# Comprendre les modèles sur Catalyst Center

## Introduction

Ce document décrit Cisco Catalyst Center et l'expérience des modèles de configuration pour les architectures de campus à trois niveaux ou à coeur de réseau regroupé.

## Informations générales

Ce document est destiné aux professionnels possédant une compréhension de base de Cisco Catalyst Center et une expérience des modèles de configuration. Il est particulièrement pertinent pour les personnes ayant travaillé ou envisageant de travailler avec des architectures de campus à trois niveaux ou à coeur de réseau regroupé.

L'objectif principal est d'aider les lecteurs à mettre en oeuvre et à automatiser des solutions de configuration et de gestion à l'aide de modèles dans Cisco Catalyst Center. En présentant des informations avancées, des techniques pratiques et des exemples concrets, ce document constitue une ressource pratique pour ceux qui cherchent à améliorer leurs compétences en matière d'infrastructure LAN et à optimiser les flux de travail grâce à l'automatisation et à la gestion basée sur des modèles.


Résumé analytique

À mesure que les réseaux d'entreprise continuent d'évoluer, le besoin d'une gestion évolutive, cohérente et automatisée n'a jamais été aussi grand. Cisco Catalyst Center offre une plate-forme centralisée, basée sur l'intention, qui simplifie la configuration, le provisionnement et l'assurance sur les réseaux de campus. Ce livre blanc explique comment les professionnels du réseau peuvent tirer parti de l'éditeur de modèles CLI et des fonctionnalités d'automatisation de Cisco Catalyst Center pour rationaliser les opérations réseau, réduire les erreurs de configuration et accélérer les déploiements sur des architectures centrales à trois niveaux et regroupées. Il détaille les meilleures pratiques pour la conception de modèles modulaires basés sur Jinja2, l'intégration de l'automatisation dans les workflows des jours 0 et N, et l'obtention d'une cohérence opérationnelle sur les couches coeur de réseau, distribution et accès. En adoptant les stratégies décrites dans ce document, vous pouvez transformer la gestion manuelle traditionnelle du réseau en un modèle agile, normalisé et automatisé, aligné sur la vision de mise en réseau basée sur les intentions de Cisco.

# Défis des réseaux de campus

À mesure que les réseaux de campus évoluent pour répondre aux exigences des entreprises modernes, ils sont confrontés à plusieurs défis majeurs :

2a. Complexité de la gestion du réseau

De nombreuses fonctions réseau sont toujours gérées manuellement, ce qui augmente le risque d'erreur humaine. Cela augmente non seulement les efforts de maintenance, mais met également à rude épreuve les ressources informatiques, en particulier avec des budgets statiques ou limités.

2 ter. Défis de déploiement et d'automatisation

L'intégration de nouveaux périphériques pour les réseaux filaires et sans fil est souvent longue et complexe, ce qui entraîne des retards de déploiement et une augmentation des frais d'administration.

2 quater. Gestion des images logicielles

Il est difficile de conserver une « image d'or » cohérente sur l'ensemble du réseau. De nombreux réseaux se retrouvent avec plusieurs systèmes d'exploitation pour les périphériques filaires et sans fil, ce qui entraîne des inefficacités et des difficultés de gestion.

2 quinquies. Configurations réseau incohérentes

Les variations dans les configurations réseau peuvent entraîner des problèmes de conformité et des inefficacités opérationnelles, ce qui complique la maintenance d'un réseau fiable et sécurisé.
2 sexies. Attentes croissantes des utilisateurs

Les utilisateurs exigent une connectivité ininterrompue et des expériences d'application transparentes, quel que soit leur emplacement ou leur appareil. Pour répondre à ces attentes, les réseaux doivent être résilients, intelligents et capables de s'adapter aux changements en temps réel.

Outre ces défis, les infrastructures LAN modernes sont confrontées à diverses autres complexités.

Simplifier les réseaux de campus avec Cisco Catalyst Center

Cisco Catalyst Center est une solution de gestion de réseau centralisée pour les réseaux de campus. Elle prend en charge le siège social, les filiales, les connexions filaires et sans fil, ainsi que les environnements IT/OT. Elle offre des options de déploiement flexibles, notamment des appareils physiques, des serveurs VMware ESXi ou le cloud AWS. Grâce à ses fonctionnalités complètes, Catalyst Center simplifie les opérations, améliore les performances et renforce la sécurité.

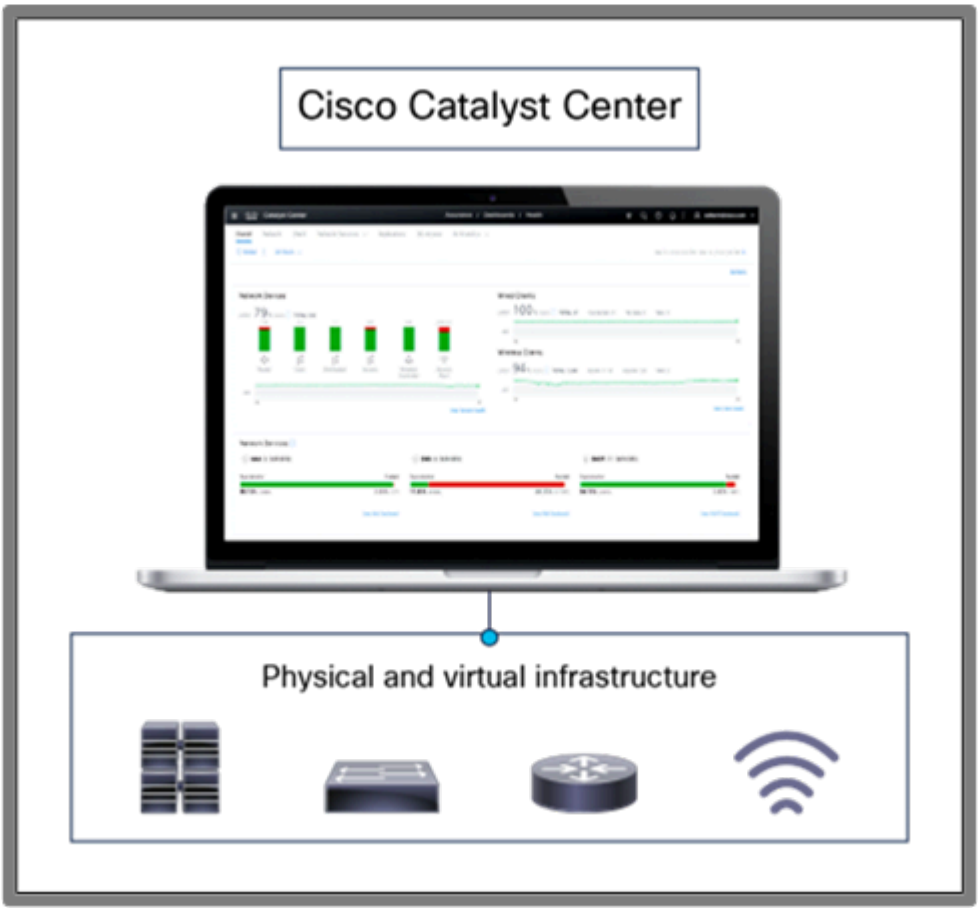



Figure 1 : Gestion de l'infrastructure avec Cisco Catalyst Center

## Caractéristiques et avantages clés

Cisco Catalyst Center (CC) offre des fonctionnalités avancées qui rationalisent la gestion et l'automatisation du réseau :

- Mise en service sans intervention (ZTP) : Automatise l'intégration des périphériques, réduisant ainsi les tâches manuelles et les délais de déploiement.

- Gestion des images logicielles (SWIM) : Garantit la cohérence des versions logicielles sur tous les périphériques grâce à des vérifications avant et après la mise à niveau pour éviter les problèmes.

Automatisation basée sur les intentions : Simplifie les déploiements en traduisant l'intention du réseau en configurations de périphériques pour les réseaux filaires et sans fil.

Automatisation LAN : Automatise l'adressage et le routage IP de couche 3 pour créer des topologies de bout en bout.

Automatisation du réseau sans fil : Des fonctionnalités telles que Plug and Play (PnP) permettent un provisionnement rapide des points d'accès sans fil.

Gestion de réseau hiérarchique : Permet des profils spécifiques au site (par exemple, SSID, paramètres RF, VLAN) pour des déploiements cohérents sur plusieurs sites.

Modèles CLI : Catalyst Center Template Editor permet aux administrateurs de créer et de gérer facilement des modèles de configuration basés sur CLI, ce qui permet un déploiement cohérent et efficace entre les périphériques.

Assurance : L'assurance permet d'avoir une surveillance centralisée des périphériques gérés via CC.

En plus de ces fonctionnalités, Cisco Catalyst Center offre de nombreuses autres fonctionnalités qui sortent du cadre de ce document. Ce document se concentre principalement sur la conception de modèles CLI à l'aide de Catalyst Center.

## Présentation générale de l'architecture de campus LAN avec Catalyst Center

Les réseaux de campus LAN traditionnels constituent l'épine dorsale de la connectivité d'entreprise, garantissant ainsi une communication fiable et évolutive pour les périphériques filaires et sans fil. Ces réseaux sont généralement conçus à l'aide de l'architecture à trois niveaux ou de l'architecture principale réduite, selon la taille et la complexité de l'entreprise.

### Architecture à trois niveaux

L'architecture à trois niveaux est un modèle de conception de réseau de base qui se compose de la couche coeur de réseau, de la couche de distribution et de la couche d'accès. Cette architecture offre évolutivité, hautes performances et gestion efficace du trafic. Reportez-vous à la présentation de chaque couche.

