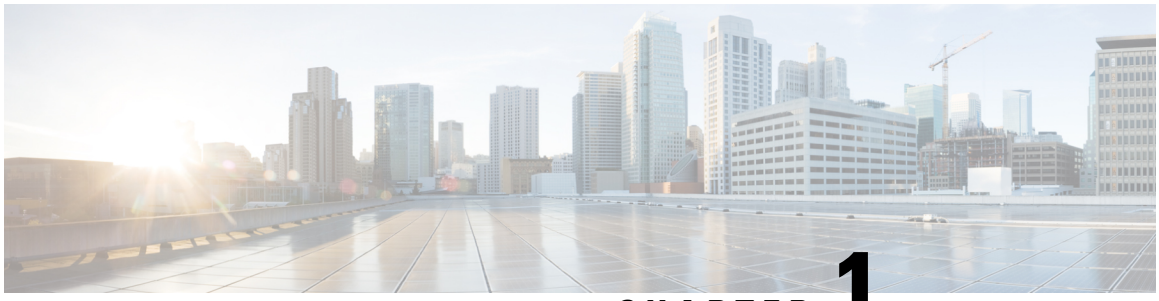
Cisco Metro Solution Guide

First Published: 2025-02-01

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Cisco Metro Solution

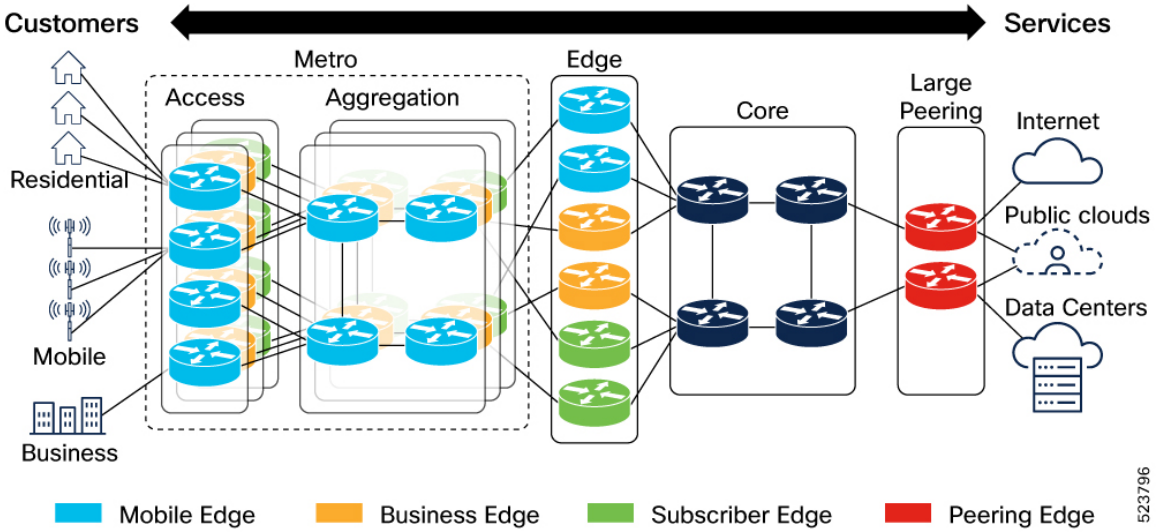
This chapter covers the overview, benefits, architecture, technologies, and deployment models of Cisco Metro solution.

Traditional edge routing networks

Traditional edge routing network architecture

Traditional Edge routing networks are networks where the edge is a set of dedicated routers placed centrally in the network. Users usually have multiple Edge networks for services such as business VPN (L2 and L3), subscriber services, internet peering, data center interconnect (DCI), cloud gateways, and so on.

Figure 1: Traditional edge routing network architecture



Challenges with traditional edge routing networks

Traditional edge routing networks pose challenges in these areas:

- Distributed services: The bandwidth profile and subscriber density for a distributed edge have changed significantly.
- Failure Impact: The large radius for failures demands more complex geographically redundant homing. It also necessitates the use of network hardware with redundant processors, fabric cards, line cards, power supplies, ports, and chassis. This increases the cost and complexity.
- Service optimization: Service providers might have different edge networks for various services. Optimization of these is difficult. The cost of building a solution that meets all the service requirements on a single platform is high. The most demanding services usually set the price for all others.
- Operational efficiency: Limits operators who want simplicity and efficiency.
- Platform lock-in: Large systems remain in place for many years thereby slowing down innovation.
- Upgrades: Upgrades are delayed as much as possible. Forklift upgrades have become a norm due to compatibility challenges between technology generations.

Business challenges

Internet traffic saw a compounded annual growth rate of 30% or higher over the last ten years. Primary challenges for modern networks include proliferation of devices, increase in end-user bandwidth speed, and the continued movement of applications to the cloud.

Traditional edge routing designs face challenges by new traffic and business realities that include

- ever growing bandwidth with flat average revenue per user (ARPU)
- increasing cost and complexity of scaling centralized network architectures, and
- content and application evolution towards distributed network architectures.

Solution

You must build networks to handle the advanced services and increased traffic associated with modern network services. Networks must evolve so the infrastructure layer can keep up with the service layer. The result of these shifts is driving traffic away from centralized delivery to a more distributed network.

Moving forward, we must consider new network architectures as design options that include these key aspects:

- Fully distributed and fixed routing systems to deliver services at any place in the network
- Fabric models delivering scale-out networks that can be built on-demand
- Flexible deployment options matching provider requirements
- Network simplification at all layers of the network

Metro solution

Cisco Metro solution is an architecture evolution of Cisco Converged SDN Transport (CSDN-T) architecture that is focused on converging network infrastructure in multiple dimensions to change the way networks are built. The Metro solution considers edge as a set of functions which can be enabled anywhere in the network.

The Metro solution architecture focuses on these key aspects:

- Enhanced scale and resiliency through distributed networking
- Simplified packet transport
- Simplified overlay services
- Enhanced automation

Benefits of Metro solution

These are the key benefits of Metro solution:

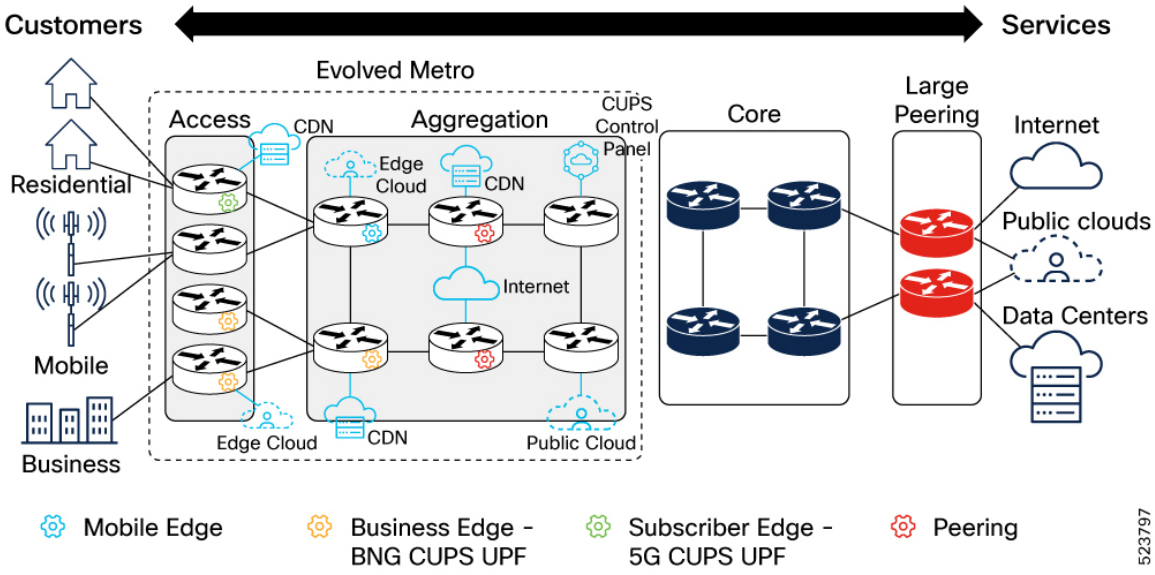
- Technology benefits:
  - High-capacity edge silicon
  - Convergence of network service functions
  - Flexible network design and systems to fit any size location in the network
- Business benefits:
  - Deliver services closer to users and applications
  - Cost savings
  - Sustainability benefits
- Operational benefits:
  - Improved services resilience

- Network efficiency
- Enhanced operations through network automation and orchestration

Metro solution architecture

The Metro network evolution is driven by increasing bandwidth demands, resulting in network functions distributed in the network closer to the end user. This evolution is driving a consequent network architecture evolution. The classical split between access, pre-aggregation, aggregation, and edge leaves room for a more homogeneous network without distinct boundaries between the domains

Figure 2: Metro solution architecture



How the Metro solution architecture differs from the traditional edge routing networks

The Metro solution architecture differs from the traditional edge routing networks in terms of these key aspects:

- Automation-first architecture that is focused on simplified deployment and operation
- Edge fabric for service and bandwidth scale, which is the concept of taking a traditional modular chassis multi-service edge and disaggregating it into a multi-device fabric with greater scale and flexibility
- Edge service placement flexibility using common silicon and feature sets at any place in the network
- A consolidated architecture to handle trends in metro deployments
- A vehicle to introduce new service generating elements such as Edge Service Gateway and inline services

Key pillars of Metro architecture

These are the key pillars of Metro architecture:

- Wide range of supported interfaces:
  - 1/10/25/50/100/400GE and beyond on unified family of Metro devices
  - Any speed user–network interface (UNI) with any service
  - High speed network-to-network interfaces (NNI) and Routed Optical Networking
- Simplified connectivity model and protocols:
  - Segment Routing IPv6 (SRv6) and SR-MPLS underlay networks; SRv6-TE and SR-MPLS TE for advanced Traffic Engineered use cases
  - Secured infrastructure using Trusted Cisco platforms and advanced distributed DDoS protection
  - Co-existence with legacy underlay and overlay technologies
- Business, residential, and mobile subscriber services:
  - EVPN and L3VPN in services layer
  - Private Line Emulation (PLE) for bit-transparent transport of Ethernet and non-Ethernet (OTN, SONET, Fiber Channel)
  - Next-generation subscriber edge using control plane and user plane separation (CUPS)
  - Converged business and subscriber access using Cisco Routed PON
- High performance end-to-end timing and synchronization
- Automation across all components in the architecture covering provisioning, monitoring, and service assurance

High-level use cases of Metro solution

High-level use cases of Metro solution

The Metro solution architecture covers these high-level use cases:

- Next-generation residential subscriber networks deployments
- Enterprise business services
- Mobile network IP transport
- Centralized and edge data center connectivity including networks that are built to support artificial intelligence
- Internet peering, content delivery, and cloud connectivity

Metro solution technologies

This topic covers the various technologies used in Metro solution.

Network technologies and protocols

The table gives a comparison of the common network technologies and protocols that are used in legacy networks vs. the Metro solution.

Table 1: Common network technologies and protocols used in legacy networks vs. the Metro solution

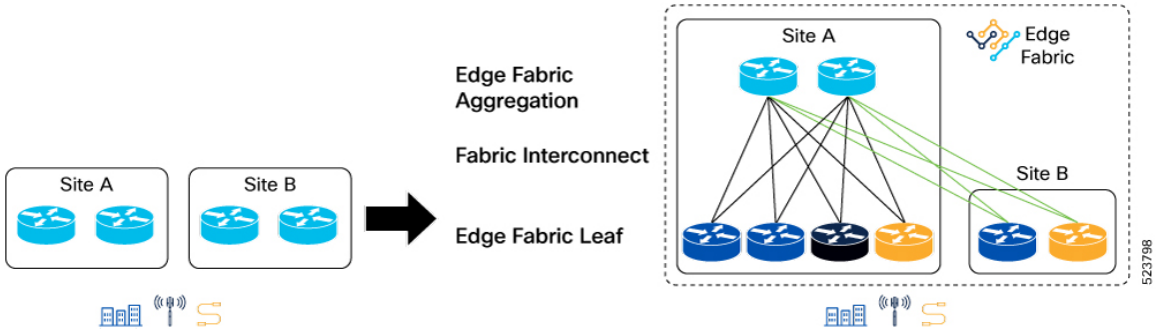
Network technology and protocol	Legacy network		Metro solution
xVPN Services	LDP	BGP	BGP for all L2VPN, and L3VPN
IP Network Scaling	BGP-LU		Segment Routing
Traffic Engineering Fast Reroute	RSVP-TE		
MPLS Overlay Protocol	RSVP-TE	LDP	
IPv6 Transport Overlay	None		
IP to DWDM Transition	Transponder or Muxponder		Routed Optical Networking
	Grey Router Interface		
Private Line Services	Dedicated OTN	Dedicated Ethernet over DWDM	Private Line Emulation
Subscriber BNG	Physical Integrated BNG		Cisco CUPS using Cloud Native BNG
PON Access	Dedicated PON Equipment		Cisco Routed Passive Optical Networking

Metro Edge Fabric

This section details the new disaggregated Metro Edge Fabric, including its components and distributed control plane.

The Metro Edge Fabric is a component of Cisco Metro solution architecture that is designed to provide scalable edge services termination. The Metro Edge Fabric is designed to enhance network efficiency and scalability by separating network functions into distinct physical layers. Cisco Fabric-based Edge solution is a composition of multiple routers in a leaf-spine architecture to accommodate required functionality and scale that cannot be met in a standalone multi-service edge (MSE) model.

Figure 3: Metro Edge Fabric in Metro architecture



Edge Fabric leaf

Leaf nodes are the routers that are used for network service termination use cases. You can split the specific services across a set of leaf devices based on the design and network services. The leaf nodes may be collapsed into a universal leaf for all functions or split between different network or even VPN service type. All Cisco IOS XR platforms can be used as a leaf in the deployment depending on the feature requirements and feature scale.

Edge Fabric spine

The Edge Fabric aggregation routers or spines are the nodes that provides underlay connectivity to all leaf nodes that include service termination nodes, core networking connecting nodes, edge DC connecting nodes, and so on. These spine nodes act as L3/SR-MPLS switch that carry overlay services across leaf nodes. Spine nodes have advanced policy-based traffic management functionalities to support end-to-end QoS for selective overlay services.

Fabric interconnect

The fabric interconnects are the links connecting leaf nodes to spine nodes. Each leaf node must be connected to every spine node to provide maximum resiliency and load balancing across the fabric. It is recommended to standardize local interconnects to one type—copper (CU) or active optical cables (AOC) being the most cost-effective method. Interconnects may also utilize WAN connectivity in the case of remote leaf devices. Longer distances can be covered using Routed Optical Networking components such as ZR/ZR+, DP04QSDD-ER1, and QSFP-DD 100G ZR coherent optics.

Fabric control plane

The Fabric uses standard routing protocols; it does not use proprietary communication between the elements. This allows providers to easily insert any type of node, including third-party node, into the fabric.

Routed Optical Networking

Routed Optical Networking is a Cisco architecture for simplified IP and optical convergence. It is a component of the Metro architecture, utilized as an interconnect technology for WAN links. The high-bandwidth networks require agile, flexible connectivity taking full advantage of the high bandwidth capabilities of next-generation NPUs. Routed Optical Networking must be utilized in places where longer distance interconnects are required between core and edge elements.

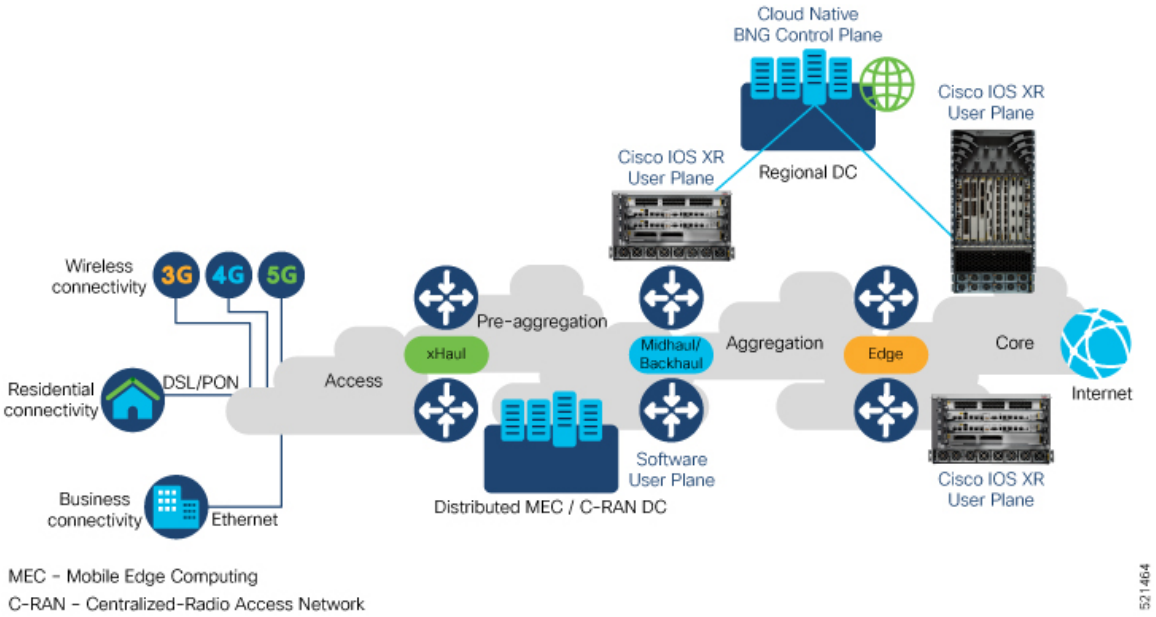
For details, see Cisco Routed Optical Networking Solution Guide.

