスイッチ設定は、Day-0およびDay-Nテンプレートを使用して導入できます。

基本的なDay 0テンプレートは構造化できます。手順1:

手順1：テンプレートの定義

```
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
!
```

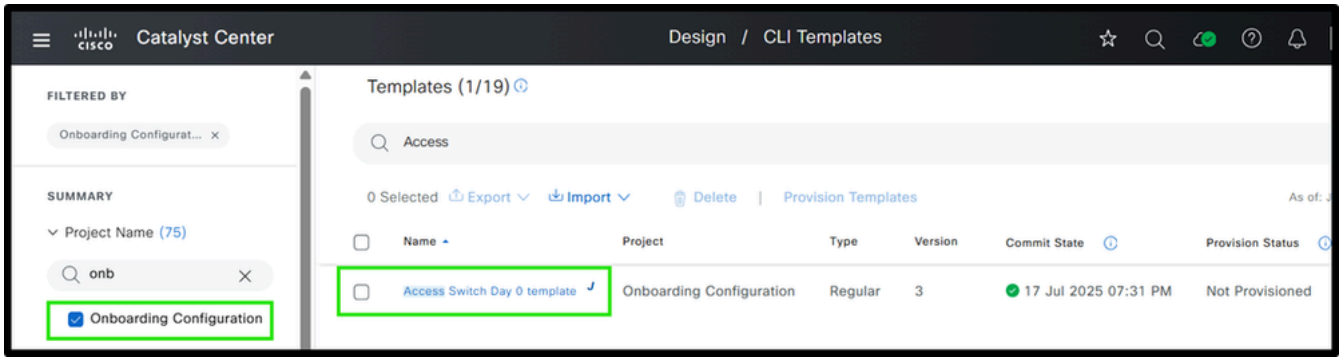
手順1：テンプレートの定義

このテンプレートでは、ブランチ内のすべてのスイッチが同じ管理VLANサブネットマスクを共有するため、ユーザ名、パスワード、イネーブルシークレット、サブネットマスクなどの定数をハードコードすることで、設定が簡素化されます。ただし、管理IPアドレスはスイッチごとに一意であり、変数として定義されます。このDay 0テンプレートを利用するDay Nテンプレートには、包括的なテンプレート構造が必要です。N日目のテンプレートでは、アクセススイッチの各機能は専用のモジュールで管理されます。たとえば、1つのモジュールでレイヤ2 VLANを処理し、別のモジュールでアップリンクとダウンリンクのアクセスインターフ

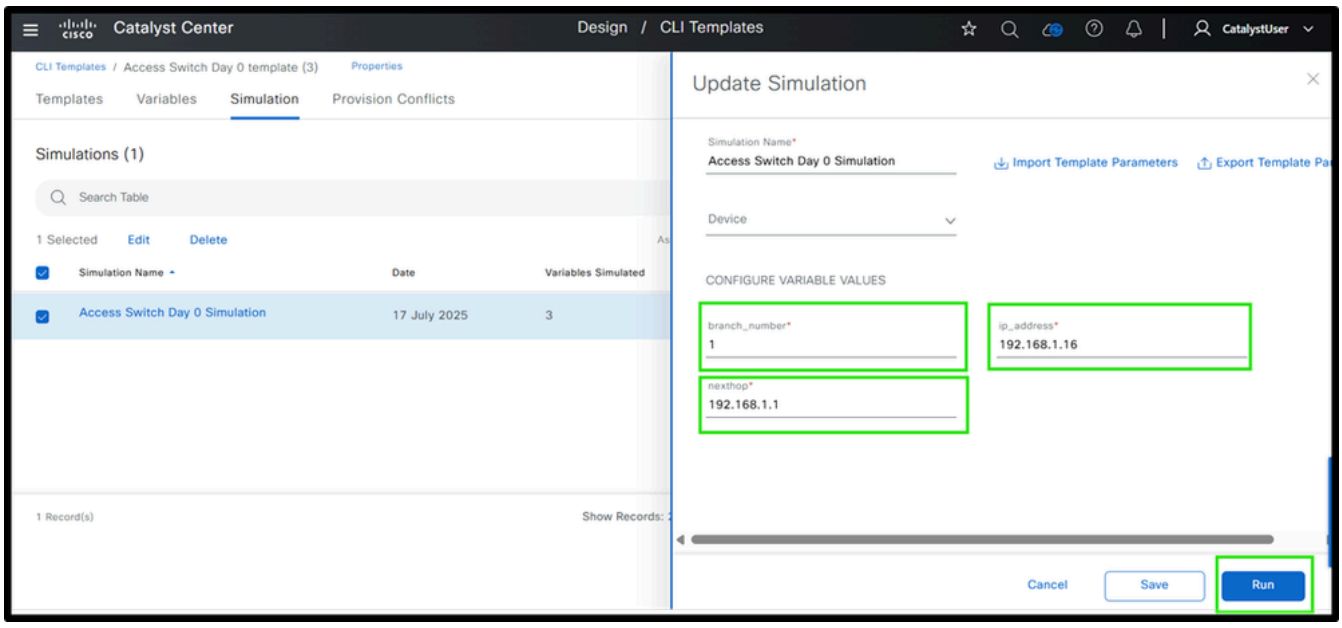
ェイスを管理し、別のモジュールでセキュリティの強化を重点的に行うなどです。

一貫したVLAN IDが推奨されますが、ブランチ番号に基づく式を使用して、さまざまなIDを動的に生成できます（たとえば、ブランチ1 = VLAN 113、ブランチ2 = VLAN 213）。 これにより、テンプレートを複数のブランチで再利用可能になります。同様に、ネクストホップIPは、接続されたディストリビューションクラスタに応じてブランチごとに異なる必要があるため、変数です。