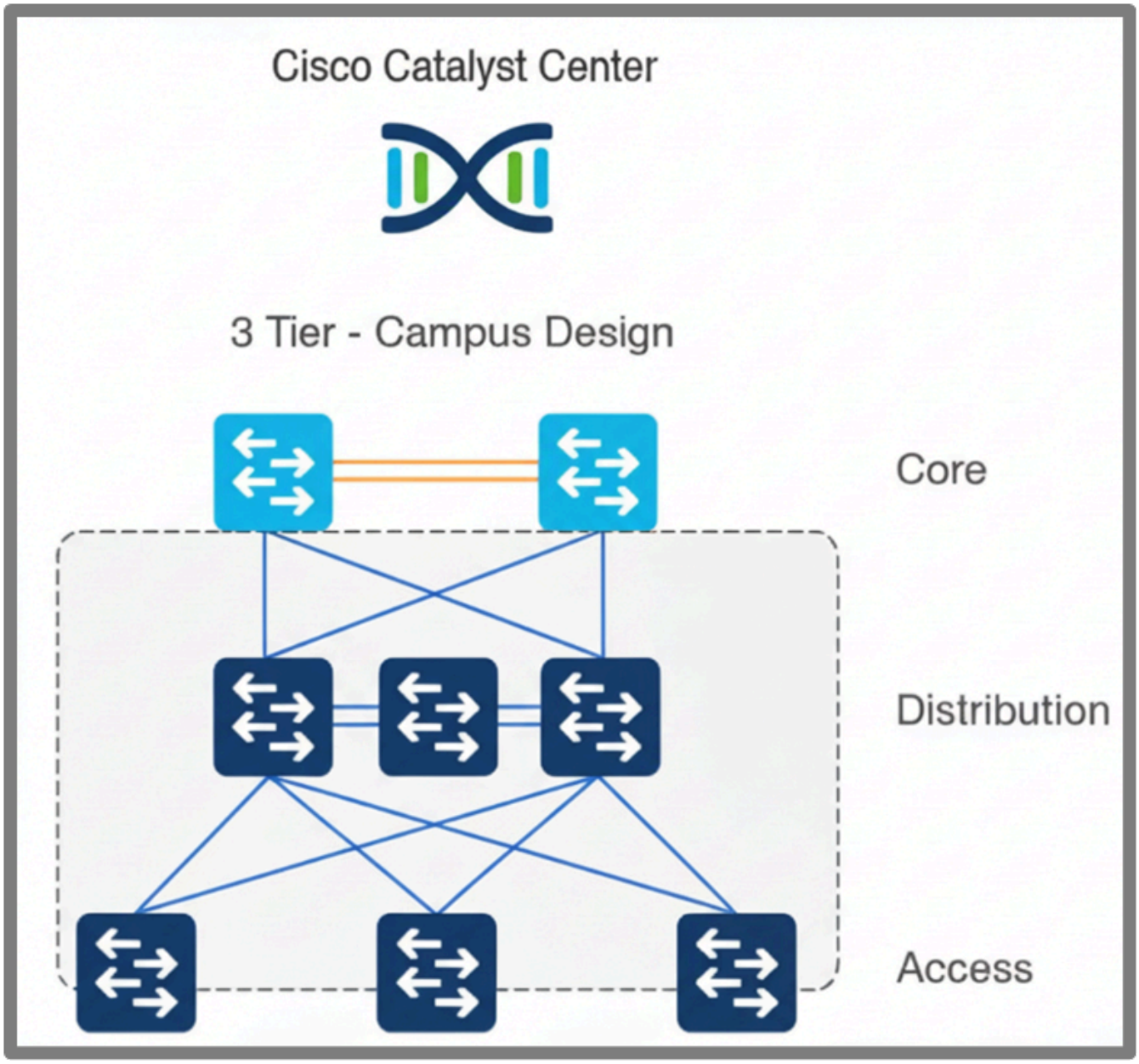



Figure 2 : Architecture de campus à trois niveaux

Couche coeur

La couche coeur de réseau sert de backbone au réseau, offrant une connectivité et une évolutivité à haut débit. Les configurations clés incluent les protocoles de routage ascendant et descendant (tels que OSPF et BGP), les politiques de routage, les configurations d'interface de liaison descendante et ascendante, le renforcement de la sécurité, etc

Couche de distribution

La couche de distribution relie les couches coeur de réseau et accès, en gérant l'agrégation du trafic, l'application des politiques et la redondance. Les configurations clés incluent HSRP/VRRP pour la redondance, STP pour la prévention des boucles, VLAN de couche 2 et

de couche 3, configurations d'interface de liaison ascendante et descendante, ACL pour la sécurité et le renforcement de la sécurité.

Couche D'Accès

La couche d’accès connecte les points d’extrémité au réseau, permettant ainsi un accès sécurisé et fiable. Les configurations clés incluent la configuration de l'interface d'accès, la configuration de l'interface de liaison ascendante, les VLAN de couche 2, les ACL pour restreindre l'accès au périphérique et le renforcement de la sécurité.

Architecture principale réduite

L'architecture coeur de réseau regroupée regroupe les couches coeur de réseau et distribution en une seule couche, ce qui réduit la complexité et les coûts tout en préservant les performances et l'évolutivité. Cette approche convient parfaitement aux réseaux de petite et moyenne taille qui ne nécessitent pas de couche coeur de réseau séparée. Reportez-vous à la présentation des couches de cette architecture.

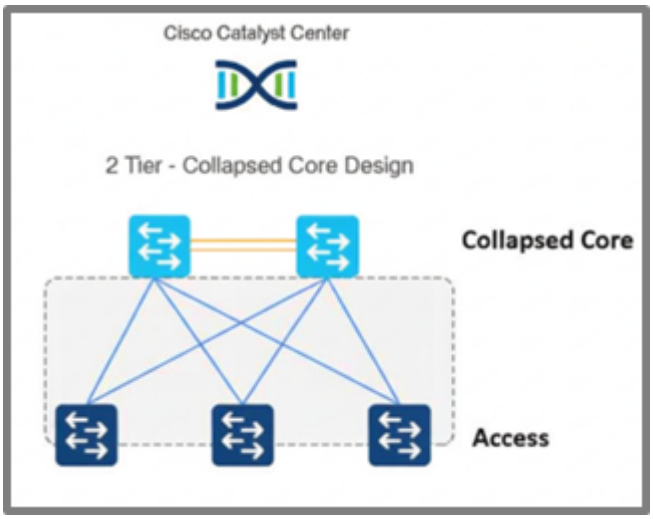



Figure 3 : Architecture de campus principale réduite

Couche coeur de réseau réduite

La couche coeur de réseau réduite combine les fonctions des couches coeur de réseau et de distribution, assurant la connectivité du réseau fédérateur, l'agrégation du trafic et l'application des politiques. Les configurations clés incluent les protocoles de routage ascendant et descendant (tels que OSPF et BGP), les politiques de routage, les configurations d'interface de liaison descendante et ascendante, BFD pour la détection de panne, le routage inter-VLAN utilisant des interfaces SVI, HSRP/VRRP pour la redondance de passerelle, STP pour la prévention des boucles et le renforcement de la sécurité. En exploitant les modèles de Cisco Catalyst Center, ces configurations peuvent être automatisées, ce qui garantit des

déploiements cohérents et efficaces.

Couche D'Accès

Comme décrit précédemment, la couche d’accès connecte les points d’extrémité au réseau, permettant ainsi un accès sécurisé et fiable. Les configurations clés incluent la configuration de l'interface d'accès, la configuration de l'interface de liaison ascendante, les VLAN de couche 2, les ACL pour restreindre l'accès au périphérique et le renforcement de la sécurité.

## Considérations relatives à la conception du modèle

Cette section explique comment concevoir des modèles dans Cisco Catalyst Center pour générer des configurations de périphériques. L'éditeur de modèles simplifie le provisionnement en permettant la création de modèles CLI réutilisables et en prenant en charge le déploiement dynamique de configurations adaptées à votre réseau. Catalyst Center prend en charge deux langues de modélisation : Jinja2 et Velocity. Ces langues facilitent la gestion de la configuration des périphériques.

Jinja est un langage de modélisation populaire et convivial, principalement utilisé avec Python pour générer du contenu dynamique comme HTML, XML ou d'autres formats textuels. Il permet d'incorporer des variables et des structures de contrôle (comme des boucles et des conditions) dans des modèles pour créer une sortie dynamique.

Apache Velocity est un moteur de modélisation Java qui utilise le langage VTL (Velocity Template Language) pour activer le contenu dynamique dans divers documents, notamment les pages Web, XML ou même le code source. Il fusionne les données des objets Java avec les modèles pour produire la sortie finale.

Ce document couvre uniquement les modèles Jinja2.

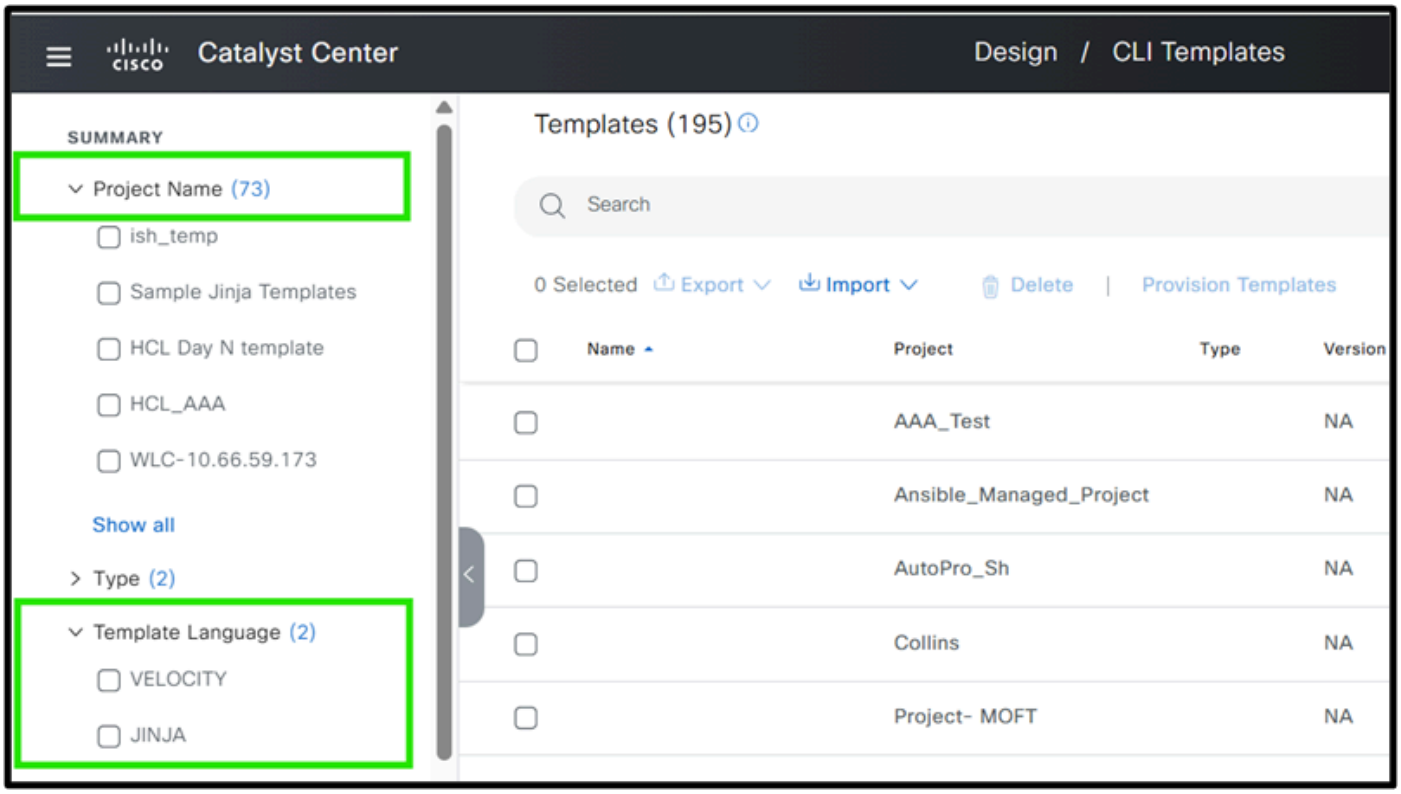



Figure 4 : Éditeur de modèle Cisco Catalyst Center

Dans ce document, nous utilisons Jinja2 en raison de sa flexibilité. Plutôt qu'une exploration en profondeur de Jinja2, l'accent est mis sur l'application pratique pour la conception de modèles. Pour plus d'informations sur la modélisation Jinja2 dans Catalyst Center, veuillez consulter le lien :

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

Avant de vous plonger dans les stratégies de conception de modèles pour un réseau de campus Cisco, il est important d'utiliser les meilleures pratiques clés pour garantir l'efficacité et la facilité de gestion lors de l'utilisation de modèles.