Subscriber services

CUPS-based subscriber management system

The figure depicts the high-level architecture of Control Plane and User Plane Separation (CUPS)-based subscriber management system in the network.

Figure 4: CUPS architecture



In the CUPS architecture, the subscriber control plane (CP) and user plane (UP) are separated. The CP is responsible for functions such as AAA of subscriber sessions, IP address management, forwarding policies and session specific protocols such as PPPoE, or IPoE. In addition to this, CP assumes the responsibility to onboard, monitor and manage each UP node through the standard interface. One CP instance can manage several UP instances distributed throughout the metro network.

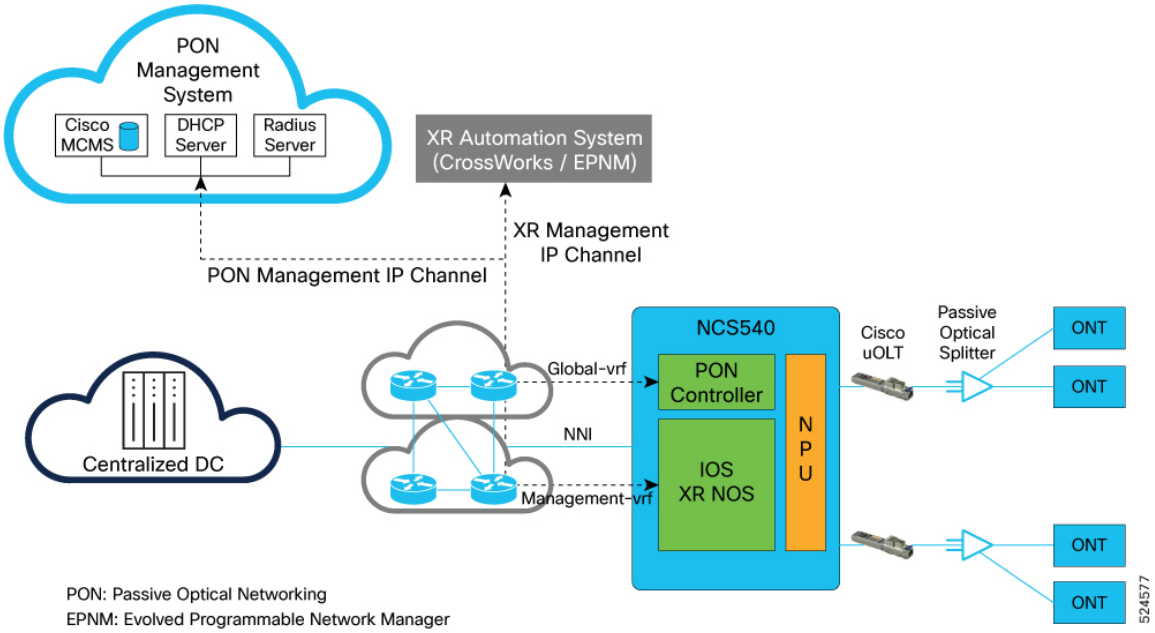
In Metro Release 1.0, the UP is based on Cisco ASR 9000 Series Routers. The onboarding and provisioning of the BNG ecosystem is done using the Cisco NSO-based solution.

For details, see Cloud Native BNG Control Plane Configuration Guide, and Cloud Native BNG Control Plane User Guide.

Cisco Routed Passive Optical Networking

Cisco Routed PON is a solution created by combining the rich access routing portfolio from Cisco and a pluggable PON solution. The core of the solution is a smart SFP acting as a single port OLT, the PON controller software managing the OLT SFPs on a device, and the PON manager software managing the overall PON network deployment. The PON SFP can be plugged in to selective set of Cisco IOS XR routers (Cisco NCS 540 Series routers) on which the PON controller software is also installed. Cisco Routed PON manager is a centralized PON manager that manages several PON controllers. The connectivity between Cisco Routed PON manager and the PON controller is secured using TLS.

Figure 5: Cisco Routed PON deployment

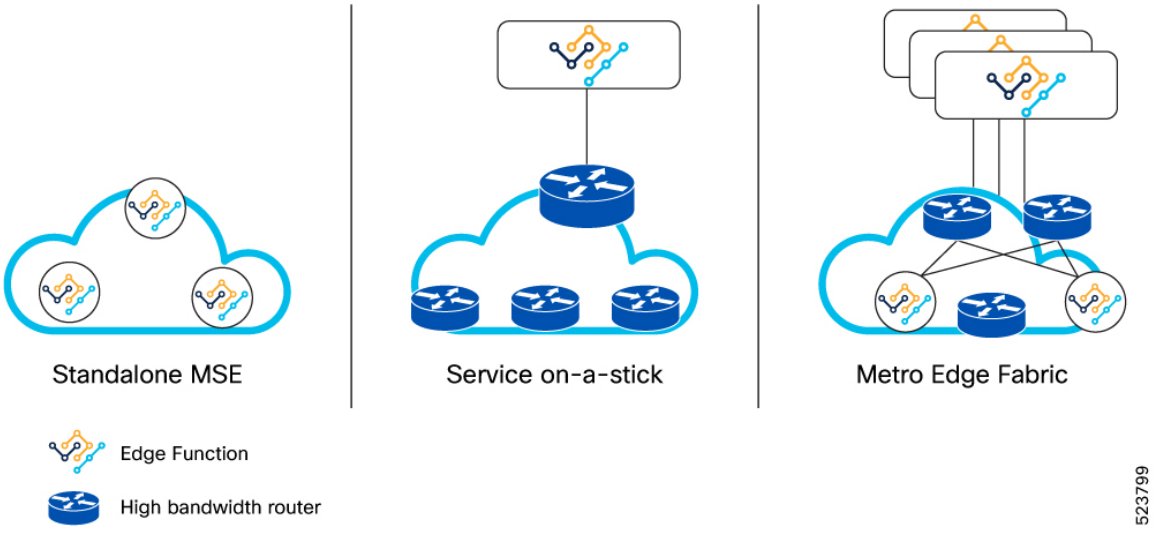


For details on Routed PON solution components, see the Components of Cisco Routed PON section.

Metro solution deployment models

The Metro solution has the flexibility to support a mix of deployments models based on factors such as service scale, service traffic bandwidth, interface types, and other network constraints.

Figure 6: Metro solution deployment models



Metro solution supports these deployment models:

- Standalone MSE: Traditional multi-service Edge (MSE) deployment model for a single Metro edge router where all functions and scale are handled at a single box level. The model is best suited for smaller edge sites which do not need the higher service scale of the Metro Edge Fabric distributed solution. The metro solution enables distribution of devices with MSE functions deeper into the network, increasing overall network efficiency.
- Service on-a-stick: Out-of-line network functions model where the edge services are taken out from the main forwarding path and hosted elsewhere, thereby decoupling the transport and service architectures. The model is ideal for deployments with a high volume of low bandwidth services compared to the volume of transit traffic through the aggregation router.
- Metro Edge Fabric: Scale out Edge services infrastructure model which is a composition of multiple routers in a leaf-spine architecture. The leaf nodes themselves perform the Edge functions of a traditional MSE router but using a distributed horizontally scaled approach as opposed to a vertically scaled approach. There is flexibility in the nodes deployed as a fabric spine, with the primary attribute being they do not terminate many services and are primarily used as transit. The spine nodes could be new nodes in large aggregation sites or re-purpose core nodes already deployed. The key point of the fabric model is horizontal service edge scaling and simplified management using automation.

All models support the *Edge as-a-function* concept where a service Edge function can be deployed anywhere in the network.

For details on the deployment options for these models, see the Metro Edge Fabric deployment options, on page 13 section.

Network infrastructure deployment

Apart from the traditional network deployments, the Metro solution architecture covers these new network infrastructure deployments:

- Metro network infrastructure: Modern underlay infrastructure including routing control plane, overlay service signaling, and timing using Precision Time Protocol (PTP), and Synchronous Ethernet (SyncE)
- Edge as-a-function: Edge features at any place in the network (PIN)
- Service on-a-stick: Out-of-line network functions
- Metro Edge Fabric: Scale out Edge services infrastructure

Metro network infrastructure

These are the Metro network infrastructure requirements for all devices participating in the Metro solution.

- **Common packet transport**: The underlay control and forwarding plane technologies used to support all network services. The common packet transport updates the underlay design outlined in Cisco Converged SDN Transport design. The underlay network is built upon proven technology such as IS-IS for v4 or v6 IGP and Segment Routing using the SRv6 or SR-MPLS data plane.
- **Base infrastructure**: Additional protocols and technologies outside the scope of routing and forwarding required in the infrastructure to support network services. The solution architecture utilizes PTP G.8275.1 with SyncE for timing distribution, supporting G.8275.2 interoperability wherever required for endpoints requiring G.8275.2.

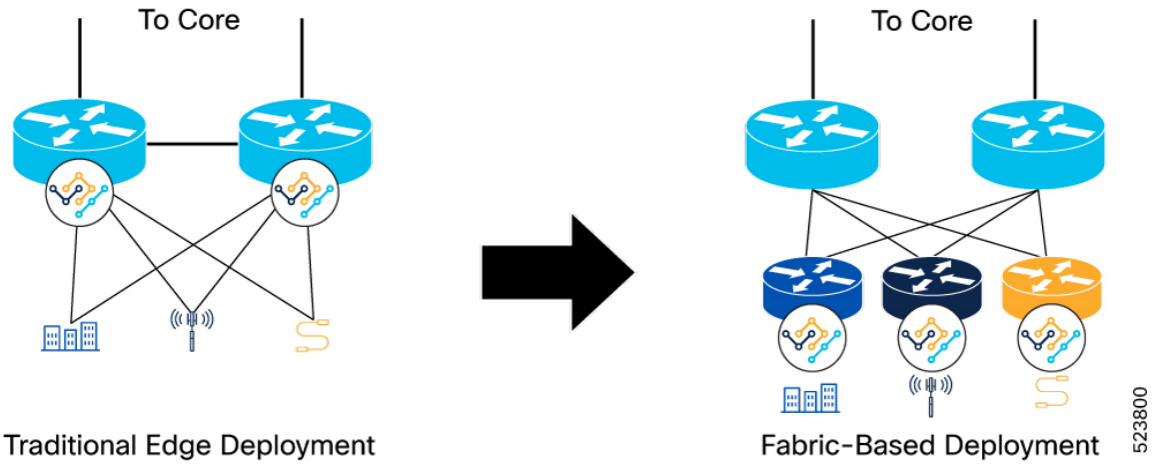
- **Base device automation**: Device support for APIs and protocols required to properly automate the network. The solution architecture leverages gNMI-based telemetry using OpenConfig and Cisco native models to monitor all aspects of the network. The interface is used by automation tools such as Crosswork Network Controller and Provider Connectivity Assurance to provide end-to-end infrastructure assurance.
- **Base operations**: Additional operational tooling and features that are required for operators to properly monitor and troubleshoot the network infrastructure and the services using the network.
- **Base security**: Base criteria for securing the network infrastructure. This does not cover service security.
- **Base QoS**: Base set of CoS and QoS feature support to ensure that network service SLAs are met across the network.

Edge as-a-function

The edge as-a-function deployment model removes the traditional location of edge services and allows edge to be served at any point in the network. That is, any router at any place in the network can act as an MSE device that is limited only by the scale required. This is becoming critical and common as higher touch edge services must be distributed to scale. Also, centralizing the edge services is no longer feasible due to service bandwidth growth.

The figure shows the evolution of fabric-based deployment from traditional Edge deployment.

Figure 7: Edge network evolution



The table lists the characteristics of traditional edge deployment and Fabric-based edge deployment models.

Table 2: Traditional edge deployment vs. Fabric-based edge deployment

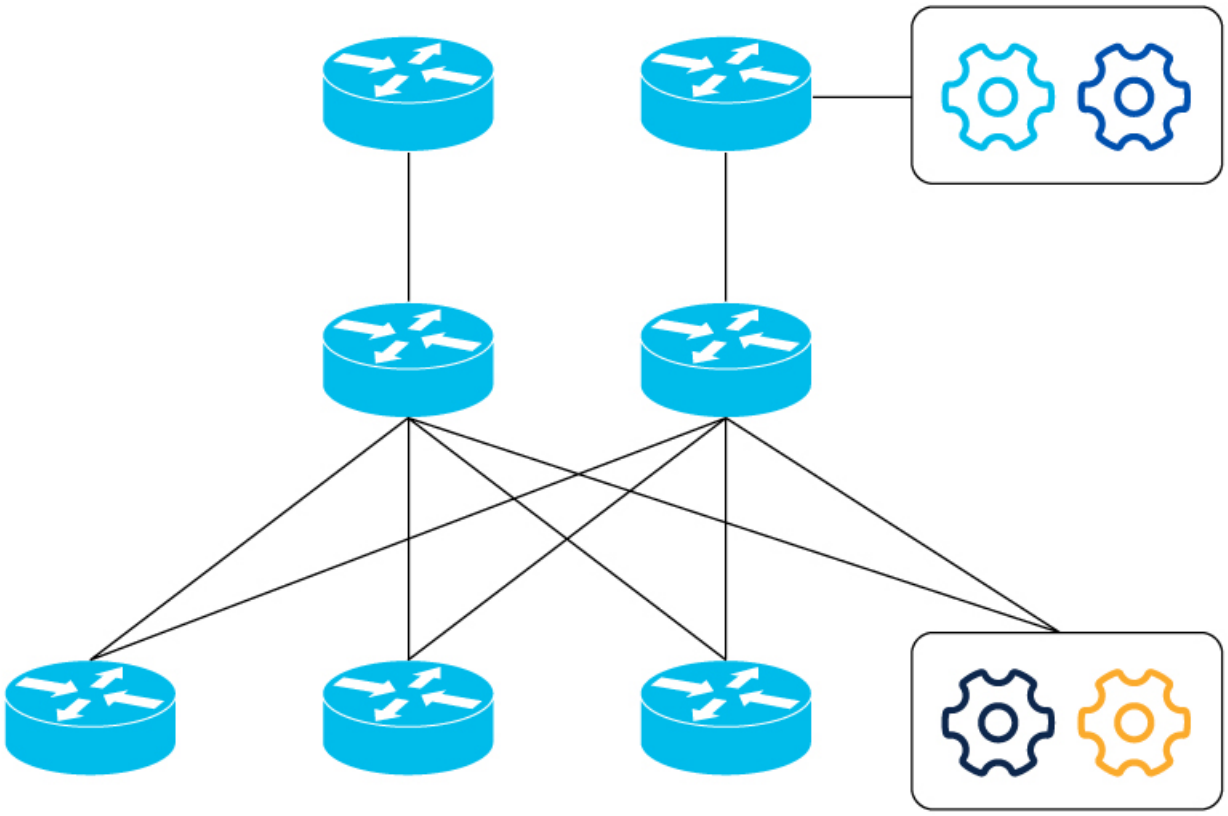
Traditional Edge deployment	Fabric-based Edge deployment
Large modular chassis	Service edge lead based on the function
Multi-service termination	Prime device for service edge
L2 or L3 aggregation that is common between end sites and edge PEs	Edge leaf can be in the same site or deeper in the network

Service on-a-stick

The service on-a-stick deployment is one where service functions are not performed on devices inline between the edge and core of the network. Instead, they are performed on devices out-of-line.

It is inefficient, and cost and power prohibitive to build higher touch network services into the distributed routing hardware when it may not be applicable to all deployments. You can do service on-a-stick deployment either as part of an edge fabric or a more centralized architecture where ingress and egress from the network function are connected to upstream routers. Service scale and bandwidth require a more distributed approach, while compute intensive functions may require them to be located at a centralized data center. The Metro architecture supports both these deployment options.

Figure 8: Service on-a-stick deployment model



Service on-a-stick deployment model is suitable for these functions and services:

- High-touch functions that are applicable only to a small subset of traffic

  For example, a NAT appliance capable of handling exception traffic.

- High-touch services that require high performance and scale, such as firewall deployments, where firewalls are placed *on a stick* to perform security functions

In Metro Release 1.0, the service on-a-stick deployment is applicable to BNG for subscriber services.

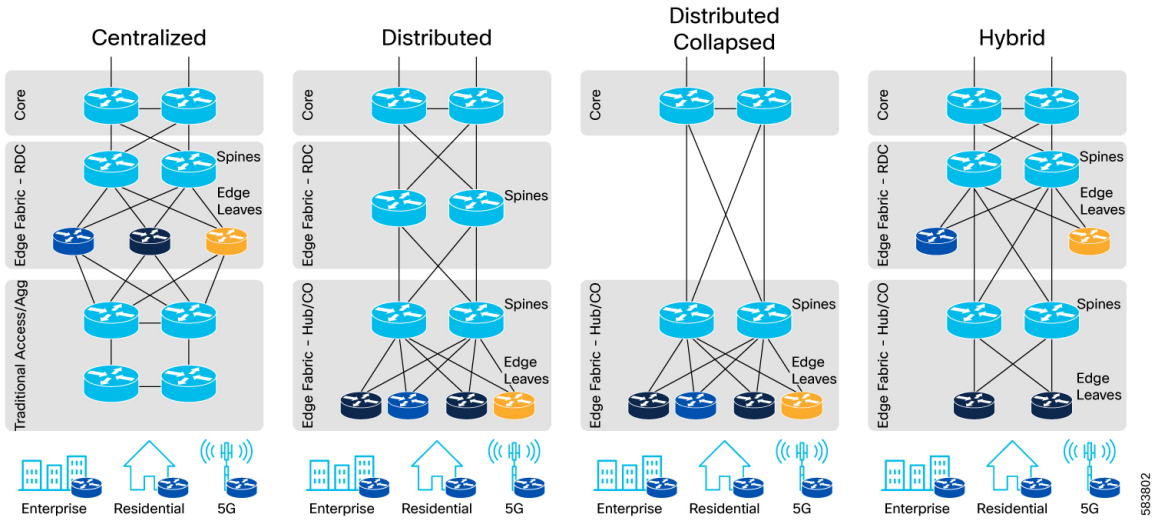
Metro Edge Fabric

Creating a network capable of serving a variety of use cases on a common fabric is best served by decomposing a modular chassis into a set of leaf nodes with a specific service role. The disaggregated Edge Fabric functions as a leaf-spine topology with edge service facing ports located on leaf devices connected through fabric ports to an upstream aggregation node. For more details on the Metro Edge Fabric components, see the Metro Edge Fabric, on page 6 section.