手順2：シミュレーションを実行し、変数を指定する

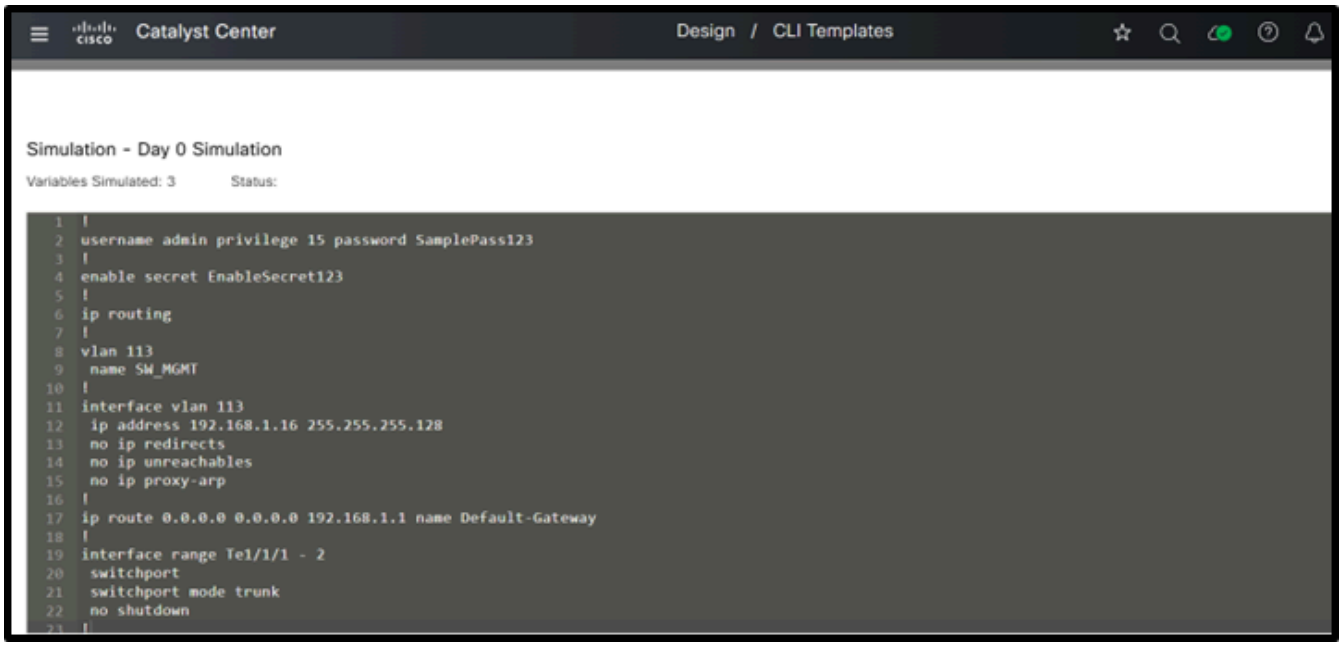


シミュレーションの入力と出力を含むアクセススイッチのDay 0テンプレート構造



例：シミュレーション入力

配置する前にテンプレートをシミュレートすることを常に推奨します。このスクリーンショットは、変数を入力した後の最終的な設定を示しています。



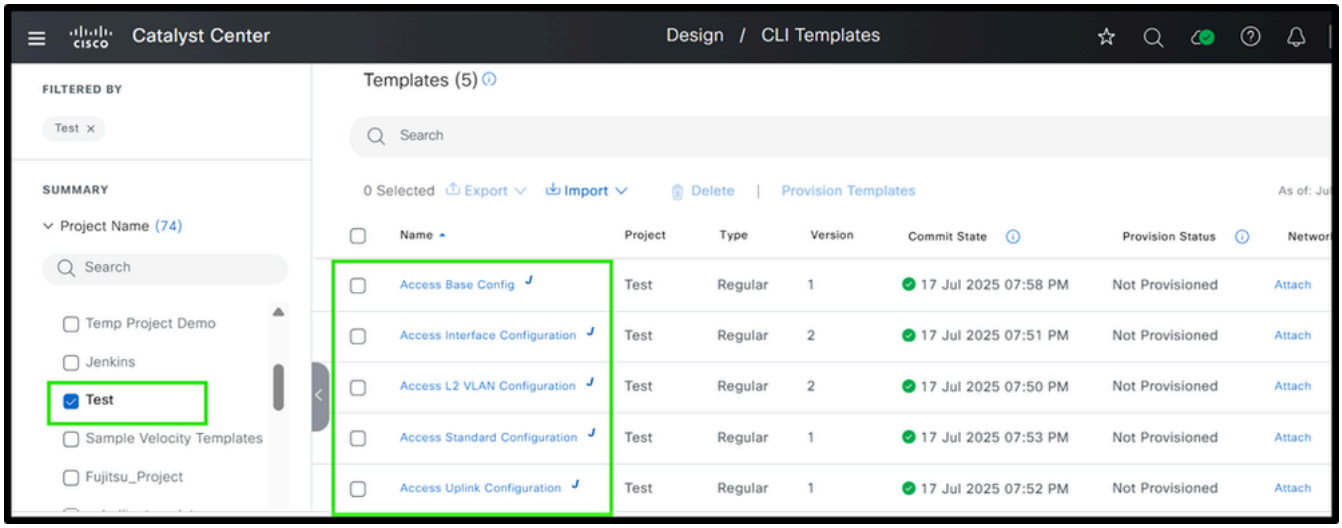
値を入力した後の初期設定

次に、モジュラDay Nテンプレートを作成する方法を見てみましょう。

アクセススイッチの設定はさまざまなモジュールに分割でき、すべて1つの基本モジュール内に組み合わせることができます。アクセススイッチの基本テンプレートは、次のように構成されています。

基本テンプレートとそのモジュールはどちらも、Cisco Catalyst Centerの「Test」という名前のプロジェクト内に作成されます。

手順1：基本テンプレートを含むさまざまなテンプレートの定義



アクセススイッチDay Nテンプレート構造

手順2：さまざまなモジュールを定義する

アクセスベースの設定：

このスクリーンショットは、基本設定の例を示しています。

```
{% 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) }}
```