## Structure du modèle et meilleures pratiques / Lignes directrices pour la meilleure stratégie

Lors de l'automatisation de la configuration des périphériques réseau à l'aide de Cisco Catalyst Center, il est essentiel d'adopter des stratégies structurées et les meilleures pratiques. Ces étapes garantissent la cohérence, l'évolutivité et la facilité de gestion de votre infrastructure réseau.

Diviser la configuration par rôle de périphérique

Commencez par classer les périphériques en fonction de leur rôle dans la topologie du réseau. Les rôles courants incluent :

- Noyau
- Distribution
- Accès

Exemple : Un périphérique fonctionnant comme commutateur principal doit avoir des exigences de configuration différentes de celles d'un commutateur d'accès.

Compartimenter la configuration en blocs modulaires

Au sein de chaque rôle de périphérique, décomposez la configuration en blocs modulaires en regroupant des fonctionnalités ou des configurations similaires. Cette approche modulaire simplifie l'automatisation, le dépannage et les mises à jour futures.

Exemples pour un périphérique principal :

- Bloc de configuration OSPF
- Bloc de configuration BGP
- Bloquer les stratégies QoS

Identifier les blocs de configuration indépendants des rôles

Certains blocs de configuration s'appliquent universellement à tous les rôles de périphériques. L'identification et la normalisation de ces blocs garantissent des pratiques d'excellence et la cohérence sur l'ensemble du réseau.

Blocs de configuration indépendants des rôles courants :

- Configuration de base : nom d'hôte, bannières de connexion
- Protocoles de gestion : DHCP, DNS, NTP, SNMP

Politiques d'accès : configurations de sécurité standard

Ces blocs peuvent être réutilisés pour les périphériques principaux, de distribution et d'accès, ce qui simplifie le processus d'automatisation.

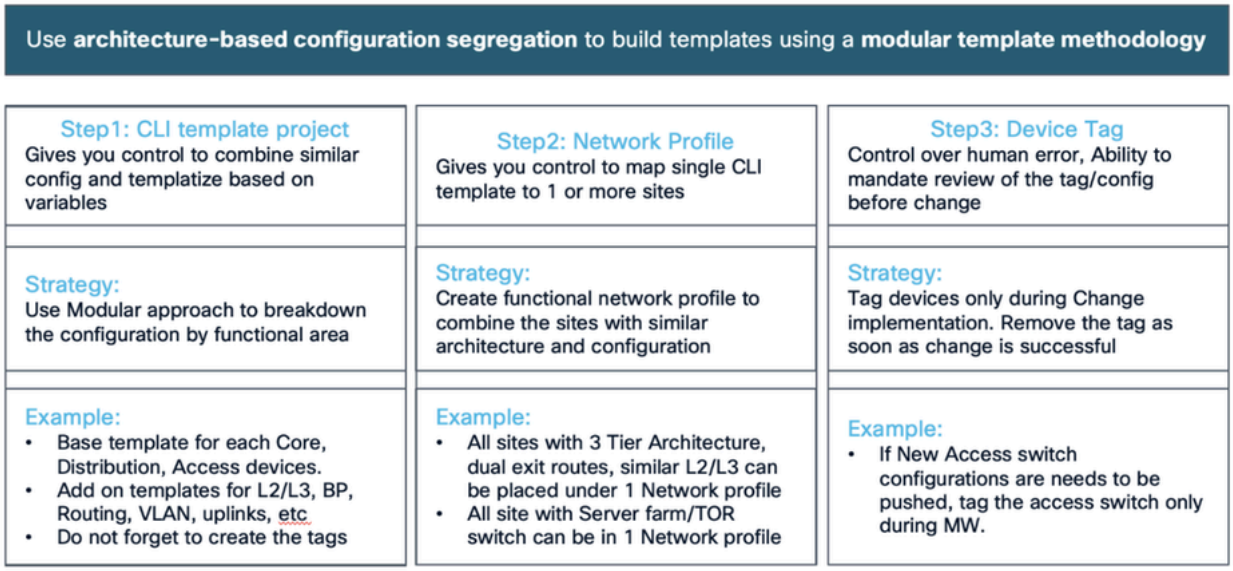



Figure 1 : Meilleure pratique avec exemple

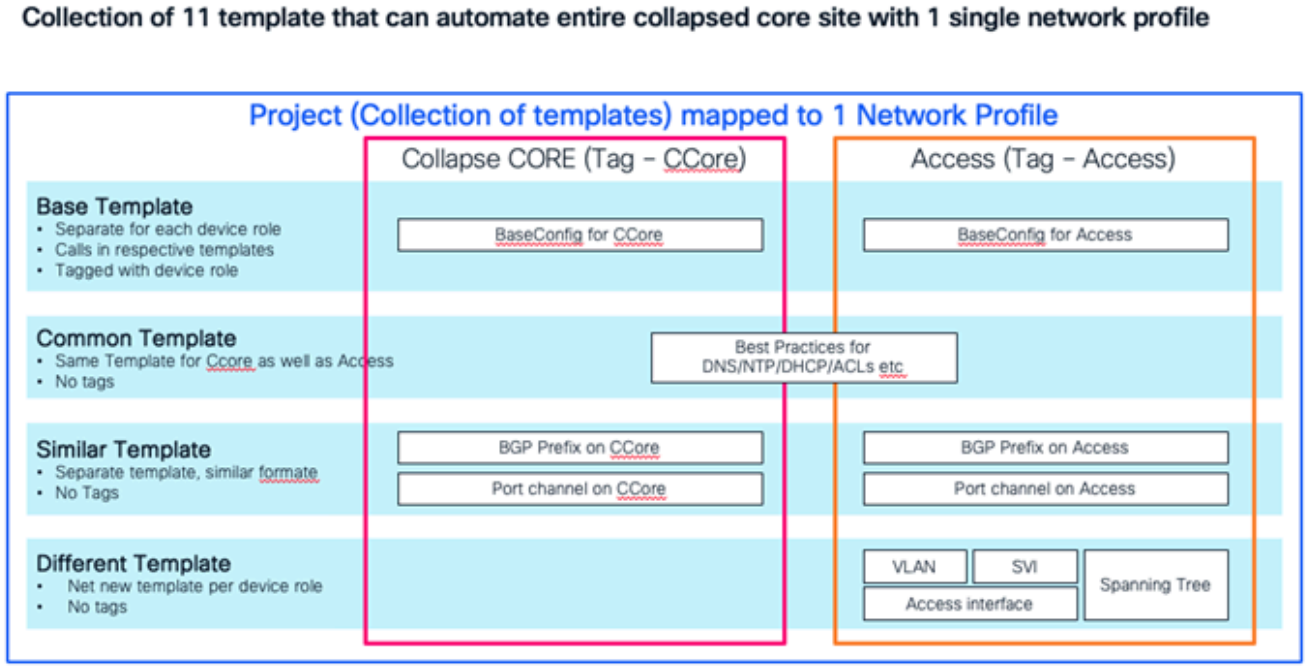



Figure 2 : Exemple de modèle Coeur réduit

Méthodes conseillées pour l'utilisation des modèles

Conception de modèles modulaires pour une configuration automatisée

Lors de l'automatisation des configurations de périphériques dans Cisco Catalyst Center, évitez d'incorporer toutes les configurations dans un modèle monolithique unique. Adoptez plutôt une approche modulaire :

Créez un modèle de base qui fait référence à des modèles (modules) plus petits et spécifiques :

Répartissez la configuration en modules logiques (par exemple, paramètres d'interface, protocoles de routage, fonctions de sécurité).

Cette structure rend les mises à jour plus efficaces : les modifications apportées à un module spécifique sont automatiquement répercutées partout où ce module est utilisé, ce qui réduit considérablement les erreurs et la complexité.

Exemple : Configuration modulaire pour un périphérique de filiale

Supposons que vous automatisez la configuration d'un périphérique de filiale.

Modèle de base :

Inclut des références aux modèles de module pour les zones de configuration clés.

Passe les variables nécessaires à chaque module pour la personnalisation.

Modèles de module :

paramètres_interface : Gère les configurations d’interface.

protocoles_routage : Contient les paramètres OSPF, EIGRP ou BGP.

security_features : Définit les ACL, les règles de pare-feu ou d'autres stratégies de sécurité.

```
{% 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) }}
```

Exemple de structure de modèle de base :

Avec cette structure, les modifications apportées aux configurations de routage ou de sécurité ne doivent être apportées que dans leurs modules respectifs, et ces modifications sont instantanément reflétées partout où le modèle de base est utilisé. Vos configurations sont ainsi plus faciles à gérer et plus cohérentes sur tous les routeurs des filiales.

Ici, le nom du projet est Branch et 3 autres modules différents sont définis sous project. Tous ces éléments sont combinés dans le modèle de base.

Réduire les variables dans le modèle

Réduisez au minimum le nombre de variables dans votre modèle, afin de réduire la complexité et les erreurs. Un nombre réduit de variables simplifie le déploiement, en particulier sur les grands réseaux, ce qui rend le processus plus efficace et plus cohérent.