Metro Edge Fabric deployment options

Metro Edge Fabric can have different deployment options based on various factors such as space, power, geographic diversity, or aggregate bandwidth required for specific services. Certain service types such as business edge services lend themselves to being distributed all the way to the access portion of the network. Other types of services such as value-added inline services which are high compute and lower bandwidth maybe be better suited in a centralized deployment model.

Figure 9: Metro Edge Fabric deployment options



These are the main deployment options for Metro Edge Fabric:

- **Centralized**: This option follows the centralized MSE approach by housing the edge functions within a regional or core data center but distributing the edge services across a fabric instead of a traditional large modular MSE router. This case is ideal for types of services which are high compute and lower bandwidth such as value-added inline services.
- **Distributed**: This option fully distributes services to what would traditionally be the access or aggregation level of the network. This case is ideal for most service termination types such as business edge services (L2VPN, L3VPN, mobile backhaul, and dedicated internet access (DIA)) and subscriber edge services using a CUPS user plane.

  Distributed Collapsed: In this option, the spine nodes in the access or aggregation directly connect to the core routers without connecting to the intermediate spine nodes.
- **Hybrid**: This option distributes some edge services to further into the network, but also has some service termination at a centralized data center following the service on-a-stick network deployment.

Network infrastructure operations use cases

The network infrastructure operations use cases of Metro solution include network commissioning and infrastructure assurance operations.

Network commissioning

Zero Touch Provisioning

Zero Touch Provisioning (ZTP) is the method of provisioning network devices without manual intervention. ZTP helps to seamlessly onboard new devices across the network within a short span of time. ZTP also reduces the manual tasks required to scale network capacity.

Network infrastructure assurance

Infrastructure assurance refers to monitoring of the health of the base infrastructure network.

These are the factors contributing to the assurance of the network:

- Base Cisco IOS XR software features: SR-PM to measure latency and loss across logical links

  Starting with CNC 6.0 and the TSDN 6.0 there is an example function pack to configure and deploy performance measurement profiles across the network. This can be used to maintain consistent performance measurement definitions across the network.

- Provider Connectivity Assurance (PCA) container-based agents to measure network performance (latency, loss, and jitter) between edge devices to millisecond precision
- PCA compute SFP+ for higher accuracy to microsecond resolution when needed
- CNC to monitor and visualize the performance (bandwidth and latency) of the network
- PCA to aggregate infrastructure assurance data

These are the various network infrastructure assurance use cases of Metro solution:

- Segment Routing performance monitoring (SR-PM)
- SRv6 integrated performance measurement (SR-IPM)
- SRv6 path tracing
- SRv6 traffic accounting
- Cisco Provider Connectivity Assurance

Segment Routing performance monitoring

Segment Routing performance monitoring (SR-PM) is enabled to perform link monitoring for the end-to-end metro network to support both infrastructure monitoring and end-to-end TE path monitoring. Latency measurement is performed on all links as per the hardware capability.

SRv6 integrated performance measurement

SRv6 integrated performance measurement (SR-IPM) supports loss, latency plus jitter, and liveness detection by using high frequency TWAMP packets between SRv6 endpoints. This feature is available on Cisco ASR

9000 Series Routers, and Cisco 8000 Series fixed routers and modular routers with Q200-based Silicon One ASICs.

SRv6 path tracing

SRv6 path tracing uses a small IPv6 header of 3 bytes that is added to the transit packets to record the outgoing interface, time, and load at each hop. At the head-end node these probe packets are sent out to each ECMP path across the network. This is done to see a true view of the network even when ECMP is utilized at each hop along the path between the source and the destination nodes. The data can then be used by various tools to analyze the health of the network.

SRv6 traffic accounting

SRv6 traffic accounting is applicable to infrastructure monitoring and long-term network capacity planning. SRv6 traffic account provides statistics for each egress locator per egress interface. That is, if ECMP is available on the head-end node, the statistics per Locator and OIF is recorded and made available through both CLI show commands and model-driven telemetry. The SRv6 traffic accounting feature is available on Cisco ASR 9000 series routers, and Cisco 8000 Series fixed routers and modular routers with Q200-based Silicon One ASICs.

Cisco Provider Connectivity Assurance

Metro solution leverages the existing Cisco Provider Connectivity Assurance (PCA) components for automation workflows.

Cisco Provider Connectivity Assurance use cases

These are the Cisco PCA use cases of Metro solution:

- Monitor loss, latency, and jitter between infrastructure service endpoints (PE to PE).
- Monitor loss, latency, and jitter.

Cisco Provider Connectivity Assurance deployment

These are the Cisco PCA deployment use cases of Metro solution:

- PCA hardware-based agents at each endpoint- both inline and out-of-line
- PCA hardware-based agents at one endpoint with Cisco device acting as a reflector
- Use of PCA to monitor the overall health of the metro network including any fabric elements

Network infrastructure security

The network infrastructure security use cases of Metro solution include base device security, and edge protect use cases.

Base device security

The base architecture provides a level of device security to protect against common device-level attacks.

Edge Protect DDoS

Cisco Edge Protect DDoS is a solution that is used to add another layer of protection against DDoS attacks to the network infrastructure.

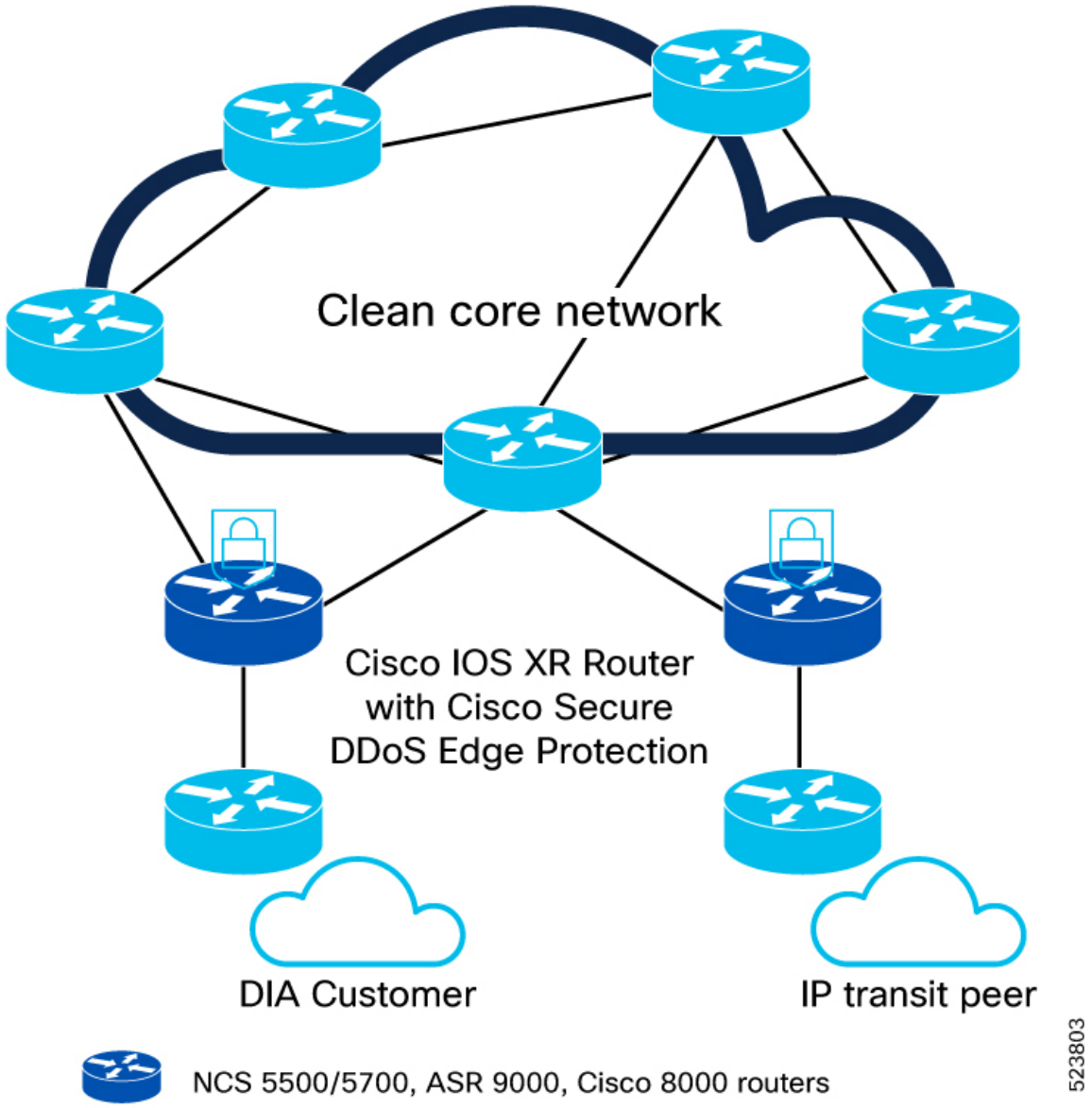
In the initial release of Metro, these use cases are covered by Edge Protect—Mobile Edge Protect on Cisco NCS 540 Series Routers, and Edge Protect for peering use cases on the Cisco NCS 5500 and NCS 5700 Series Routers.

Edge Protect is a small, lightweight, and containerized DDoS detection engine, running inside Cisco IOS XR operating system on selected Cisco routers. As a feature of the router itself, the solution provides out-of-path, full detection, and granular mitigation capabilities against volumetric DDoS attacks. This allows the router to become the first line of defense against DDoS attacks where attack traffic can be blocked in-place and no longer needs to be backhauled to dedicated scrubbing infrastructure. This gives you the opportunity to extend DDoS protection to the edge of the network which would otherwise be cost-prohibitive and negatively impacting the application latency and the quality of experience.

You can apply this solution to the existing install base and to new router deployments without requiring any additional hardware.

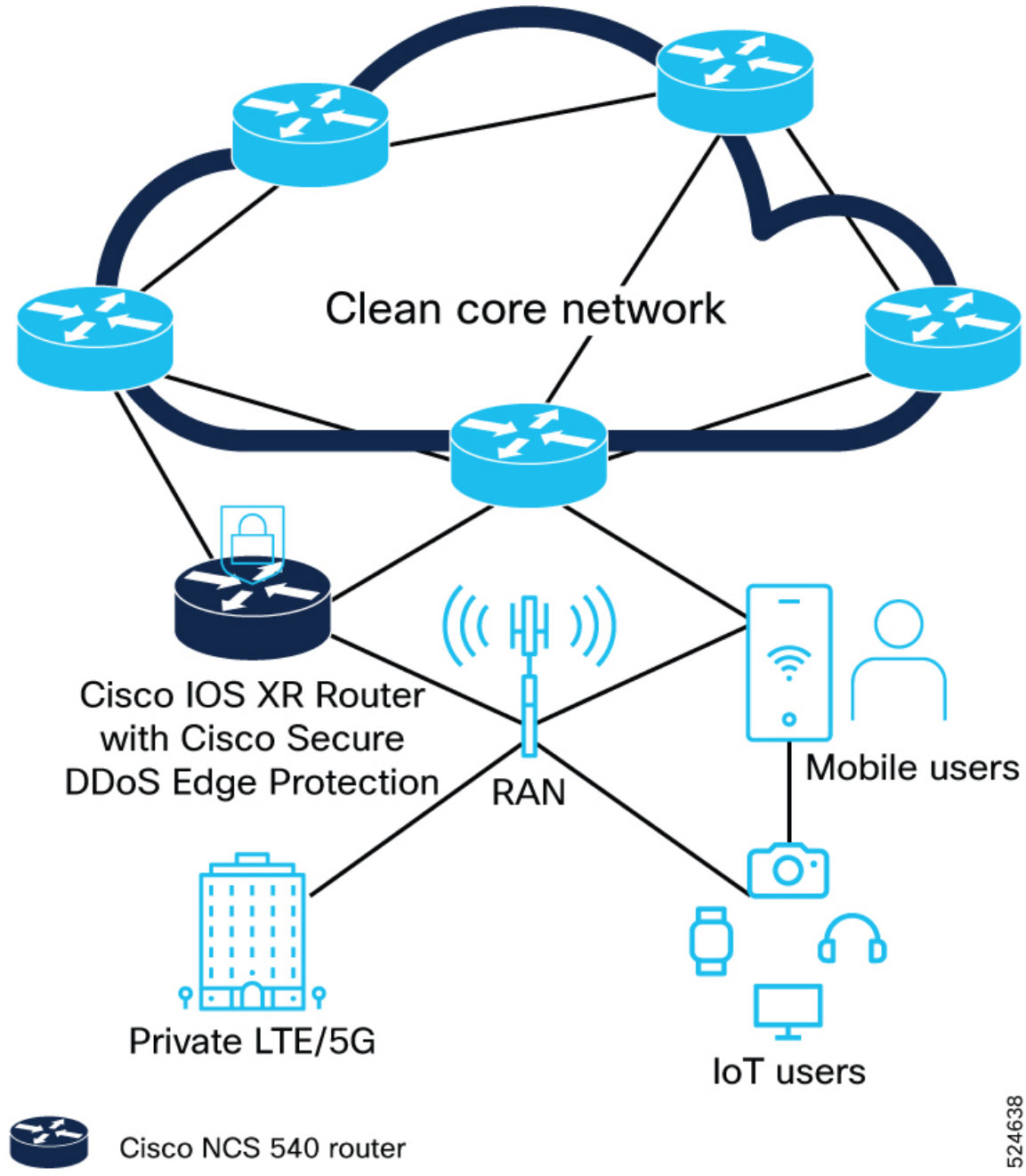
The figure shows the peering and internet custom use case for Edge Protect.

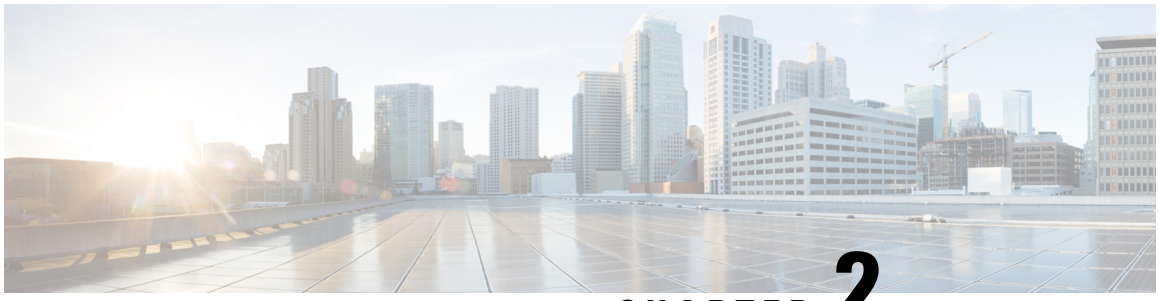
Figure 10: Peering and internet custom use case for Edge Protect



The figure shows the mobile network protection use case for Edge Protect.

Figure 11: Mobile network protection use case for Edge Protect





CHAPTER 2

Metro Solution Components

This chapter covers the main components, the supported Cisco IOS XR OS products, and the automation components of Cisco Metro solution.

Main components of Metro solution

The table lists the main hardware and software components of Metro solution and their compatible versions.

Table 3: Compatibility matrix for Metro solution components

Metro solution component	Hardware or software component	Version
Cisco routers	Cisco ASR 9000 Series Routers Cisco NCS 540 Series Routers Cisco NCS 5500 Series Routers Cisco NCS 5700 Series Routers Cisco 8000 Series Routers Cisco 8700 Series Routers Cisco Catalyst (SD-WAN)	Not applicable
NOS for Cisco ASR 9000, NCS 540, NCS 5500, NCS 5700, Cisco 8000, and Cisco 8700 Series Routers	Cisco IOS XR Software	24.4.1
NOS for Cisco Catalyst 8500 Series Edge Platforms	Cisco IOS XE Software	17.15
Edge Fabric Management	Metro Fabric Manager Function Pack	1.0
DDoS Controller	Cisco Secure DDoS Edge Protection	24.07.09.2976

Metro solution component	Hardware or software component	Version
IP Controller	Cisco Crosswork Network Controller	7.0.1
Multi-Layer Controller	Cisco Crosswork Hierarchical Controller	9.0
Network Services Orchestrator	Cisco Crosswork Network Services Orchestrator	6.1.11.2
Workflow Management	Cisco Crosswork Workflow Manager	1.2
SD-WAN Controller	Cisco Catalyst SD-WAN Manager	17.15.1
Provider Connectivity Assurance Sensor Management	CPCA Sensor Control CPCA Orchestrator	22.x
PON Management	Cisco Routed PON Manager	5.0
CUPS Control Plane	Cisco Cloud Native BNG (cnBNG) Control Plane	2024.04.0 with Cloud Native Deployment Platform (CNDP) 24
CUPS User Plane	Cisco Cloud Native BNG (cnBNG) User Plane: Cisco ASR 99XX modular chassis with Cisco ASR 9000 5th generation High Density Ethernet line cards: ASR 9902 ASR 9903	24.4.1
CnBNG CFP for Day-0 and Day-1 Management	CNBNG SMI-NSO CFP	2024.04.0

Supported Cisco IOS XR OS products for Metro solution

The table lists the supported Cisco IOS XR OS products for Metro solution.