アクセスベースの設定

このモジュラ設定テンプレートは、VLAN設定、アップリンクインターフェイス設定、アクセスインターフェイス設定、および標準設定の4つの部分で構成されています。branch_numberとis_poeの2つの変数のみを使用するため、管理が簡単です。

branch_numberは、Day 0テンプレートに示されているようにブランチ固有のVLAN IDを計算します。is_poeは、アクセススイッチがPoEスイッチか非PoEスイッチかを判断します。これらの変数はプロビジョニング中に提供され、モジュールに渡されて正しい設定が作成されるため、作業が軽減され、効率が向上します。

次に、各モジュールを見直して、構成全体の特定の部分の生成にそれらがどのように貢献するかを確認します。

アクセスL2 VLANの設定

```
{% 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 %}
```

アクセスL2 VLANの設定

このモジュールは、すでに説明したように、ブランチ番号に基づいてVLANを作成します。データおよび音声VLANは、PoEをサポートしているかどうかに関係なく、すべてのスイッチで作成されます。AP管理VLAN (ブランチ1の場合は114など) は、is_poeが「Yes」に設定されている場合、つまりスイッチがPoEをサポートしている場合にのみ作成されます。
is_poeが「No」の場合、AP管理VLANはスキップされます。これは、非PoEスイッチではアクセスポイントがサポートされないためです。これは、if条件を使用して管理されます。

```
{% 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 %}
```

アクセスインターフェイス設定

このモジュールは、アクセスインターフェイス設定を処理し、前述のPoEスイッチと同じアプローチを使用します。is_poe変数が「Yes」の場合、スイッチがPoEスイッチであることを意味し、最初の6つのポート(1 ~ 6)にAP管理VLANを設定する必要があります。それ以外の場合は、最初の6つのポートをユーザアクセスポートとして設定する必要があります。

スイッチが24ポートモデルであると仮定すると、スイッチがPoEであるかどうかにかかわらず、残りのポート(7 ~ 24)は常にユーザアクセスポートとして設定されます。

インターフェイスの範囲は標準化され、入力変数としては使用されなくなりました。これは、テンプレート内の変数の数を最小限に抑えるためのベストプラクティスと考えられています。さらに、このモジュールにはcommon_access_settingsという名前のマクロが含まれており、繰り返し行われる設定を統合することでテンプレートサイズを最小化します。このマクロは単にインターフェイス設定内で呼び出されるため、何度も指定する必要はありません。

注：このテンプレートは、すべてのアクセスインターフェイスに同じ説明を適用します。各インターフェイスに固有の説明が必要な場合は、個別のPythonスクリプトまたは同様の自動化ツールを使用してインターフェイスをプッシュすることをお勧めします。

アップリンクインターフェイスの設定を生成するモジュールを確認します。

```
{% 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 %}
```

アクセスアップリンクの設定

このモジュールは、アップリンクインターフェイスの設定を生成し、VLANプルーニングを処理します。スイッチがPoEをサポートしている場合、AP管理VLANは許可されるVLANのリストに含まれます。サポートしていない場合は除外されます。このロジックは、前述のように、コード内のif条件を使用して管理されます。

最後のモジュールを見直して、ベストプラクティスやセキュリティの強化を含む標準設定を実証します。

⚠ 注意：これは説明のみを目的としており、設定は特定の要件によって異なる可能性があるため、実際のネットワーク設定のリファレンスとしては使用しないでください

```
{% 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
```

パート1：標準構成へのアクセス

```
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 %}
```

パート2：標準構成へのアクセス

このモジュールでは、ベストプラクティス、セキュリティの強化、およびセキュアなデバイス管理の主要機能を組み込んだ標準構成を生成します。ほとんどの値はブランチ間で一貫性を持たせるためにハードコードされています。ただし、branch_numberは例外です。この番号は、各ブランチ内のスイッチの管理VLANを計算するために使用され、複数の設定で送信元インターフェイスとして機能します。