Utilisation des balises de périphérique pour les modèles

Utilisez les balises de périphérique dans Cisco Catalyst Center, telles que l'emplacement, le rôle ou le site, pour créer des modèles Jinja2 dynamiques et évolutifs. Ces balises activent la logique conditionnelle, garantissant que les configurations correctes sont appliquées aux périphériques appropriés. Cette approche minimise les erreurs et simplifie la gestion des modèles dans divers environnements réseau.

Valeurs Statiques En Code Dur Si Possible

Le codage rigide des valeurs statiques simplifie les modèles et améliore l'efficacité du déploiement. Les adresses IP des serveurs DNS, NTP ou Syslog, qui restent généralement cohérentes entre les périphériques, en sont des exemples courants. De même, l'utilisation d'ID de VLAN standard sur les commutateurs d'accès permet de coder ces valeurs en dur, ce qui réduit la variabilité et accélère le déploiement.

Adoptez une approche en deux étapes : Modèles Jour 0 et Jour N

Lorsque vous intégrez des périphériques à l'aide de services tels que Cisco Plug and Play, utilisez une stratégie de modèle en deux étapes :

Modèles du jour 0 : Poussez les configurations de base pour vous assurer que le périphérique peut communiquer avec Cisco Catalyst Center.

Modèles du jour N : Déployez des fonctionnalités et des configurations avancées une fois le périphérique accessible.

Les meilleures pratiques permettent d'obtenir des modèles efficaces et évolutifs qui simplifient les déploiements de réseaux de campus Cisco.

Contrôle d'espace blanc dans les macros de modèle Jinja

Lors de la création de modèles en langage Jinja, il est essentiel de manipuler soigneusement les espaces et les nouvelles lignes, en particulier lors du rendu de contenu dynamique dans les macros. L'accumulation d'espaces ou de nouvelles lignes peut entraîner des problèmes de formatage dans la sortie générée, ce qui doit entraîner des erreurs d'interprétation ou des erreurs dans le traitement en aval. Pour résoudre ce problème, Jinja fournit la syntaxe permettant de contrôler les espaces : en plaçant un signe moins (-) directement à l'intérieur des délimiteurs ({{- ... -}} ou {%- ... -%}), il supprime tout espace de début ou de fin autour de l'expression. Par exemple, le remplacement de {{item[1]}} par {{- item[1] -}} permet de supprimer les espaces ou les nouvelles lignes supplémentaires lors du rendu de la macro. Cette pratique est particulièrement utile lors de l'itération de listes ou de la génération de fichiers de configuration, comme indiqué dans l'extrait de modèle. Nous vous recommandons de toujours appliquer le contrôle des espaces blancs dans de tels scénarios afin de conserver des résultats nets et prévisibles.

Exemple (utilisation recommandée) :

```
{% pour l'élément dans la liste générique %}
   {% if item[0] == prefix -%}
      {{- item[1] -}}
   {%- endif %}
{%- fin pour %}
```

# Architecture à trois niveaux

Ce livre blanc commence par le développement de modèles pour les commutateurs d'accès jusqu'aux commutateurs principaux et décrit les exigences de chaque couche.

Commutateurs de couche accès

Les commutateurs d'accès sont intégrés à l'aide de Plug and Play et doivent nécessiter un modèle Jour 0. Pour plus d'informations sur le processus Plug-and-Play dans Catalyst Center, veuillez consulter le lien :

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

Comme indiqué précédemment, Catalyst Center prend en charge les langages de modélisation Velocity et Jinja2. Ce document utilise Jinja2 pour illustrer la structure du modèle, en raison de sa flexibilité. La configuration du commutateur de couche d'accès peut être déployée à l'aide des modèles Jour-0 et Jour-N.

Un modèle de base Jour 0 peut être structuré, voir Étape 1 :

Étape 1: Définir un modèle

```
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
!
```

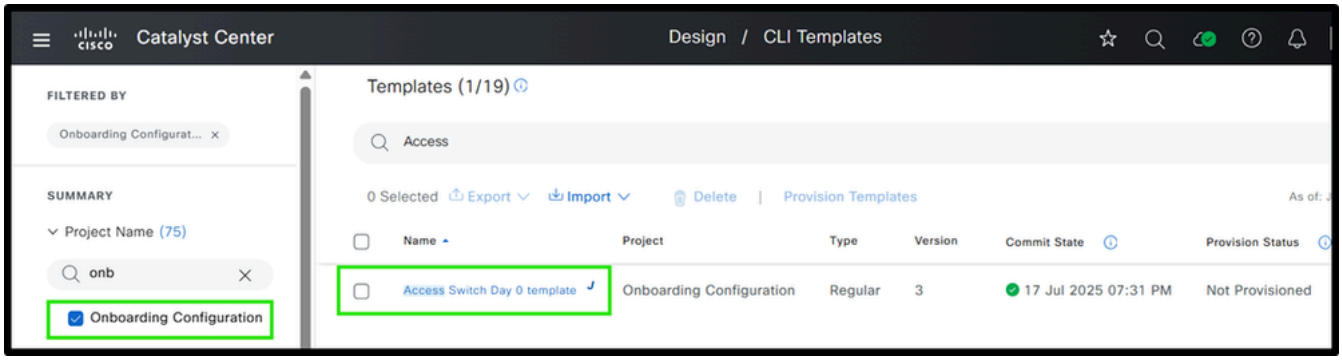
Étape 1: Définir un modèle

Le modèle simplifie la configuration en codant en dur des constantes telles que le nom d'utilisateur, le mot de passe, le secret actif et le masque de sous-réseau, puisque tous les commutateurs d'une filiale partagent le même masque de sous-réseau VLAN de gestion. L’adresse IP de gestion est toutefois unique pour chaque commutateur et est définie comme

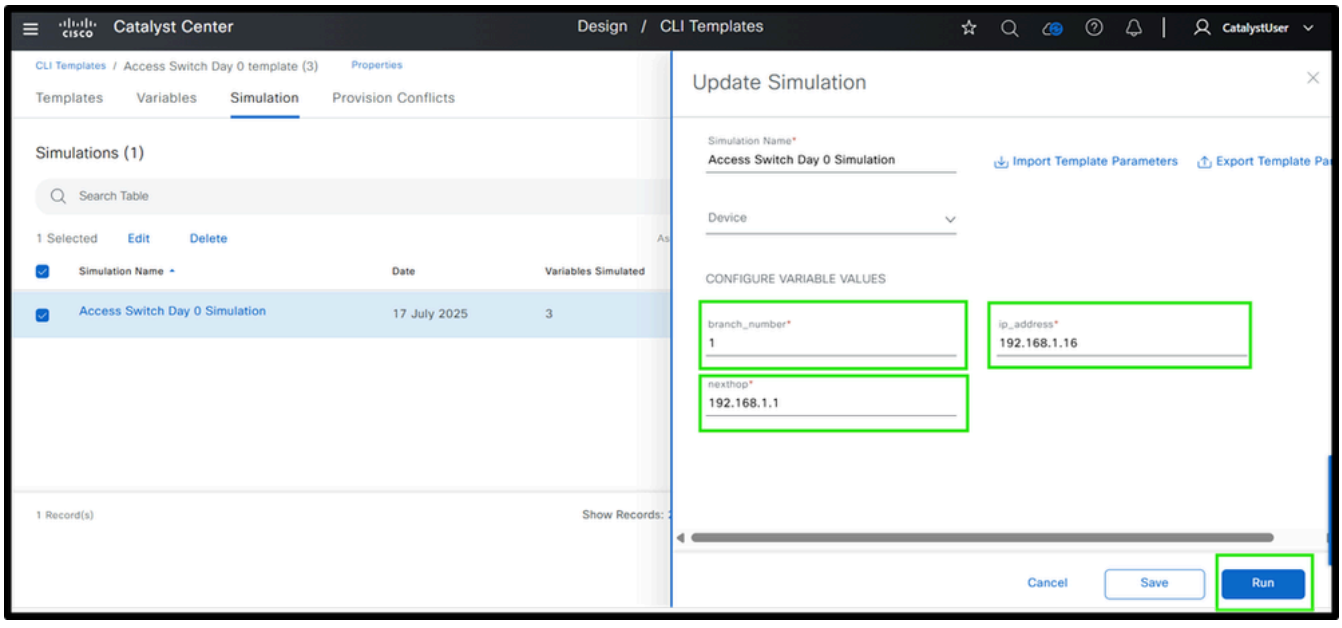
une variable. Une structure de modèle complète doit être fournie dans le modèle Jour N, qui utilise ce modèle Jour 0. Dans le modèle du jour N, chaque fonctionnalité du commutateur d'accès est gérée par un module dédié. Par exemple, un module gère les VLAN de couche 2, des modules distincts gèrent les interfaces d'accès de liaison montante et de liaison descendante, un autre module se concentre sur le renforcement de la sécurité, etc.

Bien que les ID de VLAN cohérents soient préférés, des ID variables peuvent être générés dynamiquement à l'aide d'une formule basée sur le numéro de branche (par exemple, Branch1 = VLAN 113, Branch2 = VLAN 213). Le modèle est ainsi réutilisable dans toutes les filiales. De même, l'adresse IP du tronçon suivant est une variable, car elle doit différer par branche en fonction du cluster de distribution connecté.

Étape 2: Simuler et fournir des variables




Accéder à la structure du modèle de commutateur jour 0 avec entrées et sorties de simulation




Ex. Entrées de simulation

Il est toujours recommandé de simuler le modèle avant le déploiement. La capture d'écran

montre la configuration finale après la saisie des variables.