Product Series	Product ID
Cisco ASR 9000	ASR 9902 ASR 9903
Cisco 8000 (Q200-based)	8201-24H8FH 8201-32FH 8202-32FH-M
	Cisco 8608 (Centralized): 86-MPA-14H2FH-M 86-MPA-24Z-M 86-MPA-4FH-M

Product Series	Product ID
Cisco 8000 (P100-based)	8711-32FH-M 8212-48FH-M
Cisco 8000 (K100-based)	8712-MOD-M
Cisco NCS 5500	NCS 55A1: NCS-55A1-24Q6H-S NCS-55A1-24Q6H-SS (MACsec)
Cisco NCS 55A2, NCS 57C3	NCS-55A2-MOD-SE NCS-57C3-MOD-SE-S NCS-55A2-MOD-S NCS-57C3-MOD-S
Cisco NCS 5700	NCS-57B1-5DSE-SYS NCS-57B1-6D24-SYS NCS-57D2-18DD NCS-57C1-48Q6-SYS
Cisco NCS 5500 line cards	NC57-48Q2D-S NC57-48Q2D-SE-S NC57-36H6D-S
Cisco NCS 540	N540-24Z8Q2C-SYS N540-ACC-SYS N540-24Q8L2DD-SYS N540X-16Z4G8Q2C-D/A N540-28Z4C-SYS-D/A
Cisco IOS XRv 9000	Cisco IOS XRv 9000
Cisco IOS XRd virtual router	Cisco IOS XRd vRouter

Automation components of Metro solution

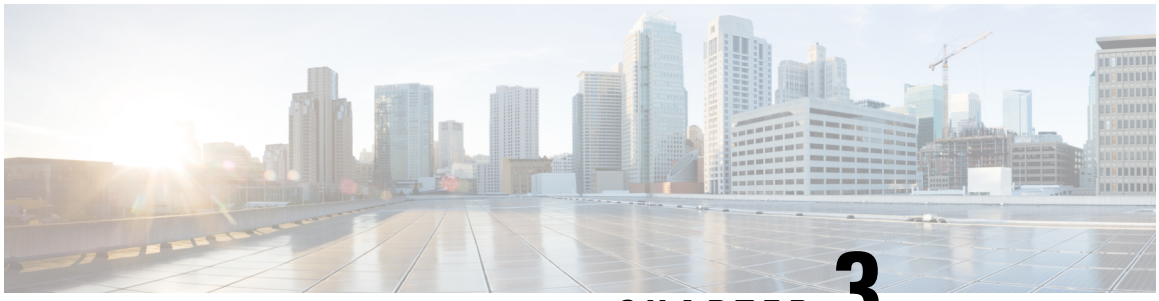
The table lists the main automation components of Metro solution and their compatible versions.

Automation component	Component version
Cisco Crosswork Network Controller	7.0.1
Cisco Crosswork Hierarchical Controller	9.0
Cisco Crosswork Network Services Orchestrator	6.1.3.1
Cisco Crosswork Workflow Manager	1.2
Cisco Cloud Native BNG (cnBNG) Control Plane	2024.04.0 with Cloud Native Deployment Platform (CNDP) 24
Cisco Cloud Native BNG NSO SMI Deployer	2024.04.0
Crosswork Planning	7.0.1
Cisco Provider Connectivity Assurance	24.2
CX Fabric Manager	1.0

Related documentation

For details on individual components of Metro solution, see the related documentation listed here.

- Cisco Catalyst SD-WAN
- Cisco Crosswork Hierarchical Controller
- Cisco Crosswork Network Controller
- Cisco Crosswork Network Services Orchestrator
- Cisco Crosswork Workflow Manager
- Cisco Routed Optical Networking
- Cisco Routed Passive Optical Networking
- Cloud Native BNG Control Plane
- Cloud Native BNG User Plane
- Cisco Provider Connectivity Assurance



CHAPTER 3

Metro Solution Use Cases

This chapter describes the primary use cases of the Metro solution. The solution is flexible enough to cover simple and advanced provider service use cases.

• Network services use cases, on page 23

Network services use cases

Network services are the key drivers for Metro evolution. These are the applicable network infrastructure use cases of Metro solution for each service type:

- Subscriber edge:
  - wireline subscriber termination, and
  - last mile access aggregation.
- Converged transport:
  - 4G and 5G mobile network IP transport , and
  - virtual CSR or virtual PE.
- Enterprise business edge connectivity services:
  - middle mile infrastructure service
  - cloud interconnect and on-ramp
  - SD-WAN interconnect, and
  - Private Line Emulation (PLE).
- SPDC centralized and edge DC interconnection
- External interconnect:
  - public cloud interconnection, and
  - internet peering and content delivery.

Subscriber Edge

This section covers the details of delivering IP services to access network wireline subscribers including next-generation BNG using a CUPS architecture.

Wireline subscriber termination

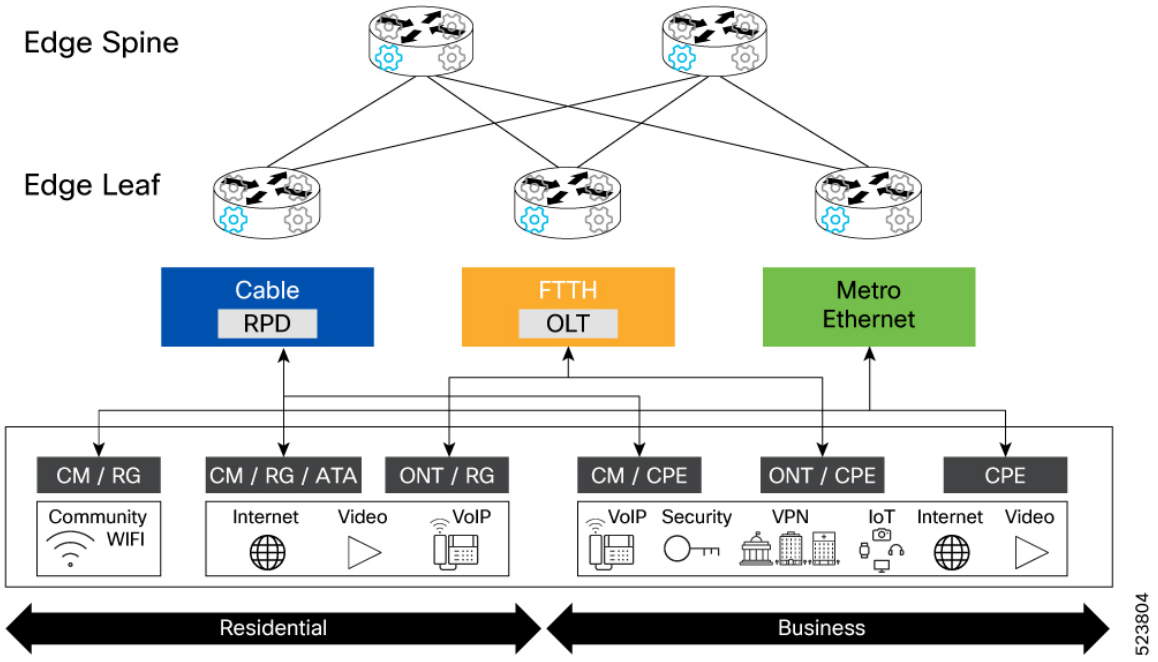
Traditional BNG deployments provide physical BNG functionalities to subscribers in edge routers (primarily, Cisco ASR 9000 Series Routers). This model has limitations in scaling out as the subscriber scale increases. The separation of BNG control plane (CP) and user plane (UP) allows us to create a scalable and resilient subscriber management platform. The breakout to cloud point-of-presence (PoP), edge service, and so on happens after the subscriber termination. Services like CGNAT is supported as in-line service or as service-on-a-stick model using Cisco Virtualized Services Module (VSM) cards.

Last mile access aggregation

Last mile access aggregation refers to the edge leaf device acting as an aggregation for access technologies using their own devices connected upstream through Ethernet. In most cases these devices are PON OLTs or CIN Remote PHY or Remote MACPHY nodes but extend to other access technologies such as Wi-Fi and Active Ethernet.

The image shows the aggregation of these nodes at the edge of the network.

Figure 12: Last mile access aggregation



This type of aggregation also applies to virtual network functions in the network which do not reside within a dedicated data center environment. For example, a virtualized cable modem termination system (vCMTS), which is a virtual element terminating cable subscribers typically deployed as part of the SP network and not in a separate DC environment.

Converged transport

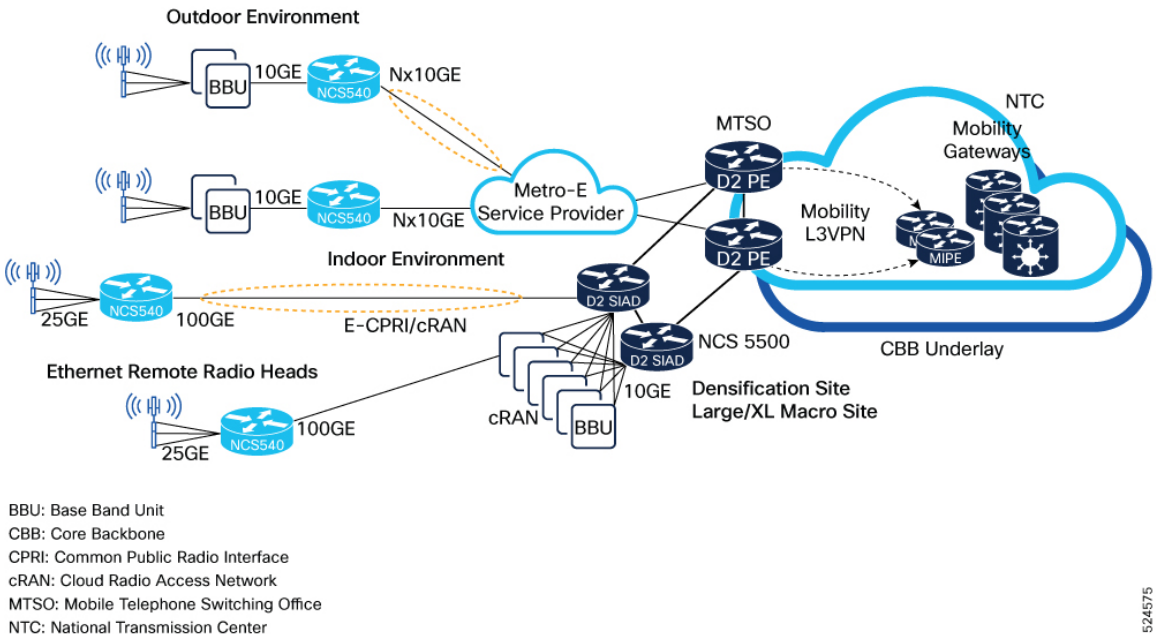
The Metro solution supports these use cases on converged transport.

4G and 5G mobile network IP transport

Historically we have played the role of backhaul IP transport provider. The dis-aggregation of radio access network (RAN) opens opportunity in new stringent transport networks like midhaul and fronthaul. In future, the resource demanding applications will drive the need for guaranteed services in transport network through network slicing and service assurance.

The current Metro solution relies on the existing cell site routers (CSRs) and pre-aggregation routers to provide access transport services. These devices are added as leaf nodes in Metro fabric to provide last mile cell site connectivity. Overlay services such as SR-EVPN in metro fabric carries the services originated from cell site to packet core, distributed unit (DU), centralized unit (CU), or security gateway (SecGW) depending on the network configuration.

Figure 13: Mobile backhaul



BBU: Base Band Unit
CBB: Core Backbone
CPRI: Common Public Radio Interface
cRAN: Cloud Radio Access Network
MTSO: Mobile Telephone Switching Office
NTC: National Transmission Center

524575

Metro Release 1.0 focuses on Cisco NCS 540 series routers as the CSRs and Cisco NCS 5500 series routers, and Cisco NCS 5700 series routers as the pre-aggregation routers.

Virtual CSR or virtual PE

Cisco IOS XRd vRouter is a containerized version of Cisco IOS XR control plane and high-performance Vector Packet Processing (VPP)-based data plane. The Metro solution allows providers to deploy high performance routing within a cloud-native network function (CNF) environment located at centralized or edge data centers. Two applications for the XRd vRouter are as a virtual Cell Site Router (vCSR) and virtual Provider Edge (vPE) device. The Metro solution covers both these use cases.

Enterprise business edge connectivity services

Enterprise services cover the use of traditional network service types to deliver end-to-end connectivity or overlay services to enterprise endpoints. These services are not specific to communications service providers (CSPs). Many enterprise and government networks are built to support internal connectivity in much the same way as a service provider network. This section is agnostic to the last mile access network being used; it could be wireline (such as fiber, copper, and so on) to the premise, within a data center facility, or using mobile, or satellite technology.

Table 4: MEF service types and carrier network services

MEF service type	Carrier network service
E-LINE	L2VPN EVPN-VPWS
E-LAN	L2VPN EVPN ELAN
E-TREE	L2VPN EVPN ETREE
E-TRANSIT (NNI-to-NNI interconnect)	L2VPN EVPN-VPWS, ELAN, or ETREE
E-ACCESS (NNI-to-UNI interconnect)	L2VPN EVPN-VPWS, ELAN, or ETREE
L3VPN	VPNv4, or VPNv6 L3VPN
Internet Access Service (Direct Internet Access)	DIA (GRT or internet in a VRF)

Middle mile infrastructure service

The Network-to-Network Interface (NNI) use cases listed above apply to several use cases. Middle mile connectivity services that help to interconnect customer endpoints using P2P service over their converged packet network fall under the category of E-TRANSIT. An example is North America alternative access vendor (AAV), which is the common name for a service provider interconnecting a cell site location to a mobile core location when the mobile carrier does not have their own infrastructure to do so.

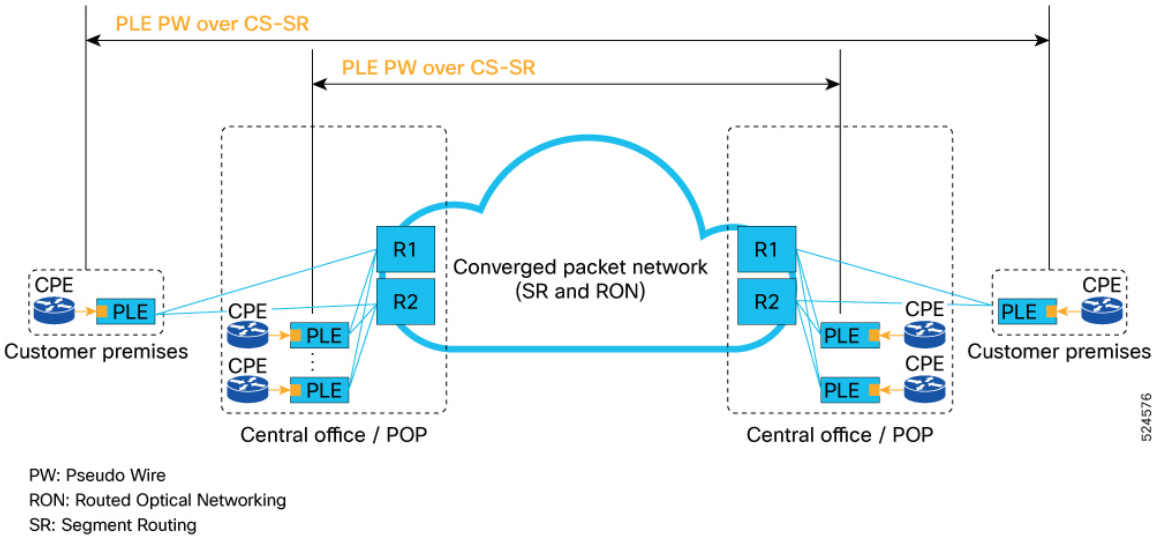
Cloud interconnect and onramp

The use case of a CSP connecting their customers to a cloud interconnect provider is another type of enterprise connectivity use case. In this case, the CSP may be the last-mile provider for the enterprise and have connectivity to worldwide interconnect providers who are responsible for interconnecting the customer to a specific cloud provider or data center using their network. This use case falls under the E-ACCESS use case since one UNI leg of the connection is considered on-net for the service provider.

Private Line Emulation

Private Line Emulation (PLE) is a technology to support bit-transparent Ethernet, OTN, SONET or SDH, and Fiber Channel services across an IP packet transport network. At present, the PLE L2VPN circuits are carried over an IP or MPLS network.

Figure 14: Private Line Emulation



PW: Pseudo Wire
RON: Routed Optical Networking
SR: Segment Routing

The solution primarily relies on PLE gateways placed at customer premises to transform private line bit stream to SR-MPLS payload. Cisco Cross Network Controller (CNC) sets up end-to-end guaranteed traffic engineering tunnel (SR-CS) to remote PLE gateway to carry the PLE traffic.