手順3：スイッチを設定する前にシミュレーションを実行します。基本設定のみをシミュレートする必要があります。

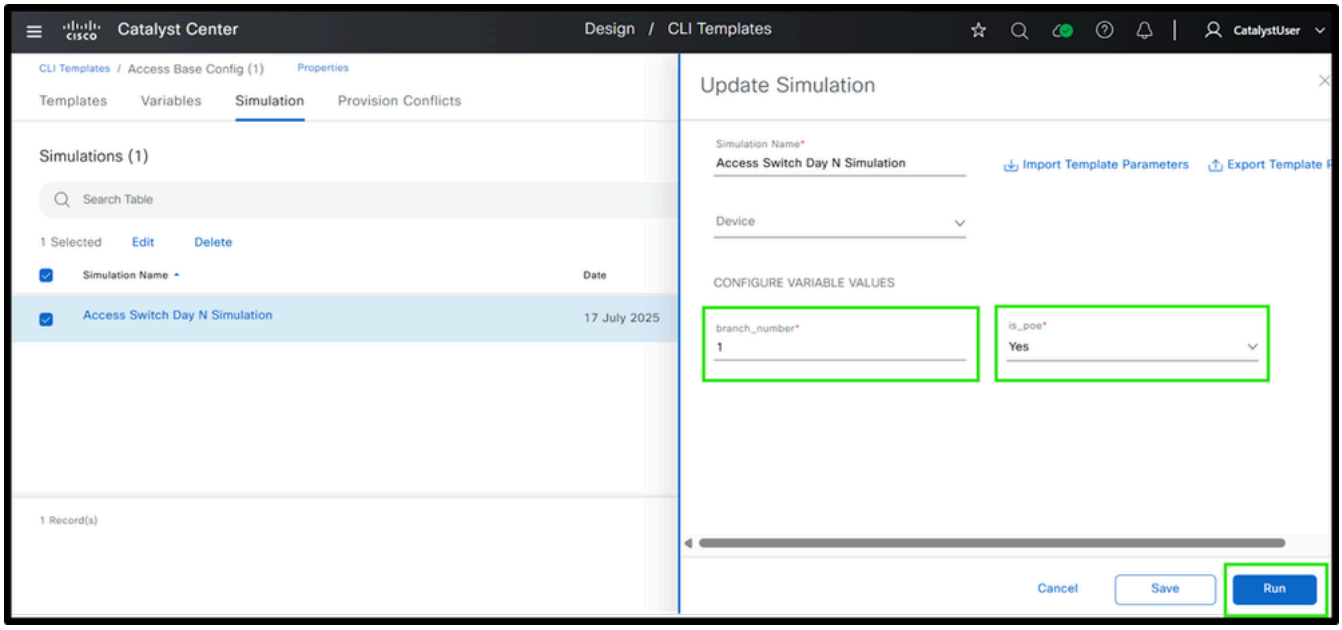
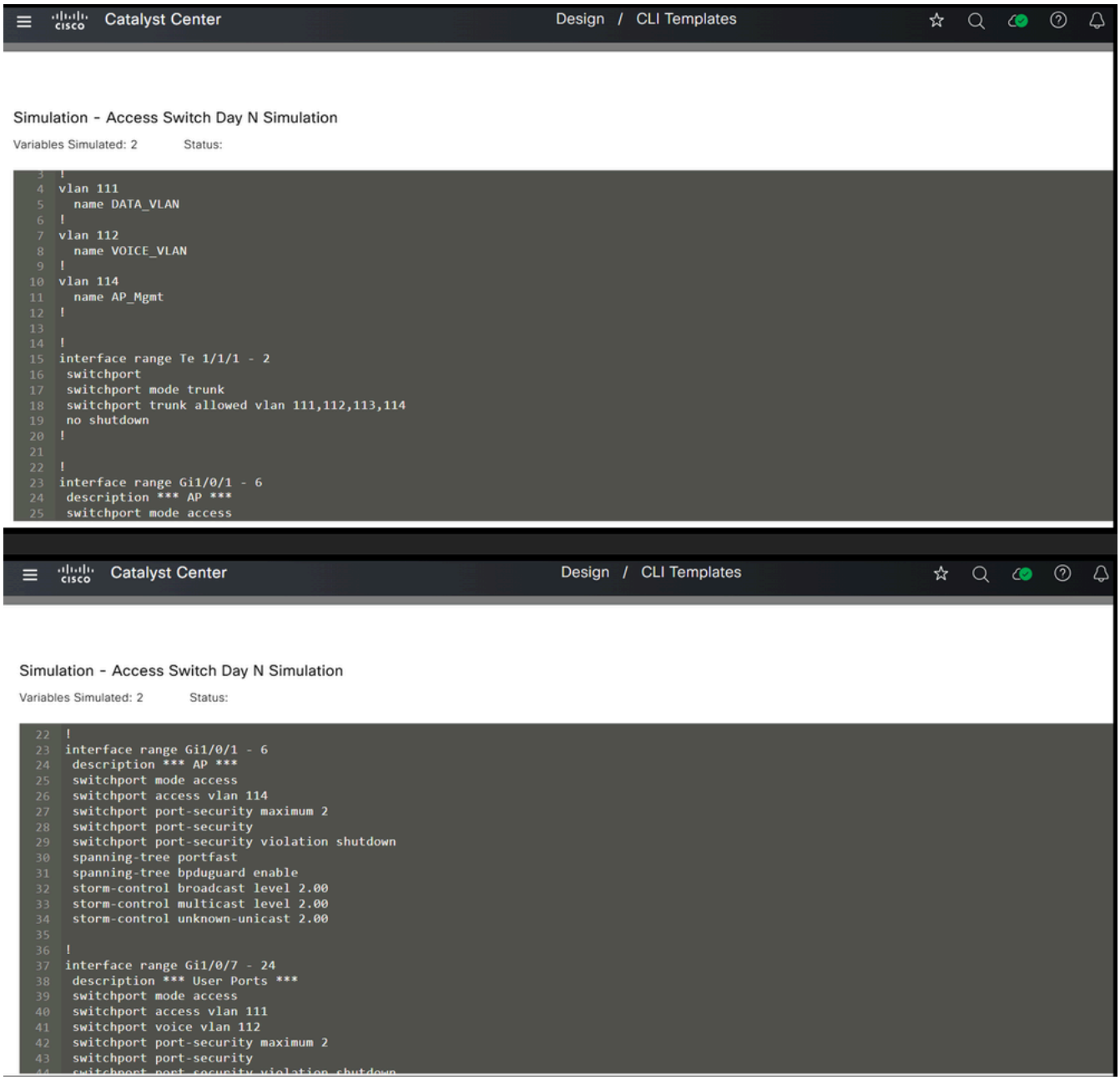