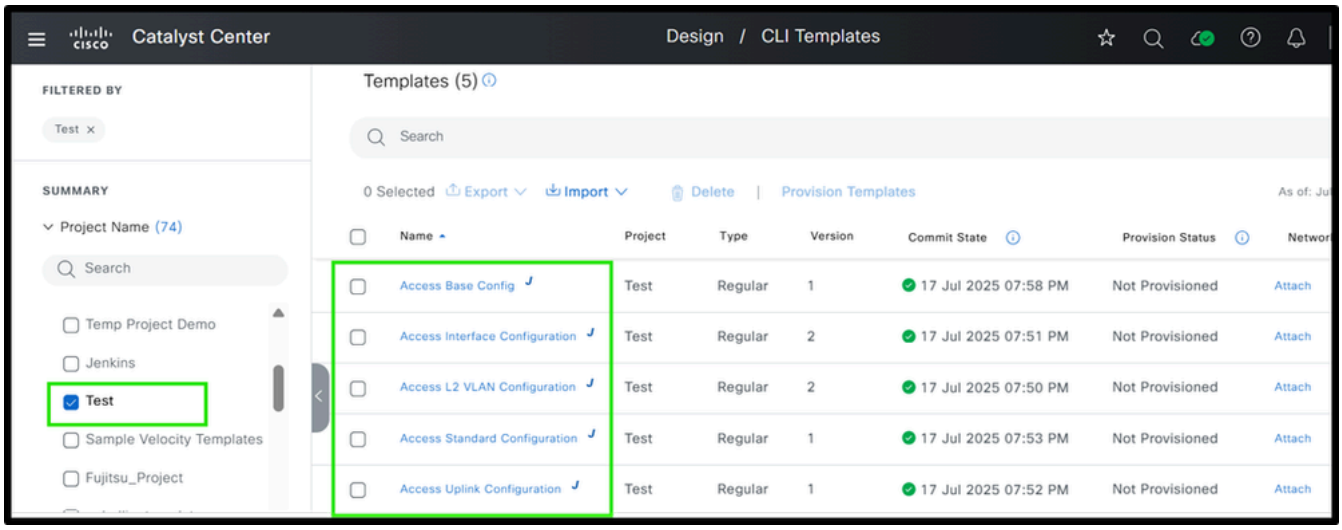

Config. finale après saisie des valeurs

Voyons à présent comment créer un modèle modulaire de jour N.

La configuration du commutateur d'accès peut être divisée en différents modules, qui peuvent tous être combinés dans un module de base. Le modèle de base pour les commutateurs d'accès est structuré comme indiqué.

Le modèle de base et ses modules sont tous deux créés dans un projet nommé « Test » dans Cisco Catalyst Center.

Étape 1: Définir divers modèles, y compris le modèle de base



## Étape 2: Définition de différents modules

### Configuration de base Access :

La capture d'écran présente un exemple de configuration de base.

```
{% 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) }}
```

Configuration de base Access

Ce modèle de configuration modulaire comprend quatre parties : Configuration VLAN, configuration d’interface de liaison ascendante, configuration d’interface d’accès et configuration standard. Elle n'utilise que deux variables : branch_number et is_poe, pour une gestion simple et facile.

Branch_number calcule les ID de VLAN propres à la branche, comme indiqué dans le modèle Jour 0, et is_poe détermine si le commutateur d'accès est un commutateur PoE ou non PoE. Ces variables sont fournies lors du provisionnement et transmises aux modules pour créer les configurations correctes, ce qui réduit les efforts et améliore l'efficacité.

Examinons maintenant chaque module pour voir comment il contribue à générer des parties spécifiques de la configuration globale.

### Accéder à la configuration VLAN L2

```
{% 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 %}
```

Accéder à la configuration VLAN L2

Ce module crée des VLAN en fonction du numéro de branche, comme expliqué précédemment. Les VLAN de données et de voix sont créés sur tous les commutateurs, qu'ils prennent en charge ou non la technologie PoE. Le VLAN de gestion AP (par exemple, 114 pour Branch1) est créé uniquement si is_poe est défini sur "Yes", ce qui signifie que le commutateur prend en charge PoE. Si is_poe est "No," le VLAN de gestion AP est ignoré puisque les commutateurs non-PoE ne peuvent pas prendre en charge les points d'accès. Cette opération est gérée à l'aide d'une condition if.

```
{% 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 %}
```

Configuration d'interface

Ce module gère la configuration de l'interface d'accès et utilise la même approche que le commutateur PoE décrit précédemment. Si la variable is_poe est "Yes," ce qui signifie que le commutateur est un commutateur PoE, les six premiers ports (1-6) doivent être configurés avec le VLAN de gestion AP. Sinon, les six premiers ports doivent être définis comme ports d'accès utilisateur.

En supposant que le commutateur est un modèle à 24 ports, les ports restants (7 à 24) sont toujours configurés en tant que ports d'accès utilisateur, que le commutateur soit PoE ou non.

La plage de l'interface a été normalisée et n'est plus considérée comme une variable d'entrée, ce qui est considéré comme une meilleure pratique pour réduire le nombre de variables dans le modèle. En outre, le module inclut une macro nommée common_access_settings, qui réduit la taille du modèle en consolidant les configurations répétées. Cette macro est simplement appelée dans les paramètres de l'interface, ce qui évite d'avoir à les spécifier plusieurs fois.

Remarque : Ce modèle applique la même description à toutes les interfaces d'accès. Si des descriptions uniques sont nécessaires pour chaque interface, il est recommandé de les pousser en utilisant des scripts Python distincts ou des outils d'automatisation similaires.

Examinez le module qui génère les configurations pour les interfaces de liaison ascendante.

```
{% 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 %}
```

Accéder à la configuration Uplink

Ce module génère la configuration des interfaces de liaison ascendante et gère l'élagage VLAN. Si le commutateur prend en charge PoE, le VLAN de gestion AP est inclus dans la liste des VLAN autorisés ; dans le cas contraire, elle est exclue. Cette logique est gérée à l'aide de la condition if dans le code, comme décrit précédemment.

Révisez le dernier module, qui présente les configurations standard, y compris les meilleures pratiques et le renforcement de la sécurité.

Mise en garde : Notez que ce document est fourni à titre indicatif uniquement et ne doit pas être utilisé comme référence pour les configurations réseau réelles, car les configurations peuvent varier en fonction des besoins spécifiques

```
{% 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
```

Partie 1 : Accéder à la configuration standard

```
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 %}
```

Partie 2 : Accéder à la configuration standard

Ce module génère une configuration standard qui intègre les meilleures pratiques, le renforcement de la sécurité et les fonctionnalités clés pour la gestion sécurisée des périphériques. La plupart des valeurs sont codées en dur pour la cohérence entre les branches, à l'exception de branch_number, qui est utilisée pour calculer le VLAN de gestion pour les commutateurs dans chaque branche et sert d'interface source pour plusieurs configurations.

Étape 3: Effectuez une simulation avant de configurer les commutateurs. Seule la configuration de base doit être simulée.

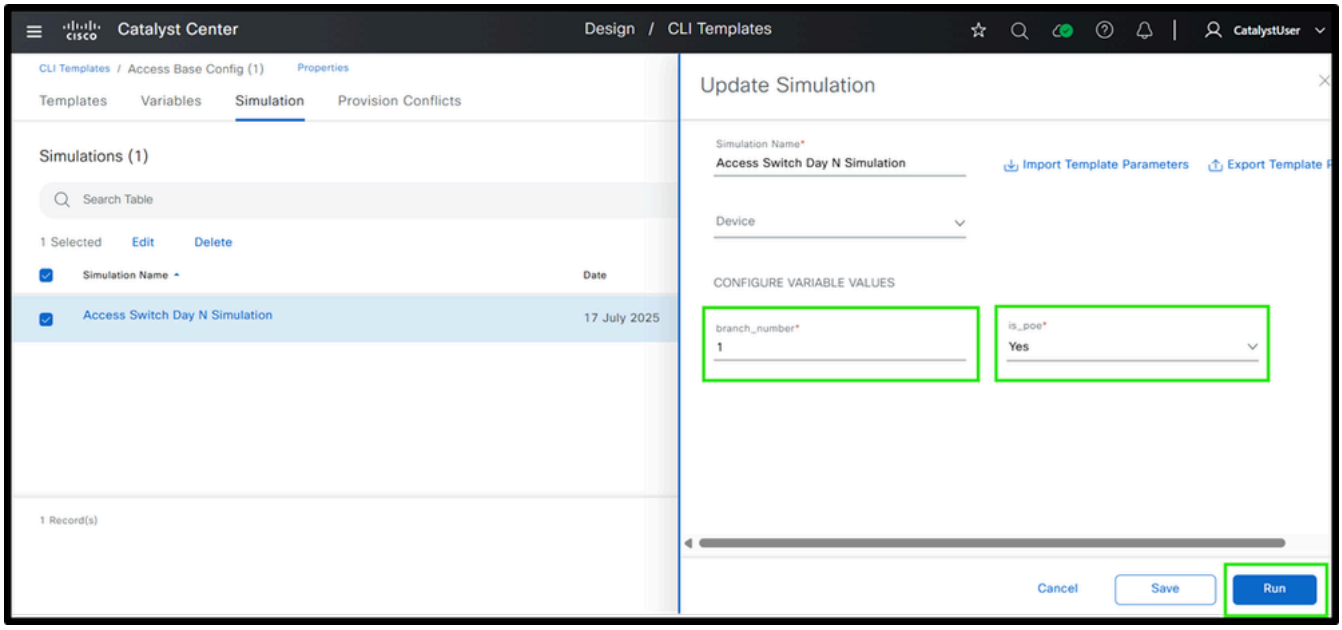
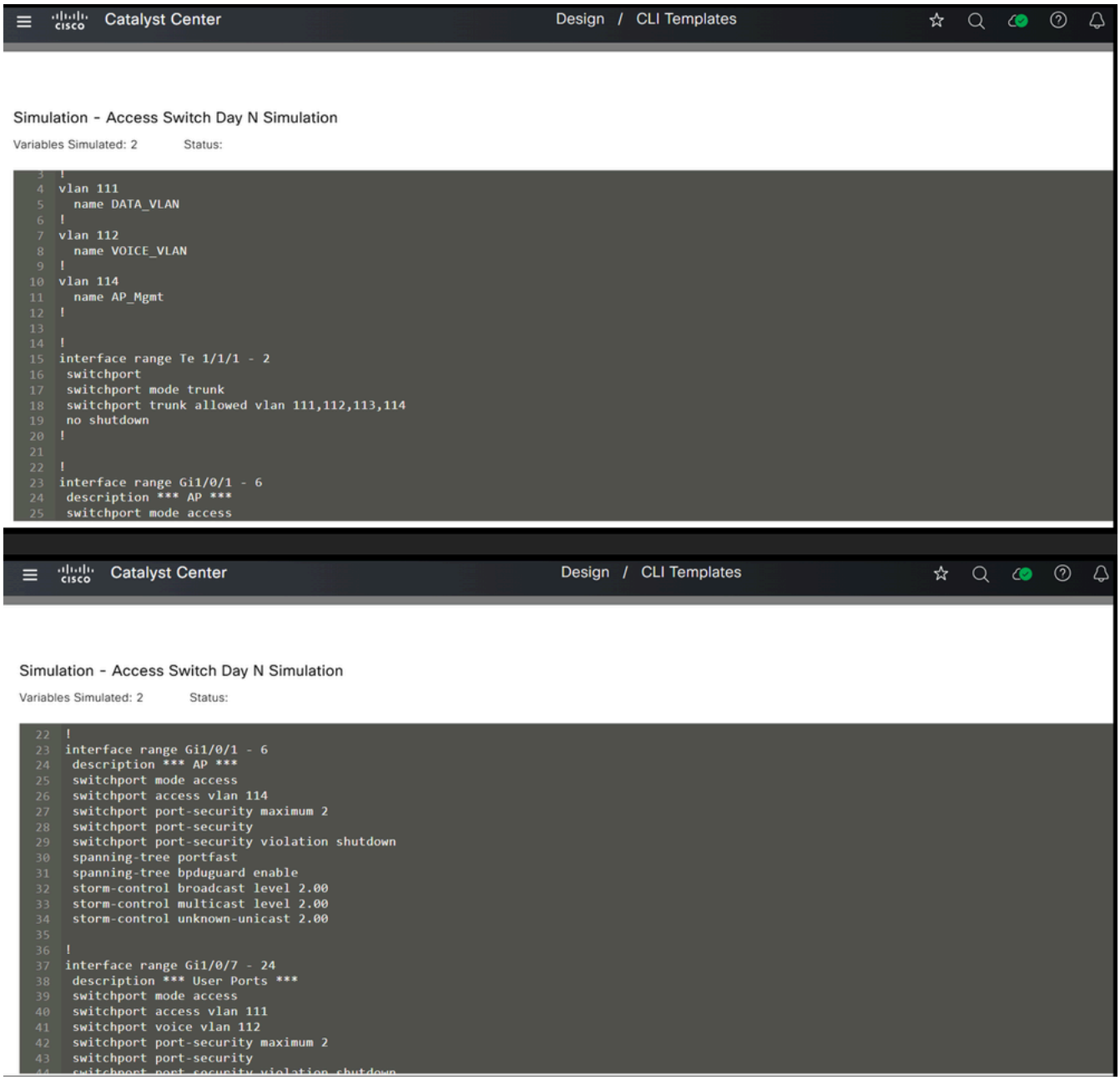