SD-WAN

These are the SD-WAN use cases related to Metro solution:

- SD-WAN interconnect
- SD-WAN assurance using Cisco PCA
- SD-WAN and SR-policy integration

SD-WAN interconnect

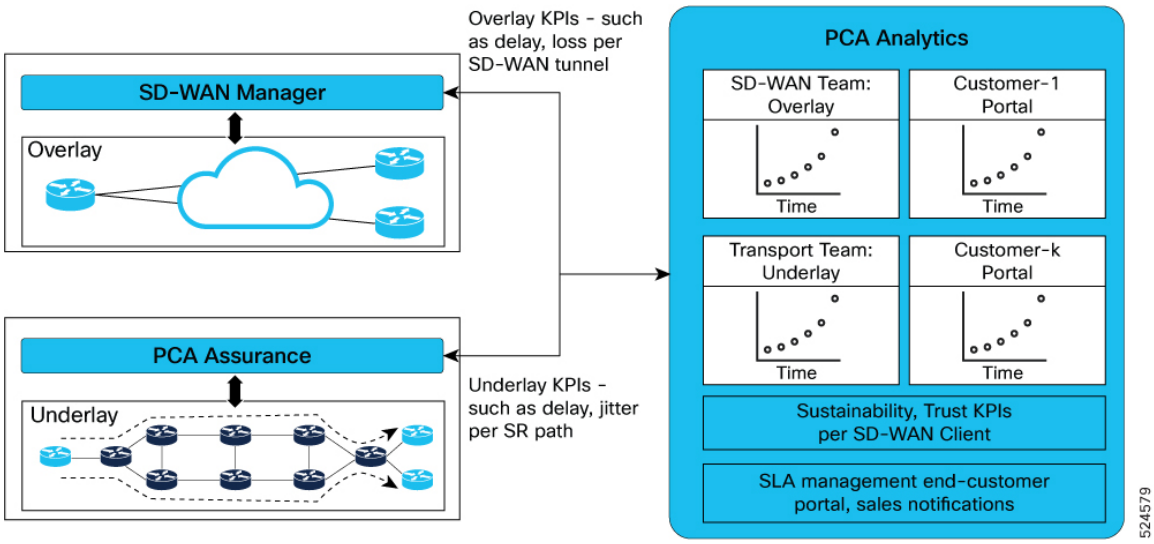
SD-WAN VPN services have become the primary way by which enterprises create private networks. These services are primarily carried over DIA from service providers. There are hybrid deployments where enterprises may have connectivity over a traditional service provider L3VPN and SD-WAN. In these cases, the service provider must provide interconnection between the provider L3VPN infrastructure and the SD-WAN infrastructure. SD-WAN interconnect covers this use case. Primarily this is done through an L2 or simple L3 hand off since the provider L3VPN and SD-WAN control plane are not interoperable.

SD-WAN assurance using Cisco Provider Connectivity Assurance

The goals of this use case include

- exposing underlay KPIs that provide business value to overlay operator
- exposing overlay KPIs that provide business value to underlay operator
- designing and conducting proof of concept (PoC) in lab set-up, and
- minimizing overlay and underlay dependencies.

Figure 15: SD-WAN visualization and assurance

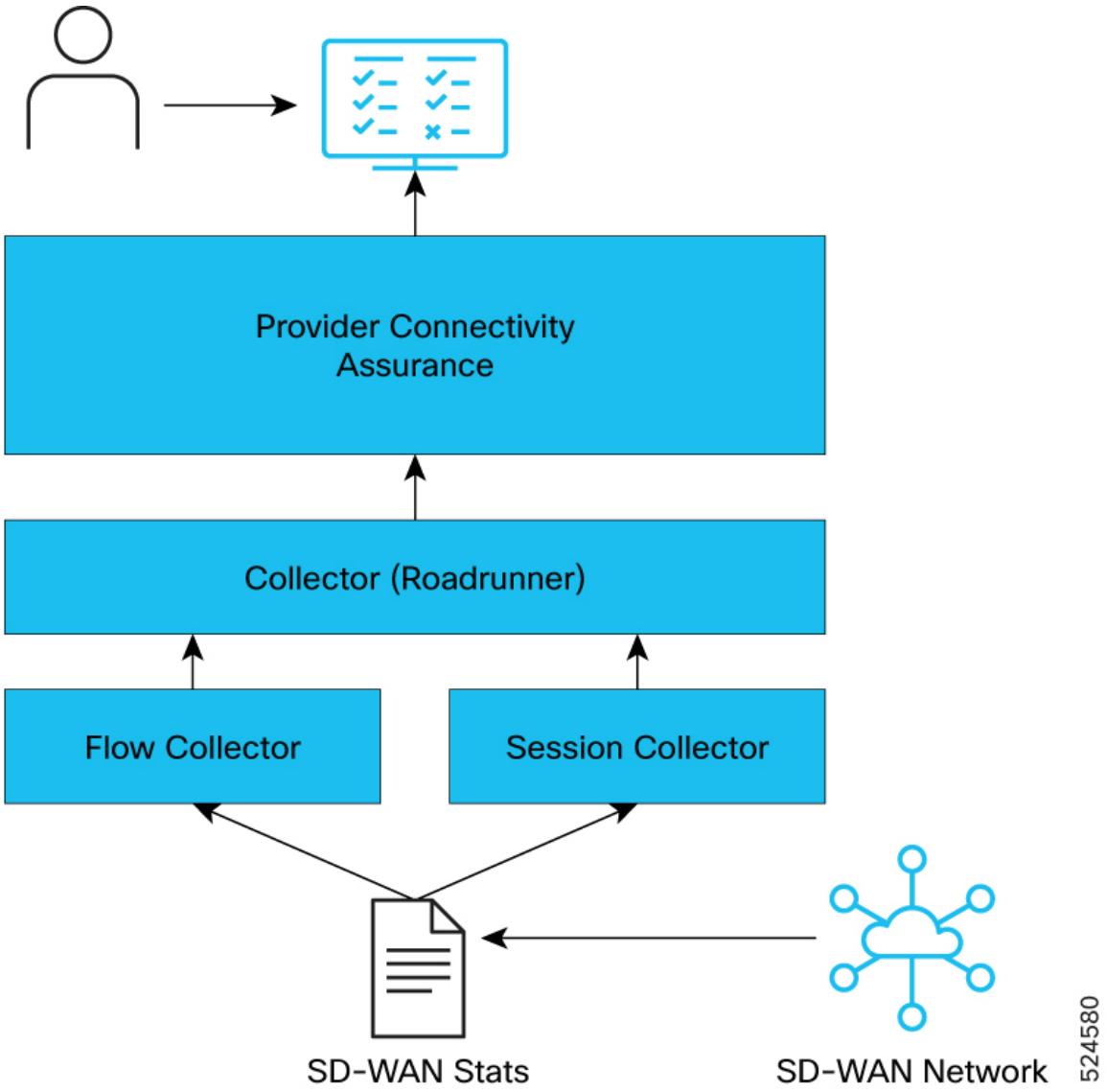


SD-WAN service assurance covers these functionalities:

- Collect overlay KPI data from Cisco Catalyst SD-WAN solution to monitor the overlay quality
- Collect KPI data from CNC for topology data and additional infrastructure-level monitoring data
- Correlate underlay and overlay data providing a complete end-to-end assurance view

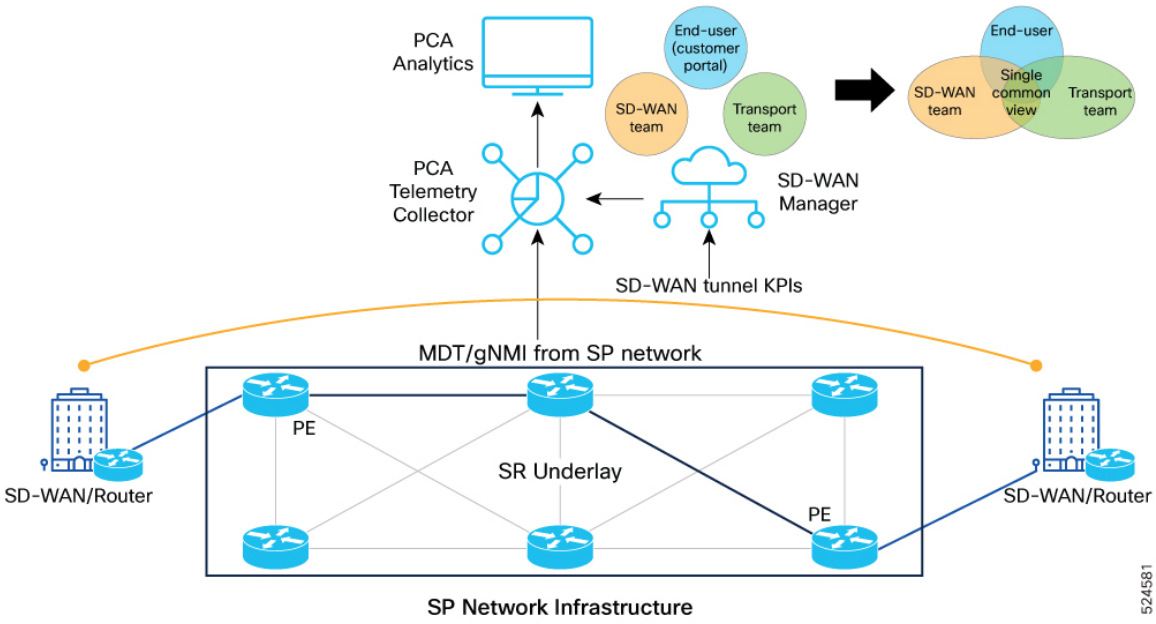
In Metro Release 1.0, SD-WAN assurance using PCA is in PoC status.

Figure 16: SD-WAN service assurance functionalities



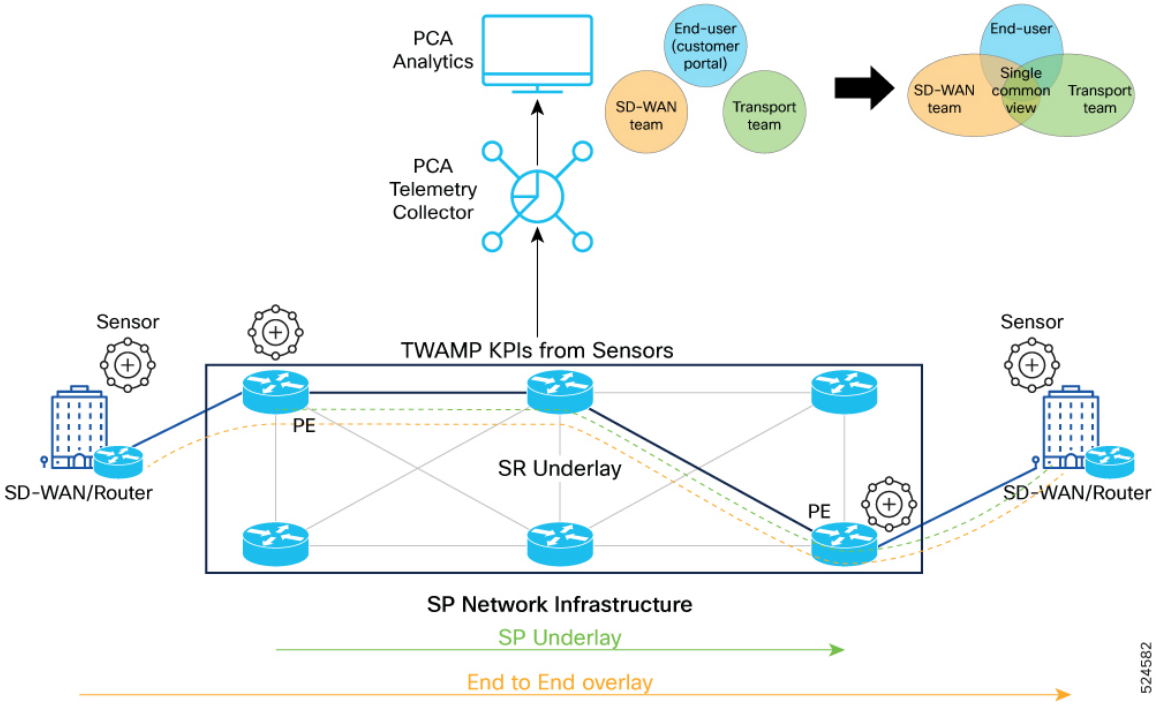
The figure shows the basic visibility and insights leveraging PCA telemetry collector and SD-WAN manager.

Figure 17: SD-WAN visualization: basic visibility



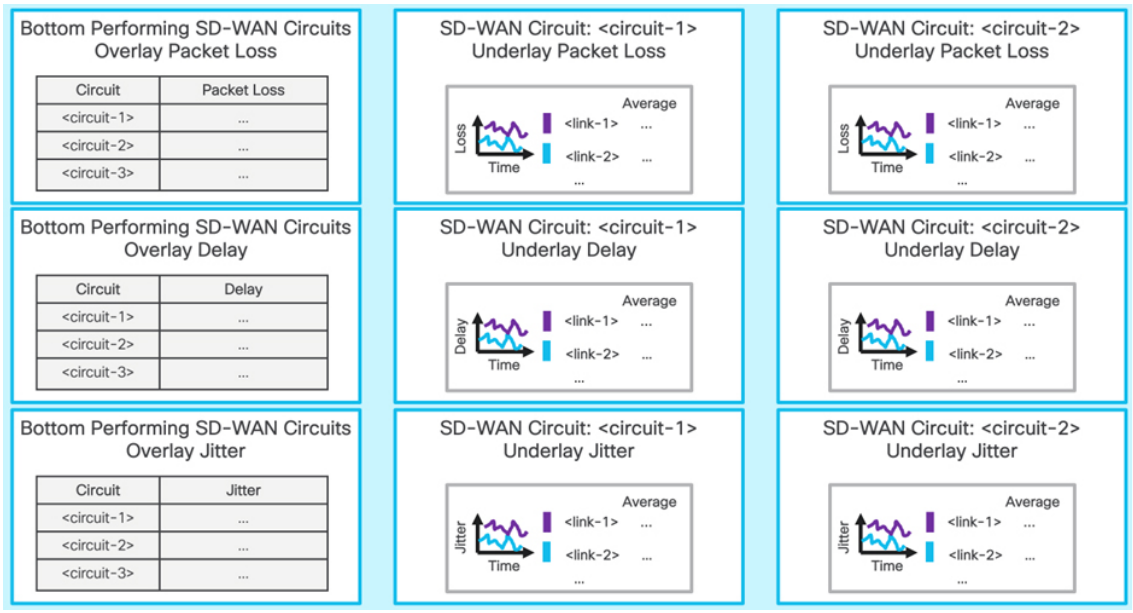
The figure shows the advanced visibility and insights leveraging PCA sensors.

Figure 18: SD-WAN visualization: advanced visibility



The figure shows a sample dashboard highlighting both SD-WAN and underlay performance data in a single unified dashboard.

Figure 19: SD-WAN visualization: sample dashboard



Benefits of SD-WAN visualization and assurance

The benefits of SD-WAN visualization and assurance include several values for transport operator, SD-WAN operator, and SD-WAN customer:

- Transport operator:
  - Per-slice monitoring and accounting
  - Per-tenant, per-circuit monitoring and accounting
  - Improved reactive and proactive Mean Time to Identify (MTTI) and Mean Time Between Failures
  - Pro-active capacity planning
- SD-WAN operator:
  - Per-SLA intent monitoring and assurance
  - Improved reactive and proactive MTTI and MTBF
  - Guaranteed SLA intent on overlay and underlay
- SD-WAN customer:
  - Guaranteed SLA intent on overlay and underlay
  - Greater visibility (for example, trust, sustainability, and SLA reporting)

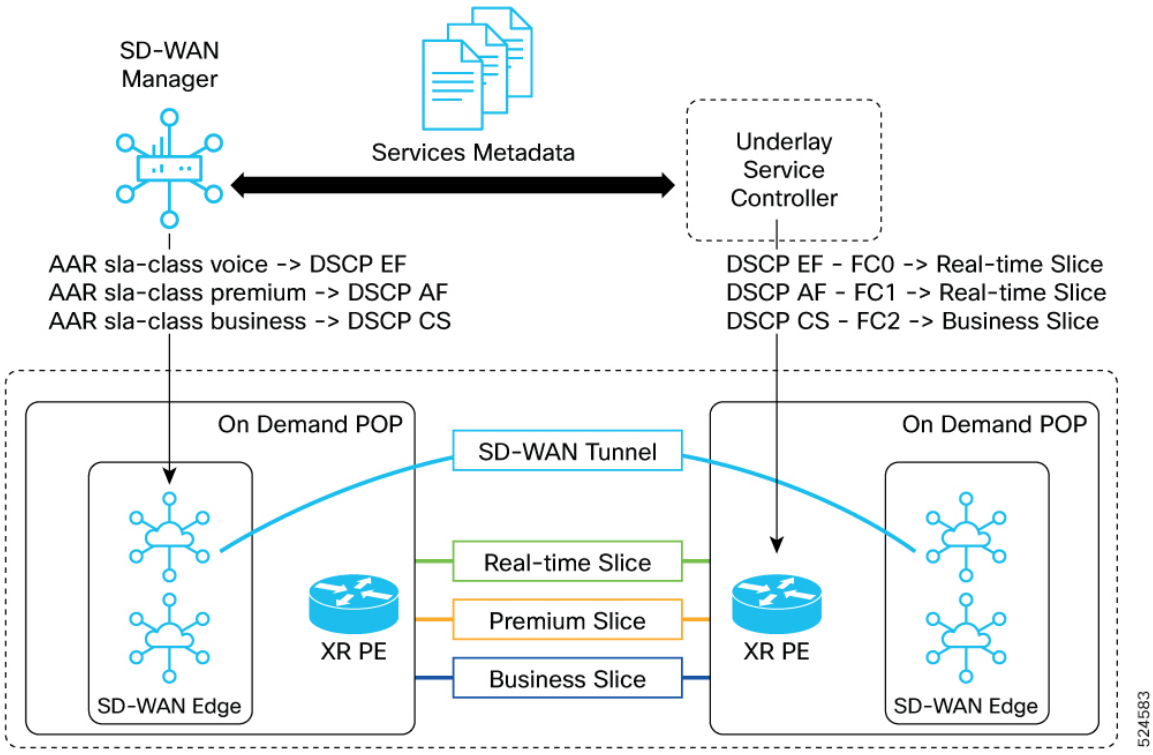
SD-WAN and SR policy integration

The objectives of SD-WAN and SR policy integration use case include

- creating business value for the SD-WAN operator by offering SLAs enabled by the underlay operator for SR and SRv6
- designing validation in lab set-up, and
- minimizing overlay and underlay operational complexity.