図7：アクセススイッチDay Nテンプレートシミュレーションの入力と出力



シミュレーション

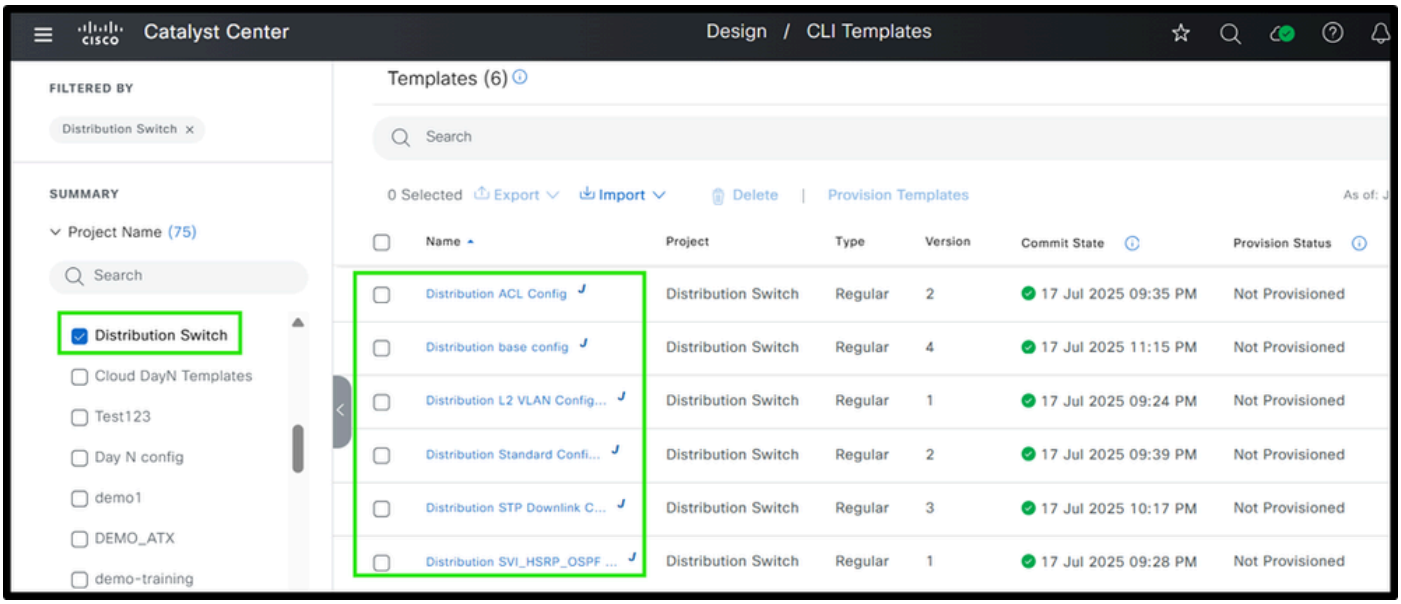
このようにして、アクセスレイヤでテンプレートを使用して設定を生成できます。

次に、ディストリビューション層のデバイスを見て、それらにどのようにテンプレートを適用できるかを確認します。

ディストリビューション層スイッチ

次に、ディストリビューションスイッチ用のモジュラテンプレートを設計します。基本テンプレートとそのモジュールは、Cisco Catalyst Centerの「ディストリビューションスイッチ」プロジェクトの一部です。

ステップ1：ディストリビューションスイッチテンプレートの構造



例：ディストリビューションテンプレート

手順2：各モジュールを定義します。

提供されている基本設定では、各モジュールおよびすべてのモジュールが参照されます。

```
{% 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：ディストリビューションベーステンプレートモジュール

アクセススイッチと同様に、すべてのテンプレートは「ディストリビューションスイッチ」プロジェクト内で作成され、基本テンプレート内で参照されます。一部のテンプレートはアクセススイッチで使用されるテンプレートと同じですが、このセクションでは、ディストリビューションスイッチに固有の違いについて説明します。モジュール「ディストリビューションL2 VLANの設定」は、アクセススイッチに関して前述したものと同じです。この情報については、『アクセスL2 VLANコンフィギュレーションモジュール』を参照してください。変数に指定された入力値に基づいて、必要なVLANを生成します。

ここで、「ディストリビューションSTPダウンリンク設定」モジュールを確認します。このモジュールは、ディストリビューションスイッチのスパニングツリーおよびアップリンク設定の生成を処理します。

```
{% 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 %}
```

ディストリビューションSTPダウンリンク設定

ここでは、Jinja2マクロ機能が使用されています（この機能は基本モジュールで参照されています）。そのため、この構造はモジュール型アプローチの構築に役立ちます。

このモジュールは、「branch_number」に基づいて、およびスイッチがPoE対応であるかどうかによって、スパニングツリープロトコル(STP)とダウンリンクインターフェイスを設定します。「branch_number」変数は各ブランチに一意のベースVLANを生成するために使用され、アクセススイッチについてすでに説明した方法と同様に、別個のVLANを保証します。スイッチがPoE対応の場合(「is_poe」==「Yes」)、AP管理VLANなどの追加VLANがリストに追加されます。「distribution_number」変数はSTPプライオリティを決定し、Distribution 1の設定は4096（優先ルートブリッジにする）、セカンダリ同報スイッチの設定は8192です。最後に、適切なVLANがトランクインターフェイスに適用され、スイッチがPoE対応であるかどうかによって、関連するVLANのみが許可されます。

次に、「Distribution SVI_HSRP_OSPF Config」モジュールを確認します。このモジュールは、効率的なネットワークルーティングおよび冗長性のためのSVI、HSRP、およびOSPFのセットアッ

プに重点を置いています。

```
{% 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 %}
```

ディストリビューションSVI_HSRP_OSPFの設定

このモジュールDistribution_SVI_HSRP_OSPF_Configは、ディストリビューションスイッチのSVI、HSRP、OSPF、およびアップリンクインターフェイスの設定に役立ちます。この例では、

データサブネットのSVIに焦点を当てていますが、音声や管理など他のSVIにも同じ方法を使用できます。

データサブネットのIPアドレスの計画がすでに行われている場合は、branch_number変数とdistribution_number変数に基づいて、各SVIのIPアドレスを自動的に計算できます。たとえば、Branch 1のサブネットが172.17.0.0/20、Branch 2のサブネットが172.17.16.0/20、Branch 3のサブネットが172.17.32.0/20の場合、ゲートウェイIPは172.17.x.1（xはブランチ番号）になります。 最初のディストリビューションスイッチの2番目のIPは172.17.x.2、2番目のディストリビューションスイッチの3番目のIPは172.17.x.3です。これにより、IPアドレスが自動的に計算され、エラーが減り、プロセスが簡素化されます。

ループバックインターフェイスには、変数loopback_ipからIPが割り当てられます。この変数はOSPFルータIDとして機能し、ネットワーク全体で安定した一貫性のあるルーティングを保証します。OSPF設定では、このループバックIPがルータIDとして使用され、関連するインターフェイスがOSPFエリア0に追加されます。HSRPでは、プライオリティ値が設定されます。最初のディストリビューションスイッチは255、2番目のディストリビューションスイッチは250で、フェールオーバーが適切に行われるようにします。また、セキュリティを強化するために、キーチェーン(HSRP_KEY)を使用してHSRP認証を設定します。

設定をクリーンで管理可能な状態に保つために、一部の値はハードコードされています。たとえば、サブネットマスク(255.255.240.0)と特定のHSRP設定（バージョンやBFDなど）がすべてのブランチで同じであるため、変数の数が減ります。これにより、設定が簡単になり、適用が容易になり、ミスが発生する可能性が低くなります。最後に、アップリンクインターフェイスにIPが設定され、スイッチ間の適切なルーティングのためにOSPFエリア0に追加されます。このアプローチにより、設定プロセスの管理が容易になり、エラーが発生しにくくなるだけでなく、さまざまなブランチに柔軟に対応できます。

ここで、ディストリビューションレイヤでセグメンテーションを提供する「ディストリビューションACLの設定」モジュールを確認します。