Figure 7 : Accéder aux entrées et sorties de simulation du modèle jour N du commutateur




Simulation

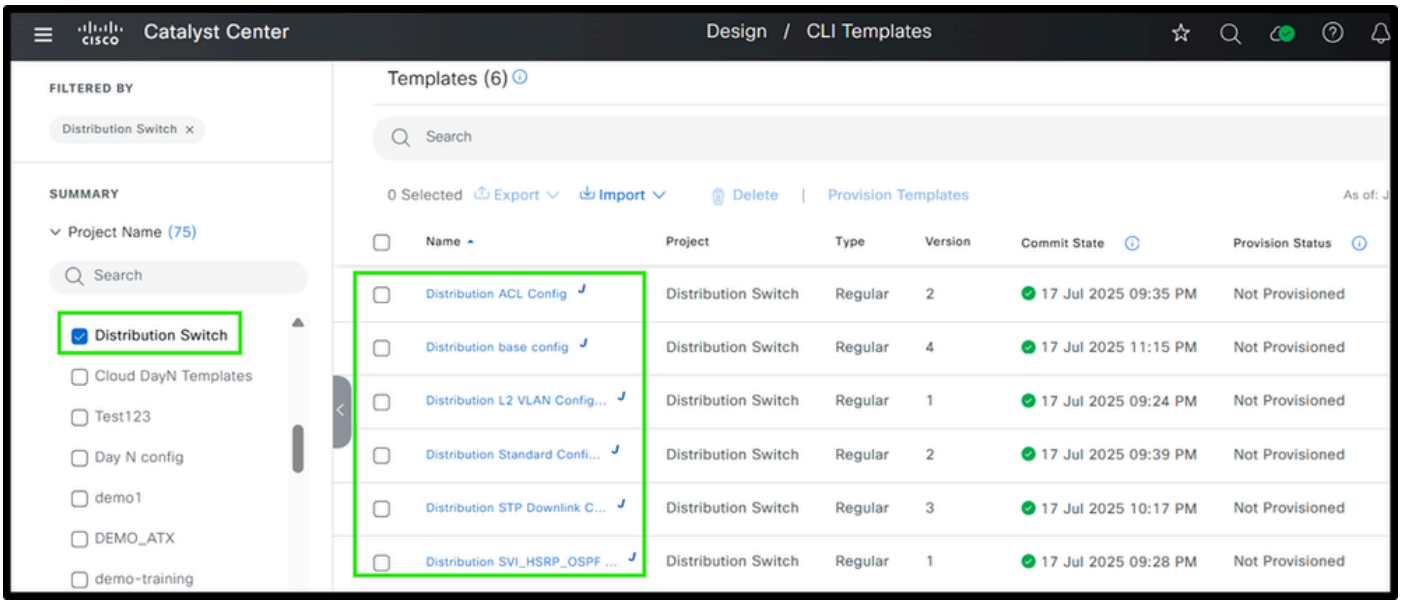
C'est ainsi que les modèles peuvent être utilisés au niveau de la couche d'accès pour générer des configurations.

Examinons maintenant les périphériques de la couche de distribution pour voir comment la modélisation peut leur être appliquée.

## Commutateurs de couche de distribution

Nous allons maintenant concevoir un modèle modulaire pour les commutateurs de distribution. Le modèle de base et ses modules font partie du projet « Commutateur de distribution » dans Cisco Catalyst Center.

Étape 1:  Structure du modèle de commutateur de distribution




Ex. Modèles de distribution

Étape2:  Définir chaque module

La configuration de base fournie définit chaque module et tous les modules sont référencés.

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

Ex. Modules de modèle de base de distribution

Comme pour les commutateurs d'accès, tous les modèles sont créés dans le projet 'Commutateur de distribution' et référencés dans le modèle de base. Bien que certains modèles soient identiques à ceux utilisés pour les commutateurs d'accès, cette section explique les différences spécifiques aux commutateurs de distribution. Le module "Distribution L2 VLAN Configuration" est identique à celui décrit précédemment pour les commutateurs d'accès. Vérifiez le module Access L2 VLAN Configuration qui fournit ces informations. Il génère les VLAN requis en fonction des valeurs d'entrée fournies pour les variables.

Examinons maintenant le module "Distribution STP Downlink Config", qui gère la génération de configurations Spanning Tree et de liaisons ascendantes pour les commutateurs de distribution.

```
{% 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 %}
```

Configuration de liaison descendante STP de distribution

Ici, la macro-fonctionnalité Jinja2 est utilisée, qui est référencée dans le module basé. Donc cette structure aide à construire une approche modulaire.

Ce module configure le protocole STP (Spanning Tree Protocol) et les interfaces de liaison descendante en fonction du "numéro_de_branche" et si le commutateur est compatible PoE. La variable "branch_number" est utilisée pour générer des VLAN de base uniques pour chaque branche, assurant des VLAN distincts, similaires à l'approche déjà mise en évidence pour les commutateurs d'accès. Si le commutateur est compatible PoE ("is_poe" == 'Yes'), un VLAN supplémentaire, tel que le VLAN de gestion AP, est ajouté à la liste. La variable "distribution_number" détermine la priorité STP, en définissant 4096 pour Distribution 1 (ce qui en fait le pont racine préféré) et 8192 pour les commutateurs de distribution secondaires. Enfin, les VLAN appropriés sont appliqués à l'interface d'agrégation, ce qui garantit que seuls les VLAN appropriés sont autorisés selon que le commutateur est compatible PoE ou non.

Examinons maintenant le module "Distribution SVI_HSRP_OSPF Config", qui se concentre sur la configuration des interfaces SVI, HSRP et OSPF pour un routage et une redondance réseau

efficaces.

```
{% 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 %}
```

Configuration de distribution SVI_HSRP_OSPF

Ce module, Distribution_SVI_HSRP_OSPF_Config, aide à configurer les interfaces SVI, HSRP, OSPF et de liaison ascendante pour les commutateurs de distribution. Dans cet exemple, nous

nous concentrons sur l'interface SVI pour les sous-réseaux de données, mais la même méthode peut être utilisée pour d'autres interfaces SVI comme la voix ou la gestion.

Si la planification des adresses IP pour les sous-réseaux de données est déjà effectuée, les adresses IP peuvent être calculées automatiquement pour chaque interface SVI en fonction des variables branch_number et distribution_number. Par exemple, si la branche 1 a le sous-réseau 172.17.0.0/20, la branche 2 a 172.17.16.0/20, et la branche 3 a 172.17.32.0/20, l'adresse IP de la passerelle est 172.17.x.1 (où x est le numéro de la branche). La deuxième adresse IP pour le premier commutateur de distribution est 172.17.x.2, et la troisième adresse IP pour le deuxième commutateur de distribution est 172.17.x.3. De cette façon, les adresses IP sont automatiquement calculées, ce qui réduit les erreurs et simplifie le processus.

L'interface de bouclage reçoit une adresse IP de la variable loopback_ip, qui sert d'ID de routeur OSPF pour assurer un routage stable et cohérent sur le réseau. Dans la configuration OSPF, cette adresse IP de bouclage est utilisée comme ID de routeur et les interfaces appropriées sont ajoutées à la zone OSPF 0. Pour HSRP, les valeurs de priorité sont définies : 255 pour le premier commutateur de distribution et 250 pour le second, garantissant un basculement correct. En outre, l'authentification HSRP est configurée à l'aide d'une chaîne de clés (HSRP_KEY) pour améliorer la sécurité.

Pour que la configuration reste propre et gérable, certaines valeurs sont codées en dur. Par exemple, le masque de sous-réseau (255.255.240.0) et certains paramètres HSRP (comme la version et BFD) sont identiques dans toutes les branches, ce qui réduit le nombre de variables. La configuration est ainsi plus simple, plus facile à appliquer et moins sujette à des erreurs. Enfin, les interfaces de liaison ascendante sont configurées avec des adresses IP et ajoutées à la zone OSPF 0 pour un routage correct entre les commutateurs. Cette approche facilite la gestion du processus de configuration et réduit le risque d'erreurs, tout en étant flexible pour les différentes filiales.

Revoyez maintenant le module "Distribution ACL Config", qui permet la segmentation au niveau de la couche de distribution.