The figure shows the SD-WAN and SR policy integration.

Figure 20: SD-WAN and SR policy integration



Benefits of SD-WAN and SR policy integration

The benefits of SD-WAN and SR policy integration include several values for

- Transport operator:
  - New revenue streams by offering more services
  - Pay-as-you-use (per-slice monetization)
  - Unlock new SLAs by moving beyond traditional QoS
- SD-WAN operator:
  - Offer differentiated traffic steering
  - Offer more value with application intent and tighter integration with underlay services
- SD-WAN customer:

- Better value for services
- Improved application performance
- Significant reduction in downtime

SPDC centralized and edge DC interconnection

There is a resurgence in use cases requiring edge data center connectivity. The same basic connectivity is typically used in the case of a centralized data center. The interior of the data center is considered a different network domain than the metro transport network and is often connected through a border device. However, there are use cases where a single end-to-end control and data plane is used across both metro transport and data center.

External interconnect

This Metro solution use cases on external interconnect include public clould interconnection, and internet peering and content delivery use cases.

Public cloud interconnection

Public cloud interconnect refers to these use cases:

- Interconnecting the carrier or enterprise network to a local instance of a cloud provider infrastructure. For example, AWS Outpost.
- Carrier or enterprise connecting their network to the network of the cloud provider through private interconnect. For example, AWS Direct Connect, which is used by enterprises to connect directly to AWS, or by service providers aggregating the AWS connections of their customers through a connection to AWS.

Internet peering and content delivery

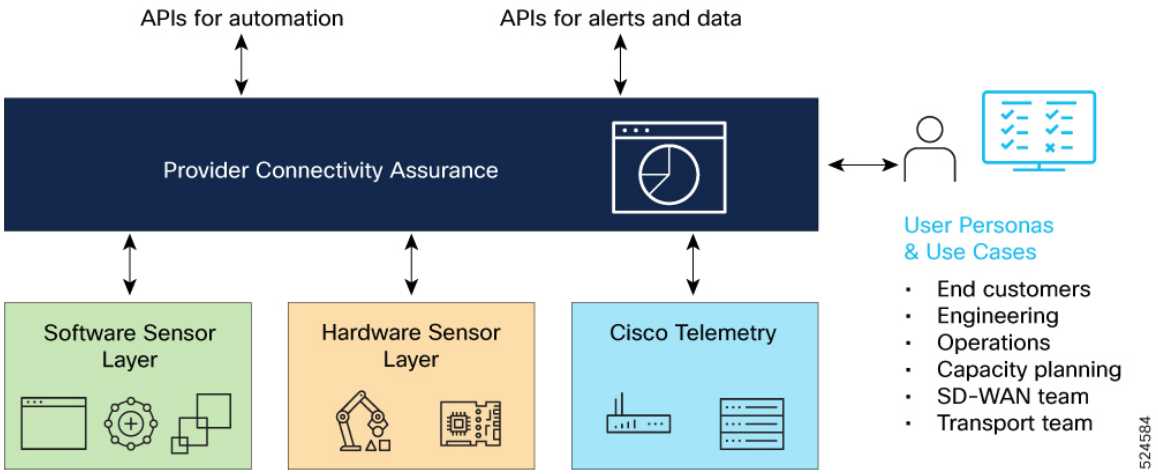
As bandwidth to subscribers continues to increase, offloading traffic from a provider network is a key method to reduce capacity costs. CDN caches are now deployed in most networks across the world, and continue to be deployed to help optimize video delivery from both on-net and off-net sources. The joint solution of Cisco and Qwilt on open edge caching is one method that providers can use to optimize traffic across the network. The way these caching clusters or traditional internet peering, or transit services are connected to the new edge network is a supported use case.

The robust family of Cisco routers can be used for internet peering within the new Metro network, or at traditional carrier peering facilities.

Network service assurance using PCA

The figure shows the Cisco Provider Connectivity Assurance (PCA) solution components for network service assurance.

Figure 21: Cisco PCA solution components for network service assurance



These are the network services assurance use cases using Cisco PCA:

- **Routed PON service assurance**: PCA adds value to the existing RPON monitoring applications such as PON Manager, by combining network infrastructure information with subscriber-specific RPON data. The PCA RPON dashboard ingests data from the core provider network as well as data from the PON manager.

  For details, see Routed PON service assurance using Provider Connectivity Assurance, on page 81.

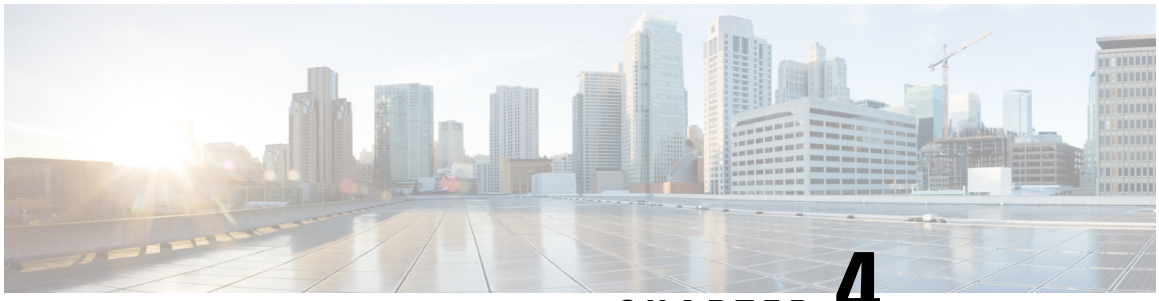
- **Business service assurance**: PCA covers this use case today, enhancing the capability of service assurance in tools like CNC. PCA sensors are deployed at the customer endpoint, sessions created per service or per SLA are probed, and then reported back to the analytics platform monitoring it per service. Within the context of the Metro solution, these existing capabilities are augmented using additional network infrastructure assurance data. The network infrastructure assurance data sources are SR-PM and additional PCA sensors.

- **SD-WAN service assurance**: SD-WAN connections are overlay services that are considered as an overlay technology. That is, the transport for the SD-WAN connection is agnostic of the underlay network with the only requirement that the underlay network provides IP connectivity between SD-WAN endpoints. In Metro solution we combine data from the overlay and underlay together to enhance visibility across the service when carried over a service provider network.

  The ability to combine and correlate data from overlay and underlay using PCA provides several positive outcomes:

  - Simplified troubleshooting of performance issues
  - Enhanced QoE by mapping high priority SD-WAN sessions to specific underlay paths
  - Ability to offer richer service types

  For details, see SD-WAN assurance using Cisco Provider Connectivity Assurance, on page 27.



CHAPTER 4

Metro Edge Fabric Manager

This chapter covers details on Metro Edge Fabric automation use cases that include fabric definition, commissioning of devices participating in an Edge Fabric, and operations such as software life-cycle management and configuration template management.

Metro Edge Fabric Manager

Fabric Manager is a container for fabric attributes, fabrics, fabric devices, and fabric connections. It also supports API calls to perform compliance actions.

Primary elements of Metro Edge Fabric Manager

These are the primary elements of Metro Edge Fabric Manager:

- Fabric definition: is a container or database that is used to store all attributes of the fabric including the devices, their roles, and the connectivity between fabric devices.
- Role definition: is a container for a set of attributes which may be applied to multiple devices in the same role.

  The role definition must contain the topology role (leaf or spine) along with all required role attributes. The role definitions are located outside of a specific Fabric instance. So, they may be applied across devices in different fabrics.
- Fabric connectivity: is a part of a specific fabric instance that is used to define the connection between all devices and contains information about the type of connection and the explicit endpoints of the connection.

Metro Edge Fabric automation

Metro Edge Fabric automation use cases

Metro Edge Fabric automation covers these use cases:

- Device fabric onboarding: that covers the commissioning of devices participating in an Edge Fabric
- Compliance operations that include
  - Fabric Software Life-Cycle Management (FSLM): that ensures that the devices in the fabric match their intended OS software version, and
  - Configuration Template Management: that ensures that the configuration on devices that are previously deployed matches the updated configuration template.

Metro Edge Fabric automation components

The automation components of Metro Edge Fabric include

- NSO fabric service: for fabric definition
- CX fabric-enabled ZTP: for automating fabric device onboarding
- CX fabric compliance functions: for OS version and template compliance, and
- CWM workflows: to better automate Fabric management that includes operations such as performing device upgrades, template compliance check, applying new templates, and so on.

Use Case: Device Fabric onboarding

Fabric-enabled ZTP

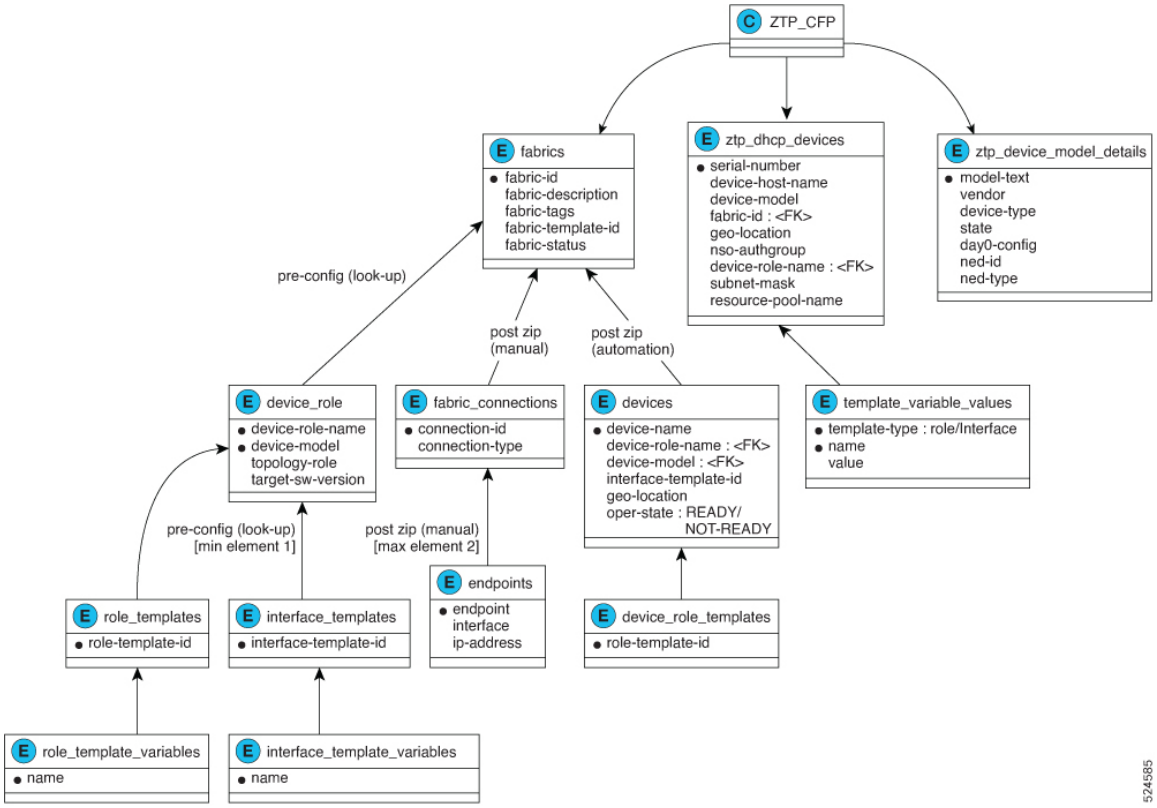
You can manually onboard devices into the fabric or use zero touch provisioning (ZTP).

Zero touch provisioning (ZTP) is the method used for automated device onboarding, with support for various ZTP attributes and scripts to complete device and fabric onboarding.

Components of Fabric-enabled ZTP

These are the high-level components of fabric-enabled ZTP.

Figure 22: High-level components of Fabric-enabled ZTP



Fabric

The Fabric Definition is the container or database used to store all attributes of the fabric including the devices, their roles, and the connectivity between fabric devices.

A fabric is defined along with device roles to be assigned to devices when onboarded.

Sample configuration:

```
config xmlns="http://tail-f.com/ns/config/1.0">
  <fabrics xmlns="http://fabric.manager/edge-fabric-manager-instance">
    <fabric-id>edge123</fabric-id>
    <device-role>
      <device-role-name>xr-leaf</device-role-name>
      <device-model>N540-12Z20G-SYS-A</device-model>
      <topology-role>leaf</topology-role>
      <role-templates>
        <role-template-id>day1-leaf-l2vpn</role-template-id>
      </role-templates>
      <interface-template>
        <interface-template-id>leaf-interface-connection</interface-template-id>
      </interface-template>
      <target-os-version></target-os-version>
    </device-role>
    <device-role>
      <device-role-name>xr-spine</device-role-name>
      <device-model>NCS-5501-SE</device-model>
      <topology-role>spine</topology-role>
```

```
      <role-templates>
        <role-template-id>day1-leaf-l2vpn</role-template-id>
      </role-templates>
      <role-templates>
        <role-template-id>day1-leaf-l3vpn</role-template-id>
      </role-templates>
      <interface-template>
        <interface-template-id>leaf-interface-connection</interface-template-id>
        <interface-template-variables>
          <name>MTU</name>
        </interface-template-variables>
      </interface-template>
      <target-os-version></target-os-version>
    </device-role>
  </fabrics>
</config>
```

Device role

The role definition is at the global <*fabrics*> level and you can use it across different fabrics.

Along with attributes such as device role name, device model, the device role in turn contains these templates:

- Role templates: templates defined under a role that are applied when the device is added to the fabric with a specific role.

  You can apply multiple configuration templates for a single role.

- Interface templates: templates that are applied when a fabric connection is made between two devices.

  Variables inside this template refer to template variables. The variables are assigned values when a device or a connection is added.

Sample configuration:

```
<device-role>
  <device-role-name>xr-leaf-xvpn</device-role-name>
  <device-model>NCS-5501-SE</device-model>
  <topology-role>spine</topology-role>
  <role-templates>
    <role-template-id>day1-leaf-l2vpn</role-template-id>
  </role-templates>
  <role-templates>
    <role-template-id>day1-leaf-l3vpn</role-template-id>
  </role-templates>
  <interface-template>
    <interface-template-id>spine-interface-connection</interface-template-id>
    <interface-template-variables>
      <name>MTU</name>
    </interface-template-variables>
  </interface-template>
  <target-os-version></target-os-version>
</device-role>
```

Manually onboard devices

Before you begin

- Onboard the device into NSO before onboarding into a fabric.

- Create the fabric and define the roles.

Procedure

Step 1 Define the device fabric attributes (*fabric-id*, *device-role-id*) in the ZTP attributes when manually onboarding the device

Example:

```
//Template

<config xmlns="http://tail-f.com/ns/config/1.0">
  <fabrics xmlns="http://fabric.manager/edge-fabric-manager-instance">
    <fabric-id>edge123</fabric-id>
    <devices>
      <device-name>fabric-spine2</device-name>
      <device-role-name>xr-spine</device-role-name>
      <device-model>NCS-5501-SE</device-model>
      <device-role-templates>
        <role-template-id>day1-leaf-l2vpn</role-template-id>
      </device-role-templates>
      <interface-template-id>leaf-interface-connection</interface-template-id>
      <geo-location>west</geo-location>
      <oper-state>READY</oper-state>
    </devices>
  </fabrics>
</config>
```

Config templates are applied when the device is onboarded.

Step 2 Manually onboard devices.

- CLI: using the **edge-fabric-actions onboard-device-into-fabric-apply-day1** command

```
edge-fabric-actions onboard-device-into-fabric-apply-day1 commit-type commit devices { device-name
 fabric-leaf2 serial-number FOC123456 }
```

- API: API call to NSO

```
http://<NSO_IP>:8080/restconf/operations/edge-fabric-manager:edge-fabric-actions/onboard-device-into-fabric-apply-day1
```

Payload:

```
{
    "onboard-device-into-fabric-apply-day1": {
        "commit-type":"commit",
        "devices": [
            {
                "device-name": "fabric-leaf2",
                "serial-number": "FOC2127R4F9"
            }
        ]
    }
}
```

Onboard devices using ZTP

Before you begin

- Required components to onboard devices using ZTP:
  - DHCP server (not supplied by Cisco) to bootstrap the device
  - Web server to download ZTP artifacts
  - CX NSO ZTP package
- Define fabric and roles before onboarding the devices.

Procedure

Step 1 Install and configure DHCP server.

Step 2 Install CX NSO ZTP package and define device ZTP attributes.

Step 3 Install Fabric Manager package.

Guidelines and sample configurations for onboarding devices using ZTP

Sample configuration for ISC DHCPD

If you execute iPXE boot, then the system downloads and boots the image. If you perform non-iPXE ZTP, then the system downloads and executes Python script to perform NSO onboarding.