```
{% 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 %}
```

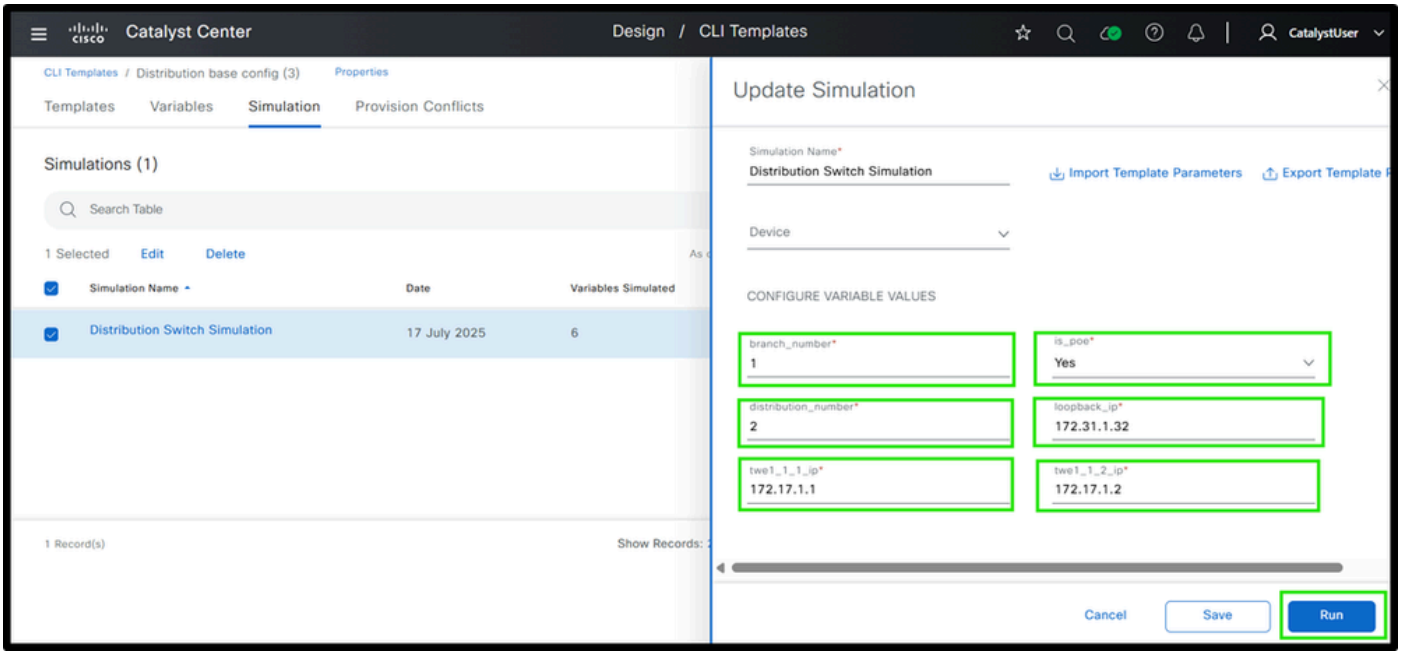
配布ACLの設定

このモジュールでは、Jinja2テンプレートを使用して、ディストリビューションレイヤでのセグ

メンテーションについて説明します。branch_number変数を使用してサブネットアドレスを動的に計算するため、自動化されたスケーラブルなACL設定が可能になります。ブランチごとに、ACLはサブネット1(172.17.X.0)とサブネット2(172.16.X.0)の間の通信を、これらの範囲のIPトラフィックを拒否することによってブロックします。また、マルチキャストアドレス239.255.255.250へのトラフィックは拒否され、その他のトラフィックはすべて許可されます。VLANインターフェイスはブランチ番号に基づいて動的に割り当てられ、ACLはそのインターフェイスの着信に適用されます。この自動化されたアプローチにより、ブランチごとの効果的なセグメンテーションが実現し、手動による設定エラーが削減され、ネットワークポリシーの適用が簡素化されます。

最後に、最後のモジュール「ディストリビューション標準の設定」は、「アクセス標準の設定」モジュールで説明したものと同じです（詳細については、そのセクションを参照してください）。 これには、ベストプラクティス、セキュリティの強化、および安全なデバイス管理のための主要機能が含まれます。唯一の違いは、送信元インターフェイスにあります。アクセススイッチテンプレートではVLAN {{ branch_number * 100 + 13 }}として定義されていますが、ディストリビューションスイッチ設定ではLoopback0としてハードコードできます。

手順3：設定を展開する前にシミュレーションを実行します。



(1)ディストリビューションスイッチテンプレートシミュレーションの入出力

(2)ディストリビューションスイッチテンプレートシミュレーションの入出力