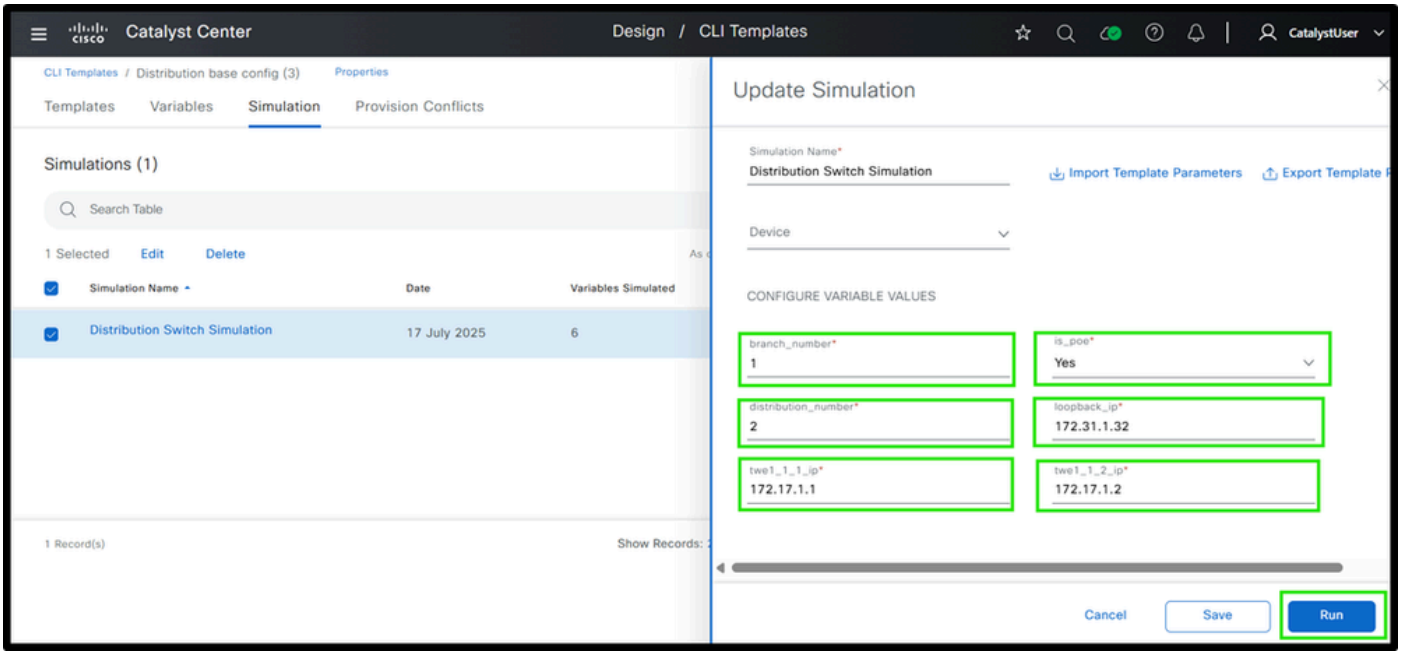
```
{% macro Distribution_ACL_Config (branch_number) %}
!
ip access-list extended BLOCK_BRANCH
 deny ip 172.17.{{ 16 * (branch_number - 1) }}.0 0.0.15.255 172.16.{{ 16 * (branch_number - 1) }}.0 0.0.15.255
 deny ip any host 239.255.255.250
 permit ip any any
!
interface vlan {{ 100 * branch_number +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

Configuration ACL de distribution

Ce module présente la segmentation au niveau de la couche de distribution à l'aide d'un modèle Jinja2. Il utilise la variable branch_number pour calculer dynamiquement les adresses de sous-réseau, ce qui permet des configurations de listes de contrôle d'accès automatisées et évolutives. Pour chaque branche, la liste de contrôle d'accès bloque la communication entre le sous-réseau 1 (172.17.X.0) et le sous-réseau 2 (172.16.X.0) en refusant le trafic IP entre ces plages. Il refuse également le trafic vers l'adresse de multidiffusion 239.255.255.250, tout en autorisant tout autre trafic. L'interface VLAN est attribuée dynamiquement en fonction du numéro de branche, et la liste de contrôle d'accès est appliquée en entrée sur cette interface. Cette approche automatisée assure une segmentation efficace par filiale, réduit les erreurs de configuration manuelles et simplifie l'application des politiques réseau.

Enfin, le dernier module, "Configuration de la norme de distribution", est presque identique à celui décrit dans le module Configuration de la norme d'accès (veuillez vous reporter à cette section pour plus de détails). Il inclut les meilleures pratiques, le renforcement de la sécurité et des fonctionnalités clés pour une gestion sécurisée des périphériques. La seule différence réside dans l'interface source : dans le modèle de commutateur d'accès, il est défini comme VLAN {{ numéro_branche * 100 + 13 }}, alors que dans la configuration de commutateur de distribution, il peut être codé en dur comme Loopback0.

Étape 3: Effectuez une simulation avant de déployer la configuration.




(1) Entrées et sorties de simulation de modèle de commutateur de distribution


(2) Entrées et sorties de simulation de modèle de commutateur de distribution


(3) Entrées et sorties de simulation de modèle de commutateur de distribution

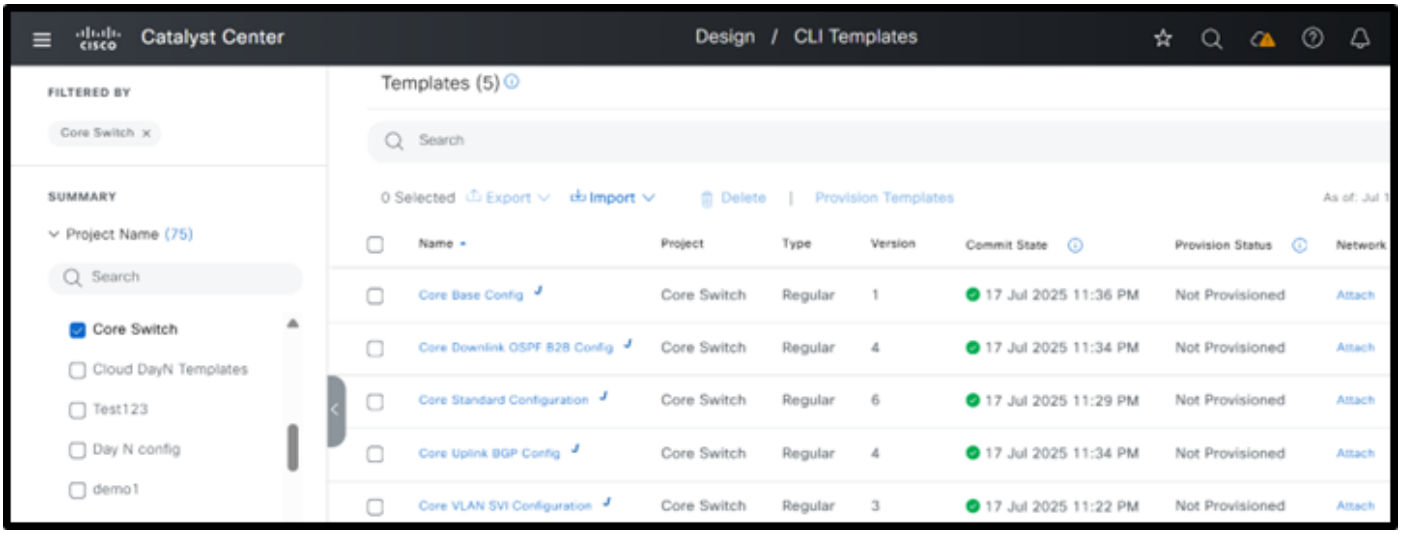

(4) Entrées et sorties de simulation de modèle de commutateur de distribution

C'est ainsi que les modèles peuvent être utilisés au niveau de la couche de distribution pour générer des configurations. Examinons maintenant les périphériques de la couche coeur de réseau pour voir comment la modélisation peut y être appliquée.

Commutateurs de couche coeur

Concevez maintenant un modèle modulaire pour les commutateurs principaux. Le modèle de base et ses modules font partie du projet « Core Switch » dans Cisco Catalyst Center. Reportez-vous au modèle de base à l'étape 1.

Étape 1: Définition de la structure des différents commutateurs principaux




Structure du modèle de commutateur principal

Étape 2: Définition de différents modules

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

Configuration de base principale

La plupart des configurations de commutateur principal sont similaires dans toutes les filiales, de sorte que les valeurs communes peuvent être codées en dur. En règle générale, seules les adresses IP changent et elles peuvent être définies à l'aide de variables. Étant donné que chaque filiale ne dispose généralement que de deux commutateurs principaux, la gestion de ces variables est simple. Même si certaines filiales disposent de commutateurs principaux supplémentaires, leur nombre reste inférieur au nombre de commutateurs d'accès ou de distribution. C'est pourquoi il est plus important de réduire les variables des commutateurs d'accès et de distribution, car elles sont utilisées en plus grand nombre et le fait d'avoir trop de variables peut rendre la configuration plus longue.

Commencez maintenant par le premier module : "Configuration de l'interface SVI du VLAN principal." Dans cet exemple, les commutateurs principaux sont placés derrière un pare-feu et doivent établir un appairage eBGP avec celui-ci. Ce module est responsable de la génération des VLAN et des SVI correspondantes nécessaires pour l'appairage eBGP et le voisinage OSPF. Le pare-feu est supposé fonctionner dans une configuration active/en veille.

```
{% 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 %}
```

Configuration SVI du VLAN principal

Comme expliqué précédemment, ce module crée les VLAN requis et les SVI associées pour établir des relations de voisinage OSPF et BGP. Tous les paramètres, à l'exception des adresses IP SVI, sont codés en dur, y compris le masque de sous-réseau s'il est aligné sur le plan d'adressage IP. Cette méthode permet de limiter les variables et de réduire le risque d'erreurs de configuration.

Examinons maintenant le module "Core Downlink OSPF B2B Config", qui génère des configurations pour les interfaces de liaison descendante, OSPF et les liaisons back-to-back entre le commutateur principal 1 et le commutateur principal 2.

```
{% 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 %}
```

Config. coeur de liaison descendante OSPF B2B

Comme dans le module précédent, la plupart des valeurs de ce module sont codées en dur pour minimiser le nombre de variables. Seules les adresses IP des interfaces de bouclage et de liaison descendante sont variables. En outre, les canaux de port dos à dos et les VLAN sont normalisés dans différentes filiales.

Maintenant, examinons le module "Core Uplink BGP Config", qui génère des configurations BGP et gère les liaisons ascendantes connectées aux pare-feu.