Although this example maps specific host to IP address, it is not mandatory to do so.

```
# dhcpd.conf

allow bootp;
allow booting;

log-facility local7;

ddns-update-style none;
ignore client-updates;
default-lease-time 21600;
max-lease-time 43200;

class "vendor-classes" {
   match option vendor-class-identifier;
}

option space VendorInfo;
option VendorInfo.clientId code 1 = string;
option VendorInfo.authCode code 2 = unsigned integer 8;
option VendorInfo.md5sum code 3 = string;
option vendor-specific code 43 = encapsulate VendorInfo;
deny client-updates;
```

```
### OOB ZTP on MGMT Network

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.5 192.168.1.50;
    default-lease-time 28000;
    max-lease-time 72000;
    next-server 192.168.1.1;
    option routers 192.168.1.1;
    option domain-name "edgefabric.local";
    option domain-name-servers 192.168.1.1;
    set client-id = substring(option dhcp-client-identifier, 1, 20);
    log (info, concat("CLIENT IDENTIFIER", client-id));
    log (info, concat("VENDOR IDENTIFIER: ", option vendor-class-identifier));

    host fabric-spine1-oob {
    option dhcp-client-identifier "FOC2033R311";
    hardware ethernet 00:a2:ee:3f:6a:c8;
    fixed-address 192.168.1.200;
    if exists user-class and option user-class = "iPXE" {
       filename = "http://192.168.1.1:8080/images/ncs5500-mini-x-7.11.1.iso";
    } elsif exists user-class and option user-class = "xr-config" {
       filename = "http://192.168.1.1:8080/scripts/ncs_ztp_script_lab.py";
    }
  }
}
```

Sample configuration for CX NSO ZTP device definition

Device configuration:

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <dhcp-devices xmlns="http://com/cisco/as/servicepack/ztp">
      <serial-number>FOC2033R311</serial-number>
      <device-host-name>fabric-spine1</device-host-name>
      <fabric-id>edge123</fabric-id>
      <geo-location>west</geo-location>
      <nso-authgroup>fabric</nso-authgroup>
      <nso-device-group>ALL-ACCESS</nso-device-group>
      <device-role-name>xr-spine</device-role-name>
      <mgmt-ip>192.168.1.192</mgmt-ip>
      <subnet-mask>26</subnet-mask>
      <resource-pool-name>fabric-lab</resource-pool-name>
    </dhcp-devices>
  </devices>
</config>
```

Resource pool configuration:

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <resource-pools xmlns="http://tail-f.com/pkg/resource-allocator">
    <ip-address-pool xmlns="http://tail-f.com/pkg/ipaddress-allocator">
      <name>fabric-lab</name>
      <allocation>
        <id>fabric-spine2</id>
        <username>nso</username>
        <allocating-service
xmlns:ncs="http://tail-f.com/ns/ncs">/ncs:devices/ncs:device[ncs:name='fabric-spine2']</allocating-service>

        <redeploy-type>default</redeploy-type>
```

```
        <request>
          <subnet-size>26</subnet-size>
        </request>
      </allocation>
      <subnet>
        <address>192.168.1.0</address>
        <cidrmask>24</cidrmask>
      </subnet>
    </ip-address-pool>
  </resource-pools>
</config>
```

Guidelines for CX NSO ZTP device definition

Follow these guidelines for defining CX NSO ZTP devices:

- Configure DHCP ZTP devices under the NSO devices->dhcp-devices tree.
- Provide serial number because it is the primary key.
- Define the ZTP definition since the fabric ID and device role are defined under ZTP.
- Use standard NSO IP resource pool package to allocate management addresses.

  This example allocated a /26 out of 192.168.1.0/24.

Create fabric connections

Fabric connections express the intent of connections between two routers. Each connection has a maximum of two endpoints but can have a single endpoint to pre-configure nodes ahead of time.

When you make a fabric connection, the system applies the interface configuration template, if defined in the role definition. You can apply the template through NSO CLI or API.

Procedure

Step 1 Create a fabric connection.

- CLI: using the NSO **populate-fabric-connections** command

```
edge-fabric-actions populate-fabric-connections commit-type commit response-type show-dry-run
fabric-connection-details { fabric-id edge123 fabric-connections { connection-id d1-d2
connection-type d1-d2-type endpoints { endpoint fabric-leaf2 interface HundredGigE1/1 ip-address
 209.165.200.225/27 } } }
```

- API: using the **populate-fabric-connections**

  Sample postman payload:

```
URL:
http://<NSO_IP>:8080/restconf/operations/edge-fabric-manager:edge-fabric-actions/populate-fabric-connections
Method: POST
Header: Content-Type : application/yang-data+json
JSON Input payload:

{
    "populate-fabric-connections": {
```

```
            "commit-type": "commit",
            "response-type": "show-dry-run",
            "fabric-connection-details": [
                {
                    "fabric-id": "edge123",
                    "fabric-connections": [
                        {
                            "connection-id": "d1-d2",
                            "connection-type": "d1-d2-type",
                            "endpoints": {
                                "endpoint": "fabric-leaf2",
                                "interface": "HundredGigE1/1",
                                "ip-address": "192.168.1.2/24"
                            }
                        }
                    ]
                }
            ]
        }
    }
```

This action API invokes methods used to add Fabric connection details by using the templates whose name is configured under the **/edge-fabric-manager-instance:fabrics{<FABRIC_ID>}/device-role/**.

Input payload has different options for these parameters:

- **commit-type**: can be **commit** or **dry-run**
- **response-type**: can be **save-dry-run** or **show-dry-run**

  The **save-dry-run** option saves the dry-run output to a file under the path configured in **/fabric-static-config:fabric-static-config{dry-run-location}**

Step 2 Verify fabric connections.

Example:

```
Router#show fabrics fabric-connections

FABRIC                                  CONNECTION
ID           CONNECTION ID              TYPE          ENDPOINT       INTERFACE       IP ADDRESS

-------------------------------------------------------------------------------------------------------------
edge123  fabric-leaf1-connection     d1-d2-type      fabric-leaf1   TenGigE0/0/0/0   192.168.1.22/24
                                                     fabric-spine1  TenGigE0/0/0/0   192.168.1.24/24
         fabric-spine1-connection    d1-d2-type      fabric-leaf1   TenGigE0/0/0/0   192.168.1.34/24
                                                     fabric-spine1  TenGigE0/0/0/0   192.168.1.26/24
         fabric-spine2-connection    d1-d2-type      fabric-leaf1   TenGigE0/0/0/0   192.168.1.31/24
                                                     fabric-spine2  TenGigE0/0/0/0   192.168.1.24/24
```

Use Case: Fabric software lifecycle management

Fabric software lifecycle management

Fabric software lifecycle management (FSLM) is a process that ensures that the devices in the fabric match their intended OS software version.

The NSO Edge Fabric Manager performs the compliance check, and compliance enforcement or remediation.

Supported FSLM base functionalities

FSLM supports these base functionalities:

- Upgrade or downgrade single device based on {fabric, device}
- Upgrade or downgrade a set of devices based on {fabric, role}
- Upgrade or downgrade all devices based on {fabric}
- Upgrade cycles including the use of GISO for evolved XR (eXR) and XR7 operating systems
- Pre- and post-operations on the device and adjacent fabric devices that can be used for pre- and post-upgrade checks
- The **dry-run** capabilities to determine devices that are not running their target software version
- Upgrades as a set of atomic functions—for example, software download (staging), software upgrade, and software commit

Unsupported FSLM functionalities

FSLM does not support these functionalities:

- Automated graceful maintenance: the upgrade use case does not cover moving traffic off the node being upgraded. You can modify your own pre-check and post-check scripts as needed to perform these tasks.
- Service awareness
- User-guided interaction

FSLM Components

FSLM involves these components:

- CX NSO OS Upgrade application
- Crosswork Workflow Manager (CWM)

How to upgrade or downgrade a set of devices

The workflow for upgrading or downgrading a set of devices is based on these user inputs: {fabric}, {fabric, role}, or {fabric, device}.

Fabric Manager definition has a metadata field at the role and device level for *target-sw-version*. The device-level field is preferred in case the field is defined at both levels. When you want to upgrade the OS software on a device or set of devices you must augment the *target-sw-version* value using NSO CLI, Web UI, or API.

Process summary

The defined CWM workflows utilize the fabric database of the Fabric Manager application, or API to retrieve data regarding devices belonging to a specific fabric or role. The upgrade process is intent based, where one workflow determines the intended software version from the device or role metadata.

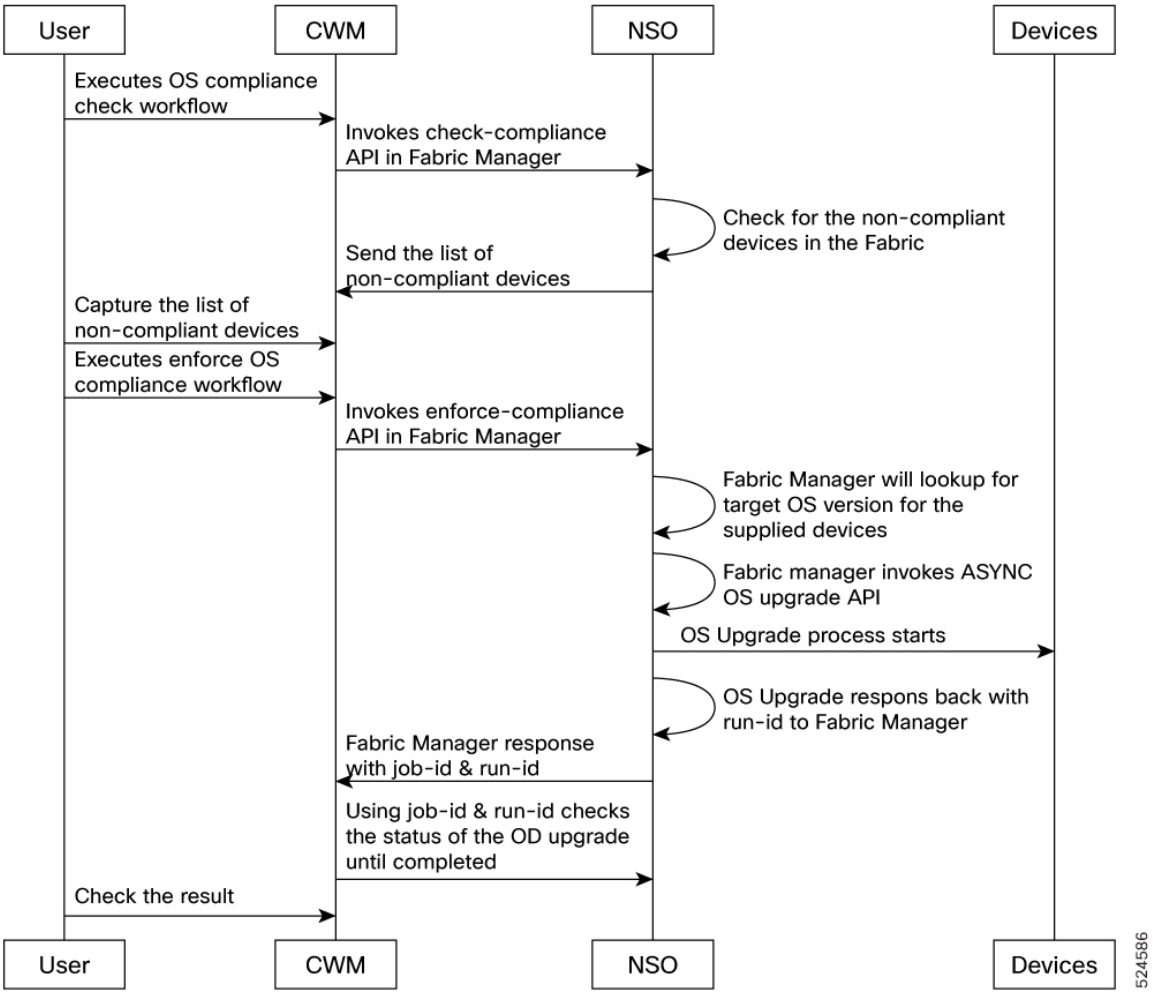
The FSLM workflow consists of

- compliance workflow: a workflow that checks the current software version in comparison to the intended software version and determines the set of devices that are currently non-compliant, and
- remediation workflow: a workflow that interacts with the OS upgrade NSO application to update the set of devices based on either {fabric-id}, {fabric-id, role}, or {fabric-id, device} input.

Automated and user-driven remediation: The compliance and remediation flows are sub-flows. You can use them independently or combine them into a single parent flow. The parent flow has an option to upgrade a set of devices without prompt—the devices will be upgraded to the target version without any further user intervention.

Workflow

Figure 23: FSLM workflow



These stages describe how FSLM works:

1. The user executes OS compliance check workflow with these input fields: *{{Fabric ID}}, {{Role ID}}, {{Device List}}*.

   The *{{Fabric ID}}* is the only mandatory attribute.

2. The CWM workflow retrieves the list of devices in a Fabric and their attributes using the Fabric Manager API.

   The input fields are evaluated for granularity and are used as a filter to retrieve an end set of devices. In Metro Release 1.0, these are free text strings and must match a defined Fabric and role in the Fabric Manager.

3. Once the set of devices is retrieved, CWM evaluates the list to determine which set of devices match the user input.

4. The next workflow evaluates which devices need to be upgraded.

5. CWM uses information from the NSO device data or live status to retrieve the current software version from each device.

   Devices which do not match the target version will be upgraded or downgraded to match the target version.

6. The result of this flow feeds into another flow interacting with the CX NSO OS Upgrade package to perform the relevant upgrades.

   The upgrade process may be iterative across a single device or done through batches in parallel.

Check OS compliance

This example shows how to check the version for all nodes in the Fabric. It checks the role definition for each device against the runtime version.

Procedure

Check OS compliance using the **check-compliance** command.

Example:

```
Router# edge-fabric-actions check-compliance compliance-type os-version-compliance fabric-details
{fabric-id edge123}
fabrics {
    fabric-id edge123
    devices {
        device-name fabric-leaf1
        os-compliant-status false
        response  Device fabric-leaf1 OS version is not compliant. Existing version is: 7.9.2 and
target version is: 24.4.1.
    }
    devices {
        device-name fabric-spine1
        os-compliant-status true
        response  Device fabric-spine1 OS version is compliant.
    }
    devices {
        device-name fabric-spine2
        os-compliant-status true
        response  Device fabric-spine2 OS version is compliant.
    }
}
```

Pre-upgrade and post-upgrade MOP

In the context of upgrades, the upgrade flow commonly includes a pre- and post-upgrade method-of-procedure (MOP) which perform multiple functions:

- Pre-upgrade MOP: ensures that the device and upgrade components are ready for the upgrade procedure. You can also use it to capture network state information for comparison after upgrade to ensure that the network returns to the correct state post-upgrade.

- Post-upgrade MOP: captures network and device state data or perform additional tasks post-upgrade. For example, in IOS XR routers, you can ensure that the device state is correct before executing **install commit** command post-upgrade.

NSO upgrade application

Metro Release 1.0 uses the CX *OS-Upgrade* NSO application to perform device upgrades. The application remains unchanged to support Edge Fabrics. The OS upgrade package must be pre-populated with data required for the upgrade. This includes the platform to software image mapping and image repository details.

Platform dependent software

The target software version is based on the NOS name and the NOS-specific numbering schema. For example, XR7 OS with software version 24.1.1. There may be platform-specific versions of the same NOS, such as eXR and XR7. In the current CX *OS-Upgrade* package, the platform type is identified as part of the upgrade process.

Error handling

Each component is responsible for its own error handling for its functions. The CWM workflow is ultimately responsible for monitoring and managing the lifecycle of the upgrades and reporting events generated by sub-components. As an example, if an upgrade is attempted on a device that is not currently reachable it would generate an event in CX *OS-Upgrade* which is being monitored by CWM. CWM would then report this underlying error in its own event reporting.

Use Case: Configuration template management

Configuration compliance

As part of the ongoing fabric infrastructure management, updates may need to be made to templates that were previously deployed. Configuration compliance mechanism helps to update the configuration on devices that were previously deployed to match the updated configuration template. The Crosswork Workflow Manager (CWM) drives this automation to update a set of devices.

Templates in Release 1.0 are defined at the fabric, role, and interface level. See the Metro Edge Fabric Manager, on page 35 section for details on configuration template definition.

Configuration template versioning

Configuration template versioning is done in two ways:

- Template versioning managed by the user: When a new template is built for a role, use a different name for the template and update the template being used in the fabric manager role definition.
- Templates stored with a revision number: Templates in NSO used for fabric management will be stored with a revision number. When a device is onboarded into the fabric with a specific role, the latest version of the template referenced in the role definition is used for configuration. At the device level, we also record the latest role and interface templates assigned to the device.

Device template compliance check and remediation

The compliance action starts with a CWM workflow that is used to first determine which devices are using a non-compliant template and then remediation of the device by deploying the correct template.

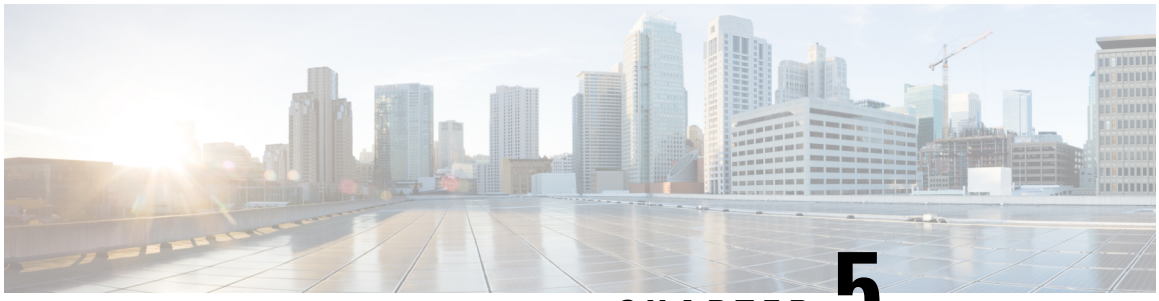
In the Fabric Manager, the *compliance* action is used with the *config* type to check each device to make sure that it is utilizing the correct template. The enforce-compliance API call is then used with the *config* type along with the list of devices to deploy the correct template.