(3)ディストリビューションスイッチテンプレートシミュレーションの入出力

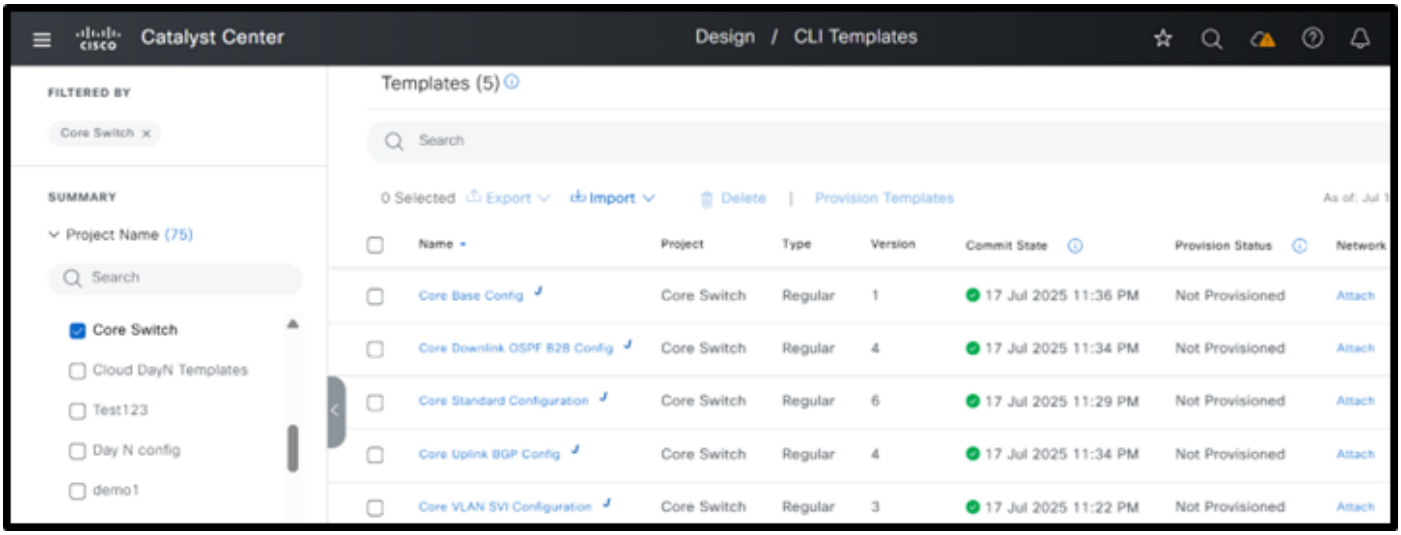
(4)ディストリビューションスイッチテンプレートシミュレーションの入出力

このようにして、ディストリビューションレイヤでテンプレートを使用して設定を生成できます。次に、コア層のデバイスを見て、そこにテンプレートを適用する方法を確認します。

コア層スイッチ

次に、コアスイッチ用のモジュラテンプレートを設計します。基本テンプレートとそのモジュールは、Cisco Catalyst Centerの「コアスイッチ」プロジェクトの一部です。手順1の基本テンプレートを参照してください。

手順1：さまざまなコアスイッチ構造の定義



コアスイッチテンプレートの構造

手順2：さまざまなモジュールを定義する

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

コアベース構成

ほとんどのコアスイッチ設定はすべてのブランチで類似しているため、共通の値をハードコードできます。通常、IPアドレスのみが変更され、変数を使用して設定できます。各ブランチには通常、コアスイッチが2つしかないため、これらの変数の管理は簡単です。コアスイッチの数が多いブランチでも、その数はアクセススイッチやディストリビューションスイッチの数よりも少なくなります。そのため、アクセススイッチとディストリビューションスイッチの変数は大きな数で使用され、変数が多すぎると設定に時間がかかることがあるため、ベストプラクティスとして、これらの変数を最小限に抑えることの方が重要です。

次に、最初のモジュール「コアVLAN SVIの設定」から始めます。 この例では、コアスイッチはファイアウォールの背後に配置され、ファイアウォールとのeBGPピアリングを確立する必要があります。このモジュールは、eBGPピアリングとOSPFネイバーシップに必要なVLANと対応するSVIを生成します。ファイアウォールは、アクティブ/スタンバイ設定で動作することを前提としています。