```
{% 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 %}
```

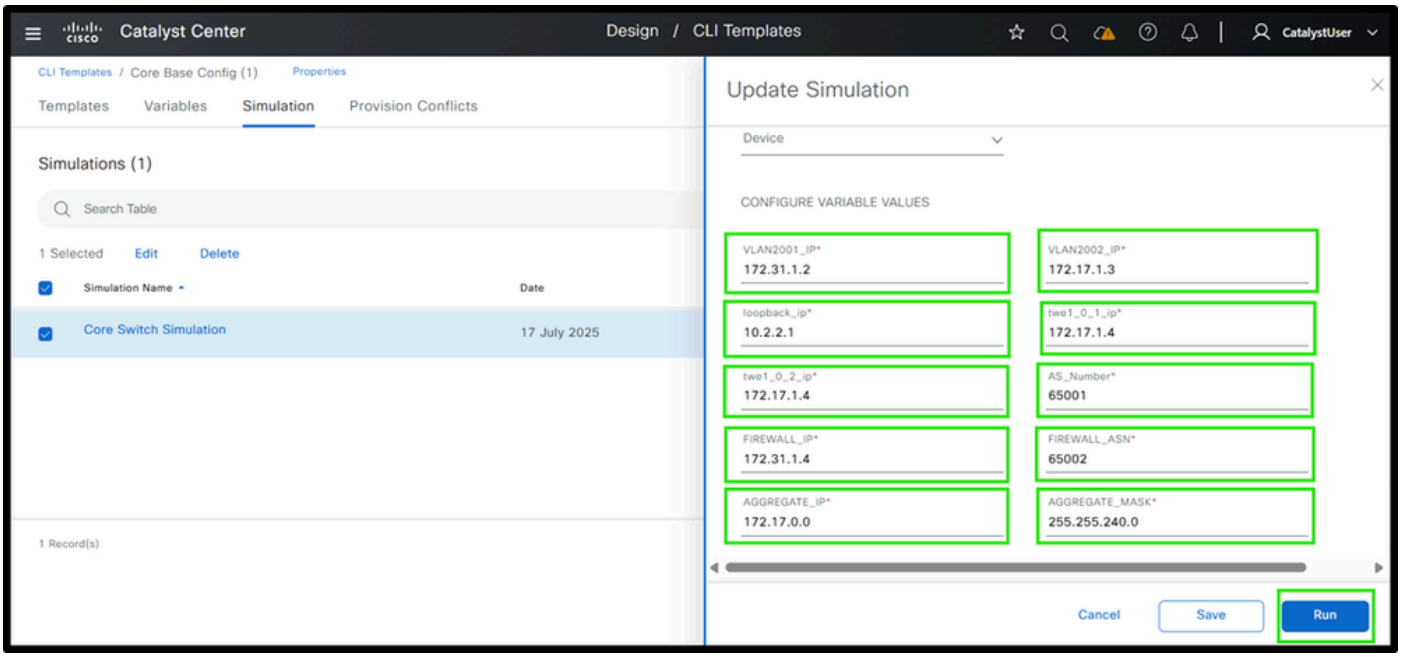
Configuration BGP de liaison ascendante principale

Ce module génère la configuration BGP nécessaire pour établir une relation de voisinage eBGP avec le pare-feu. Comme indiqué ci-dessus, la plupart des valeurs sont codées en dur car elles

restent cohérentes entre les différentes branches. Seules les adresses IP et les numéros de système autonome, qui peuvent varier pour chaque branche, sont pris comme variables d'entrée et utilisés pour générer la configuration nécessaire. La plupart des autres paramètres ont été normalisés afin de réduire le nombre de variables. Les interfaces de liaison ascendante connectées au pare-feu sont spécifiées avec le VLAN utilisé pour le voisinage eBGP, qui a été généré par le module précédent.

Enfin, le dernier module, « Core Standard Configuration », est presque identique à celui décrit dans la section Access Standard Configuration (reportez-vous à cette section pour plus de détails). Il inclut les meilleures pratiques, le renforcement de la sécurité et des fonctionnalités clés pour une gestion sécurisée des périphériques. Conformément à la configuration du commutateur de distribution, l'interface source peut également être définie sur loopback0 dans ce module, et cette valeur peut être codée en dur.

Étape 3: Simulation

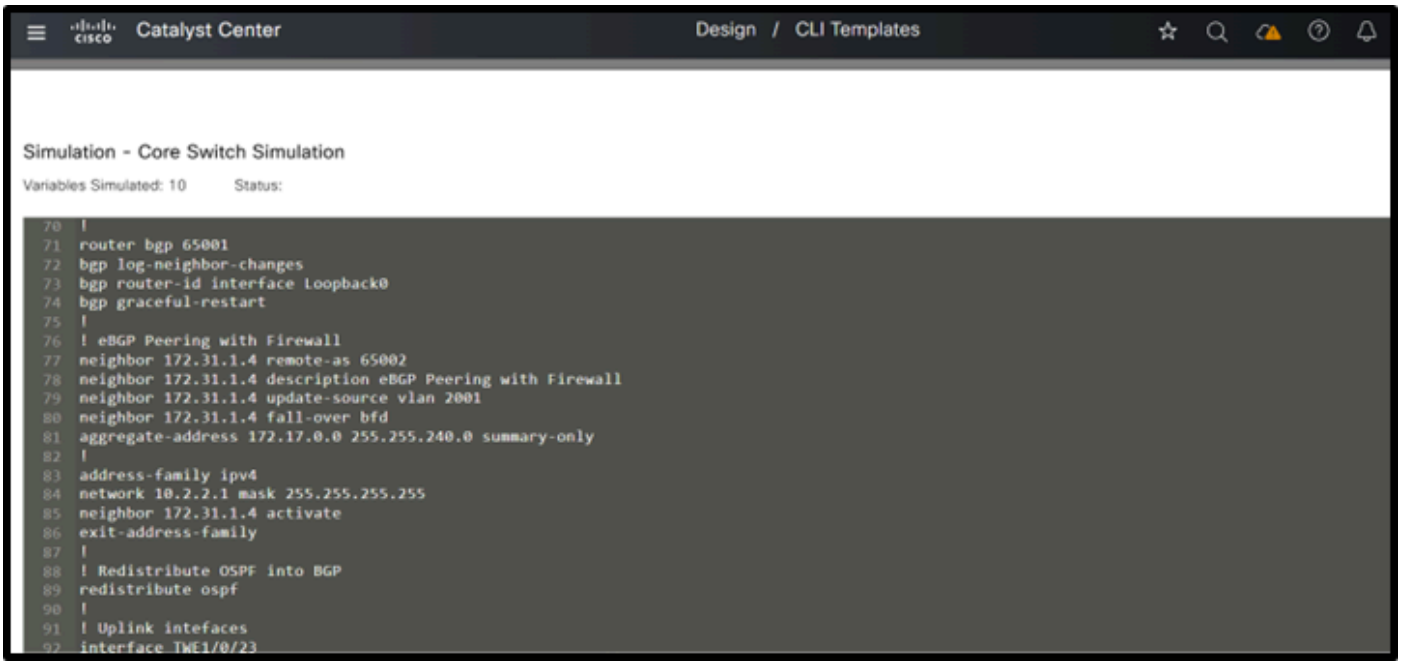



(1) Entrées et sorties de simulation du modèle de commutateur principal


(2) Entrées et sorties de simulation du modèle de commutateur principal


(3) Entrées et sorties de simulation du modèle de commutateur principal


(4) Entrées et sorties de simulation du modèle de commutateur principal




(5) Entrées et sorties de simulation du modèle de commutateur principal

Cette section complète l'explication détaillée de la conception de modèles pour l'architecture à trois niveaux, en décrivant à la fois la structure et la configuration de chaque module.

Tous ces modules utilisent les meilleures pratiques expliquées précédemment.

---

Remarque : Lorsque vous concevez des modèles pour une architecture principale réduite, reportez-vous aux explications fournies pour l'architecture à trois niveaux. La structure du

---

modèle reste la même ; cependant, les fonctions précédemment implémentées séparément au niveau des couches coeur de réseau et de distribution sont désormais combinées au niveau de la couche coeur de réseau réduite. La même approche de modèle modulaire peut également être utilisée ici, en créant un modèle de base et en référençant les modules appropriés au sein de celui-ci.

## Résumé

L'architecture de campus traditionnelle à 3 niveaux repose souvent sur une configuration manuelle étendue sur les couches coeur de réseau, distribution et accès. Cette approche prend non seulement du temps, mais elle est également sujette à l'erreur humaine. L'absence d'automatisation et de gestion centralisée augmente considérablement les frais d'exploitation, ce qui rend difficile l'évolutivité et la gestion efficaces des réseaux de campus modernes et dynamiques. Les configurations des fonctions du modèle CLI de Catalyst Center peuvent être automatisées pour les réseaux LAN traditionnels. Il est important d'utiliser l'approche modulaire lors du provisionnement des périphériques. Les modules peuvent être basés sur les différentes fonctionnalités utilisées au niveau des différentes couches. Et enfin lier tous ces modules au module de base.

## Appel à l'action

Nous invitons les entreprises à adopter la méthodologie de modèle modulaire présentée dans ce livre blanc en tant que meilleure pratique pour standardiser les configurations de commutateurs et optimiser les opérations réseau sur les architectures à trois niveaux et les architectures principales regroupées.

- En implémentant des modèles modulaires, les équipes réseau peuvent :
- Améliorer l'efficacité opérationnelle grâce à des pratiques de configuration cohérentes et reproductibles.
- Minimiser les erreurs humaines et réduire le temps de dépannage.
- Obtenez une plus grande évolutivité pour soutenir la croissance et l'évolution des besoins de l'entreprise.
- Assurez la cohérence de la configuration dans divers environnements.

Cette approche permet non seulement de rationaliser la gestion quotidienne, mais également d'accélérer les déploiements, de simplifier les cycles de mise à jour et d'améliorer l'alignement avec les exigences en matière de sécurité et de conformité. L'adoption de modèles modulaires permet à votre réseau de bénéficier d'une souplesse, d'une résilience et d'une réussite à long terme dans un environnement informatique en constante évolution.

Pour des démonstrations pratiques, en savoir plus sur les modèles, veuillez consulter la série YouTube

1 Comment créer et gérer des modèles dans Catalyst Center

https://youtu.be/SyUqEEcwye0

2 Comment utiliser les variables de liaison système dans les modèles CLI dans Catalyst Center

https://youtu.be/gV1QBuHYJdo

## Auteurs

Naveen Kumar, architecte de prestation client, expérience client Cisco

Risabh Mishra, Ingénieur-conseil, Expérience client Cisco

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.