```
{% 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 %}
```

コアVLAN SVIの設定

このモジュールは、すでに説明したように、OSPFおよびBGPネイバー関係を確立するために必要なVLANおよび関連するSVIを作成します。SVI IPアドレスを除くすべてのパラメータは、IPアドレッシング計画に一致するサブネットマスクを含めて、ハードコードされています。この方法は、変数を制限し、設定エラーの可能性を減らすのに役立ちます。

次に、「コアダウンリンクOSPF B2B設定」モジュールを確認します。このモジュールは、ダウンリンクインターフェイス、OSPF、およびコアスイッチ1とコアスイッチ2間のバックツーバックリンクの設定を生成します。

```
{% 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 %}
```

コアダウンリンクOSPF B2B設定

前のモジュールと同様に、このモジュールのほとんどの値は変数の数を最小限に抑えるためにハードコードされています。ループバックインターフェイスとダウンリンクインターフェイスのIPアドレスだけが可変です。また、バックツーバックポートチャネルとVLANは、異なるブランチ間で標準化されています。

次に、BGP設定を生成し、ファイアウォールに接続されたアップリンクを管理する「コアアップリンクBGP設定」モジュールについて確認します。

```
{% 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 %}
```

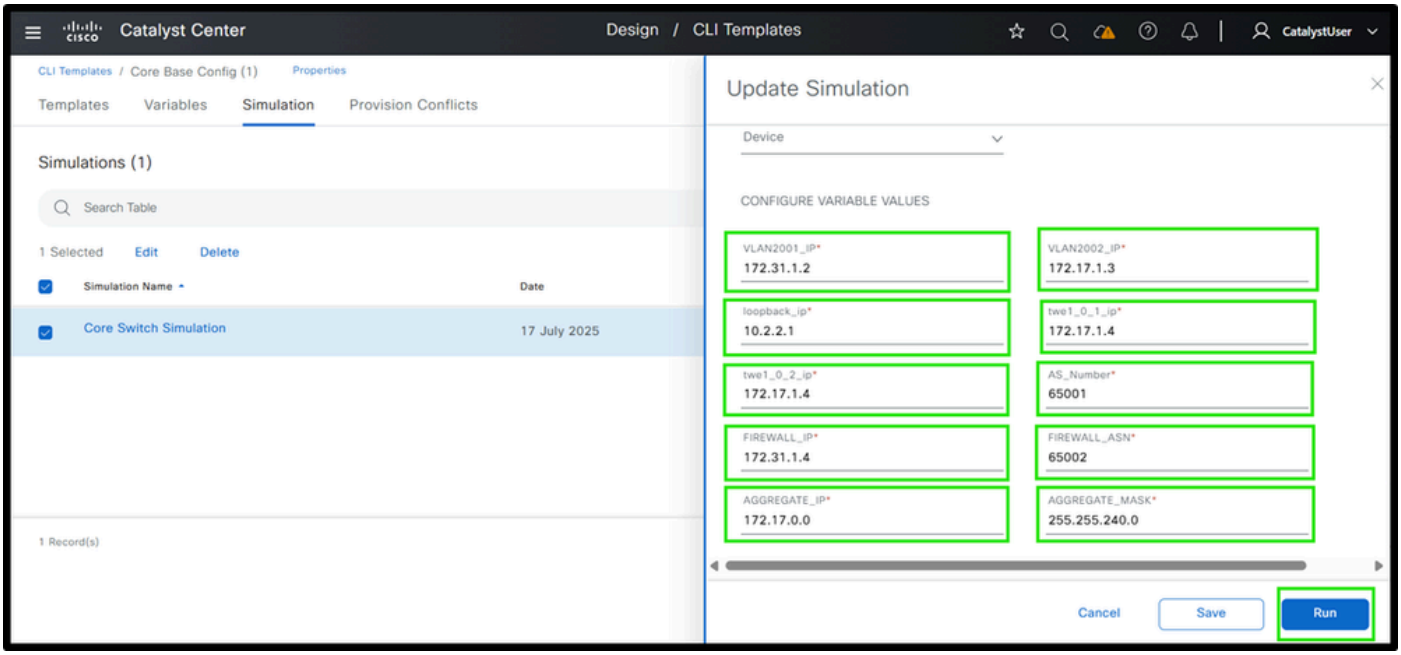
コアアップリンクBGPの設定

このモジュールは、ファイアウォールとのeBGPネイバー関係を確立するために必要なBGP設定を生成します。上に示したように、ほとんどの値はハードコードされています。これは、異なる

ブランチ間で一貫性が維持されるためです。IPアドレスとAS番号だけが入力変数として取得され、必要な設定の生成に使用されます。これはブランチごとに異なる可能性があります。他のほとんどの設定は、変数の数を最小限に抑えるために標準化されています。ファイアウォールに接続されているアップリンクインターフェイスは、前のモジュールで生成された、eBGPネイバーシップに使用されるVLANとともに指定されます。

最後に、最後のモジュール「コア標準コンフィギュレーション」は、アクセス標準コンフィギュレーションで説明したものとほとんど同じです（詳細については、このセクションを参照してください）。 これには、ベストプラクティス、セキュリティの強化、および安全なデバイス管理のための主要機能が含まれます。ディストリビューションスイッチの設定と一致するように、このモジュールでも送信元インターフェイスをloopback0に設定でき、この値はハードコードできます。

手順3：シミュレーションの実行



(1)コアスイッチテンプレートシミュレーションの入出力

(2)コアスイッチテンプレートシミュレーションの入出力

(3)コアスイッチテンプレートシミュレーションの入出力

(4)コアスイッチテンプレートシミュレーションの入出力

(5)コアスイッチテンプレートシミュレーションの入出力

これで、3層アーキテクチャのテンプレート設計の詳細な説明が完了し、各モジュールの構造と設定の概要が示されます。

これらのモジュールはすべて、前述のベストプラクティスを利用しています。

注：コラプストコアアーキテクチャ用のテンプレートを設計する場合は、3層アーキテクチャの説明を参照してください。テンプレートの構造は変わりませんが、これまでコア

層とディストリビューション層で別々に実装されていた機能が、コラプストコア層で統合されました。同じモジュラテンプレートアプローチを使用することもできます。この場合、基本テンプレートを作成し、そのテンプレート内の関連モジュールを参照します。

要約

従来の3層キャンパスアーキテクチャは、コア、ディストリビューション、およびアクセスレイヤにわたる広範な手動設定に依存していることが多い。このアプローチは時間がかかるだけでなく、人的ミスも起こりやすくなります。自動化や一元管理が行われないと、運用オーバーヘッドが大幅に増加し、最新の動的なキャンパスネットワークを効果的に拡張および管理することが困難になります。Catalyst Center CLIテンプレートを使用すると、従来のLANネットワークの機能設定を自動化できます。デバイスのプロビジョニングを行う際は、モジュラアプローチを利用することが重要です。モジュールは、異なるレイヤで使用されるさまざまな機能に基づいています。最後に、これらすべてのモジュールを基本モジュールにバインドします。

実施要請

このホワイトペーパーでは、スイッチ設定を標準化し、3層コアアーキテクチャとコラプストコアアーキテクチャの両方でネットワーク運用を最適化するためのベストプラクティスとして、モジュラテンプレート方式を採用することを推奨します。

- モジュール型テンプレートを実装することで、ネットワークチームは次のことが可能になります。
- 一貫性のある繰り返し可能な構成手法により、運用効率を向上させます。
- 人的エラーを最小限に抑え、トラブルシューティング時間を短縮します。
- 成長と進化するビジネスニーズに対応できる優れた拡張性を実現します。
- 多様な環境で構成の一貫性を確保します。

このアプローチにより、日常的な管理が合理化されるだけでなく、導入の迅速化、更新サイクルの合理化、セキュリティ要件やコンプライアンス要件への対応が可能になります。モジュール型テンプレートを採用することで、変化し続けるIT環境において俊敏性、復元力、および長期的な成功を実現するためのネットワークを位置付けることができます。

実践的なデモンストレーションについては、テンプレートの詳細を確認してください。YouTubeシリーズを参照してください。

1 Catalyst Centerでのテンプレートの作成および管理方法

https://youtu.be/SyUqEEcwye0

2 Catalyst Center CLIテンプレートでシステムバインド変数を使用する方法

https://youtu.be/gV1QBuHYJdo

著者

Naveen Kumar、カスタマーデリバリアーキテクト、シスコカスタマーエクスペリエンス

Risabh Mishra、コンサルティングエンジニア、シスコカスタマーエクスペリエンス

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。