Configuration enforcement dry-run

The configuration compliance includes a *dry-run* capability to see the changes prior to device provisioning.

Use this existing command in the CX Fabric Manager:

```
edge-fabric-actions enforce-compliance enforce-compliance-type config-compliance commit-type
 dry-run response-type show-dry-run fabric-details { fabric-id edge123 devices { device
fabric-leaf2 } }
```



CHAPTER 5

Sample Metro Deployment

This chapter covers the key components and sample end-to-end configuration involved in these Metro solution use cases:

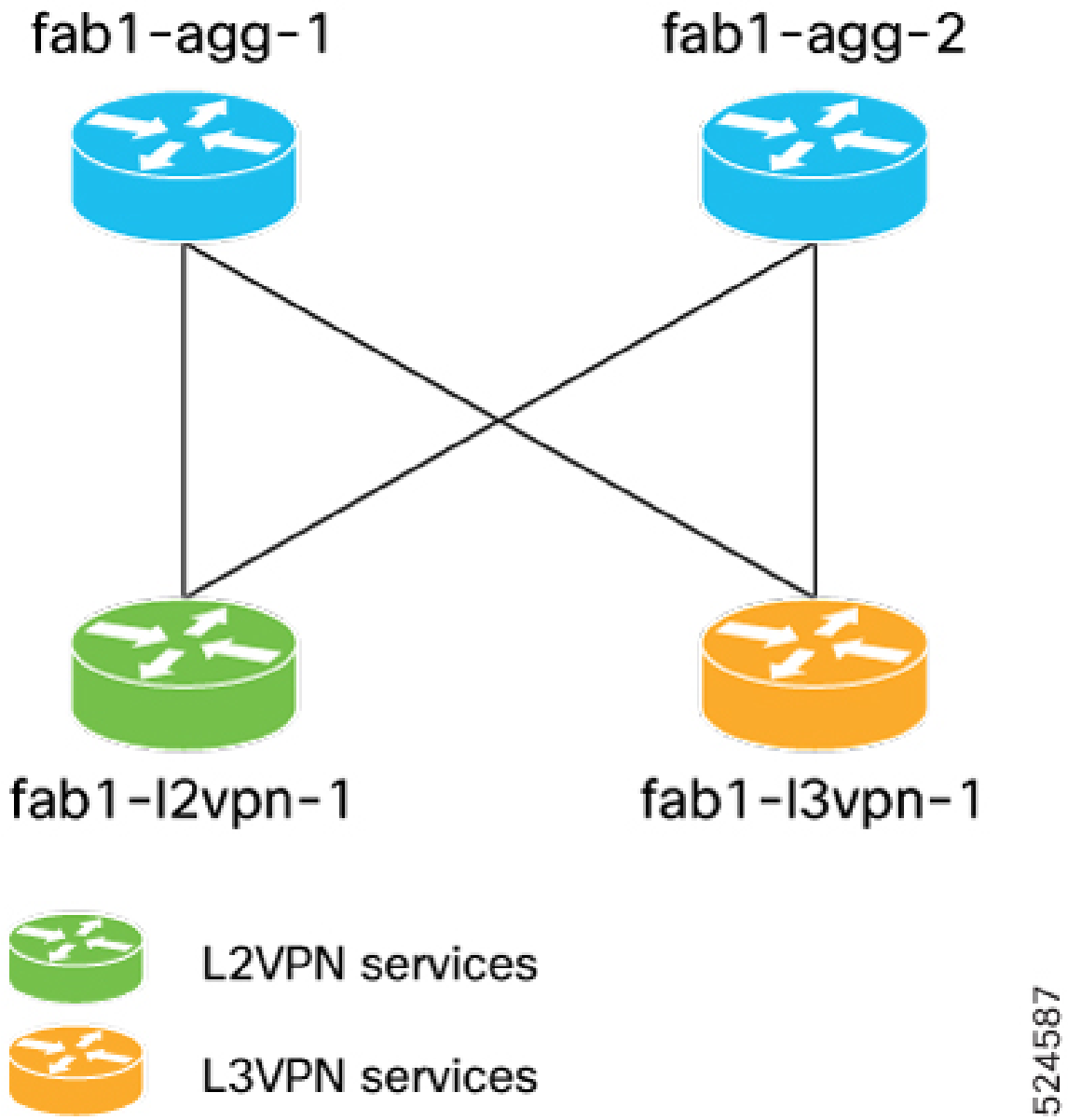
- Metro CX Edge Fabric Manager installation
- Metro CX Edge Fabric software life cycle management (FSLM)
- Metro CX Edge Fabric configuration template management
- Edge Protect DDoS deployment
- Cisco Routed PON deployment

Metro CX Edge Fabirc Manager installation

Sample Metro Edge Fabric deployment network

This is a sample topology diagram of Metro Edge Fabric deployment network.

Figure 24: Sample Metro Edge Fabric deployment network topology



The topology consists of two aggregation routers and two leaf nodes of different role types.

The aggregation routers in the fabric act as inline route reflectors. However, the route reflectors could be located elsewhere.

Control plane configuration

The table lists the protocols and parameters for control plane configuration.

Protocol	Parameters
BGP	ASN 100 IPv4 using separate loopback or route reflector (RR) IPv6 using separate loopback or RR Aggregation routers act as inline RR Aggregation routers have L3VPN and L2VPN address families enabled
IS-IS	Level 2 only SR-MPLS enabled SRv6 enabled TI-LFA enabled
Segment Routing	Flex-Algo 0 Flex-Algo 128 (low latency)
IPv4 IP Address Allocations	10.0.0.0/24 Flex-Algo 0 loopbacks 10.0.128.0/24 Flex-Algo 128 loopbacks
SR-MPLS SRGB Allocations	16000-32000 SRGB 16000-16999 Algo 0 17000-17999 Algo 128
SRv6 IP Address Allocations	2001:DC::/24 Base SRv6 global block 2001:DC00::/32 Flex-Algo 0 2001:DC80::/32 Flex-Algo 128

Additional configuration

The table lists the configuration required for additional elements.

Element	Parameter
Route Reflectors	10.0.0.100 10.0.0.101
SR-PCE	10.0.0.200 10.0.0.201
SNMP Trap Server	10.0.0.250

Address resource allocation

The table lists the address resource allocation for each node.

Node	IPv4 loopback	SR-MPLS SIDs	SRv6 locator	IPv6 loopback
fab1-agg-1	10.0.0.1/32 algo 0 10.0.1.1/32 algo 128	16001 algo 0 17001 algo 128	2001:DC00:0001::/48 algo 0 2001:DC80:0001::/48 algo 128	2001:DC00:0001::1/128
fab1-agg-2	10.0.0.2/32 algo 0 10.0.1.2/32 algo 128	16002 algo 0 17002 algo 128	2001:DC00:0002::/48 algo 0 2001:DC80:0002::/48 algo 128	2001:DC00:0002::1/128
fab1-l2vpn-1	10.0.0.3/32 algo 0 10.0.1.3/32 algo 128	16003 algo 0 17003 algo 128	2001:DC00:0003::/48 algo 0 2001:DC80:0003::/48 algo 128	2001:DC00:0003::1/128
fab1-l3vpn-1	105.0.0.4/32 algo 0 105.0.1.4/32 algo 128 10.0.0.4/32 algo 0 10.0.1.4/32 algo 128	16004 algo 0 17004 algo 128	2001:DC00:0004::/48 algo 0 2001:DC80:0004::/48 algo 128	2001:DC00:0004::1/128

CX Fabric Manager NSO solution components

This table lists the various NSO packages and its purpose.

NSO package	Version	Purpose
resource-manager	4.2.3	Define IP address pools for allocation through ZTP
ztp	3.7	CX ZTP package used for onboarding
os-upgrade	1.2	CX OS upgrade package used for software life-cycle management
edge-fabric-manager	4.3	Main Edge Fabric Management application used for fabric or role definition, config template application, and compliance actions
cisco-iosxr-cli-7.52	7.52.2	NSO CLI NED

Install CX Fabric Manager package

Follow these steps to install CX Fabric Manager package.

Procedure

Step 1 Download standard resource-manager package from CCO.

Step 2 Download CX Fabric Manager, CX ZTP, and CX OS upgrade packages

Step 3 Copy all packages to the packages directory relevant to your specific system or local installation of NSO. Example: /var/opt/nso/packages/

Step 4 Expand the CX NSO packages within the package directory so that the FM package can access the template directory.

Step 5 Verify packages.

Example:

```
Router# show packages package oper-stat


                          PACKAGE
NAME                      VERSION  UP
-----------------------------------------------
cisco-iosxr-cli-7.52      7.52.2   X
cisco-iosxr-nc-7.11       7.11     X
edge-fabric-manager       4.3      X
os-upgrade                1.2      X
resource-manager          4.2.3    X
ztp                       3.7      X
```

Manage CX Fabric Manager template

Templates are defined as NSO XML config files and are located in the *templates* directory for the expanded package post installation.

Fabric, role, and interface templates used by the Fabric Manager are standard NSO configuration templates stored with the FM Package.

Role templates are referenced in the role definition as the filename without the .xml file extension.

Note Adding or removing templates from the *templates* directory requires a package reload.

View available templates

Use the **ll** command to display the available templates in the templates directory.

```
root@nso-edge-fabric-test-1:/var/opt/ncs/packages/edge-fabric-manager/templates# ll
total 36
drwxrwxr-x 2 cisco cisco 4096 Jun 12 10:12 ./
drwxrwxr-x 6 cisco cisco 4096 Jun 21 13:56 ../
-rw-rw-r-- 1 cisco cisco 2289 May 7 14:24 day1-leaf-l2vpn-v1.xml
-rw-rw-r-- 1 cisco cisco 2113 May 7 14:24 day1-leaf-l2vpn.xml
-rw-rw-r-- 1 cisco cisco 3214 May 9 09:03 day1-leaf-l3vpn.xml
```

```
-rw-rw-r-- 1 cisco cisco 1585 May 7 14:24 delete-leaf-interface-connection.xml
-rw-rw-r-- 1 cisco cisco 532 May 7 14:24 fabric-generic.xml
-rw-rw-r-- 1 cisco cisco 3665 May 7 14:24 leaf-interface-connection-v1.xml
-rw-rw-r-- 1 cisco cisco 3369 May 7 14:24 leaf-interface-connection.xml

root@nso-edge-fabric-test-1:/var/opt/ncs/packages/edge-fabric-manager/templates#
```

Base templates

The base configuration templates apply to all routers in the fabric.

Primary loopback IP and SNMP templates

These are examples of templates defining the primary Loopback IP and SNMP.

Primary loop:

```
<devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>{$DEVICE}</name>
    <config>
      <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
        <Loopback>
          <id>0</id>
          <description>Loopback for L2VPN Leaf</description>
          <ipv4>
            <address>
              <ip>{$LOOPBACK_IP}</ip>
              <mask>255.255.255.255</mask>
            </address>
          </ipv4>
        </Loopback>
       </interface>
    </config>
  </device>
</devices>
```

SNMP:

```
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{$DEVICE}</name>
      <config>
        <snmp-server xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <ifmib>
            <ifalias>
              <long/>
            </ifalias>
            <stats>
              <cache/>
            </stats>
          </ifmib>
          <packetsize>4096</packetsize>
          <ifindex>persist</ifindex>
          <host>
            <address>10.0.0.250</address>
            <type>traps</type>
            <community-string>${SNMPTRAP_COMM}</community-string>
            <version>2c</version>
            <udp-port>1062</udp-port>
          </host>
          <community>
```

```
            <name>${SNMP_RW_COMM}</name>
            <RW/>
          </community>
          <community>
            <name>${SNMP_RO_COMM}</name>
            <RO/>
          </community>
          <location>"Fabric-A Location"</location>
          <trap-source>
            <MgmtEth>0/RP0/CPU0/0</MgmtEth>
          </trap-source>
        </snmp-server>
      </config>
    </device>
  </devices>
```

PTP configuration

```
<devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{$DEVICE}</name>
      <config>
        <ptp xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <clock>
            <domain>24</domain>
            <profile>
              <name>g.8275.1</name>
              <clock-type>T-BC</clock-type>
            </profile>
          </clock>
          <profile>
            <name>timeTransmitter</name>
            <multicast>
              <target-address>
                <ethernet>01-80-C2-00-00-0E</ethernet>
              </target-address>
            </multicast>
            <transport>ethernet</transport>
            <sync>
              <frequency>16</frequency>
            </sync>
            <clock>
              <operation>two-step</operation>
            </clock>
            <announce>
              <frequency>8</frequency>
            </announce>
            <delay-request>
              <frequency>16</frequency>
            </delay-request>
          </profile>


          <profile>
            <name>timeReceiver</name>
            <multicast>
              <target-address>
                <ethernet>01-80-C2-00-00-0E</ethernet>
              </target-address>
            </multicast>
            <transport>ethernet</transport>
            <announce>
              <timeout>5</timeout>
```

```
            <frequency>8</frequency>
          </announce>
          <delay-request>
            <frequency>16</frequency>
          </delay-request>
        </profile>
        <physical-layer-frequency/>
        <log>
          <servo>
            <events/>
          </servo>
          <best-master-clock>
            <changes/>
          </best-master-clock>
        </log>
      </ptp>
    </config>
  </device>
 </devices>
```

IS-IS configuration

```
<devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{$DEVICE}</name>
      <config>
        <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <isis>
            <tag>
              <name>CORE</name>
            <is-type>level-2-only</is-type>
              <net>
                <id>{$ISIS_NET}</id>
              </net>
              <flex-algo>
                <id>128</id>
                <metric-type>
                  <delay/>
                </metric-type>
                <advertise-definition/>
              </flex-algo>
              <distribute>
                <link-state>
                  <instance-id>{$BGP_LS_INSTANCE_ID}</instance-id>
                </link-state>
              </distribute>
              <log>
                <adjacency>
                  <changes/>
                </adjacency>
                <pdu>
                  <drops/>
                </pdu>
              </log>
              <lsp-refresh-interval>65000</lsp-refresh-interval>
            <max-lsp-lifetime>65535</max-lsp-lifetime>

               <address-family>
                <ipv4>
                  <unicast>
                    <metric-style>wide</metric-style>
                   <maximum-paths>32</maximum-paths>
                    <segment-routing>
```

```
                    <mpls/>
                  </segment-routing>
                </unicast>
              </ipv4>
              <ipv6>
                <unicast>
                  <metric-style>wide</metric-style>
                  <maximum-paths>32</maximum-paths>
                </unicast>
              </ipv6>
            </address-family>
            <interface>
              <name>Loopback0</name>
              <interface-type>passive</interface-type>
              <address-family>
                <ipv4>
                  <unicast>
                    <prefix-sid>
                      <index>{$SR_SID_INDEX_ALGO_0}</index>
                    </prefix-sid>
                    <prefix-sid-algorithm>
                      <prefix-sid>
                        <algorithm>
                          <id>128</id>
                          <absolute>{$SR_SID_ABS_ALGO_128}</absolute>
                       </algorithm>
                       </prefix-sid>
                       </prefix-sid-algorithm>
                  </unicast>
                </ipv4>
                <ipv6>
                  <unicast/>
                </ipv6>
              </address-family>
            </interface>
         </tag>
        </isis>
        </router>
      </config>
    </device>
 </devices>
```

Leaf templates

L2VPN leaf template: BGP

```
<devices xmlns="http://tail-f.com/ns/ncs">
    <name>{$DEVICE}</name>
    <config>
      <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
        <bgp>
          <bgp-no-instance>
            <id>{$BGP_ASN}</id>
            <nsr/>
            <bgp>
              <router-id>{$BGP_ROUTER_ID}</router-id>
              <graceful-restart/>
            </bgp>
            <ibgp>
              <policy>
                <out>
                  <enforce-modifications/>
```

```
            </out>
          </policy>
        </ibgp>

        <address-family>
          <l2vpn>
            <evpn/>
          </l2vpn>
        </address-family>
        <neighbor-group>
          <name>SvRR-EVPN</name>
          <remote-as>{$BGP_ASN}</remote-as>
          <update-source>
            <Loopback>0</Loopback>
          </update-source>
          <address-family>
            <l2vpn>
              <evpn/>
            </l2vpn>
          </address-family>
        </neighbor-group>
        <neighbor>
          <id>10.0.0.100</id>
          <use>
            <neighbor-group>SvRR-EVPN</neighbor-group>
          </use>
        </neighbor>
        <neighbor>
          <id>10.0.0.101</id>
          <use>
            <neighbor-group>SvRR-EVPN</neighbor-group>
          </use>
        </neighbor>
      </bgp-no-instance>
    </bgp>
  </router>
 </config>
</device>
</devices>
```

L3VPN leaf template: BGP

```
<devices xmlns="http://tail-f.com/ns/ncs">
    <name>{$DEVICE}</name>
      <config>
       <router xmlns="http://tail-f.com/ned/cisco-ios-xr">
        <bgp>
          <bgp-no-instance>
            <id>{$BGP_ASN}</id>
            <nsr/>
            <bgp>
              <router-id>{$BGP_ROUTER_ID}</router-id>
              <graceful-restart/>
            </bgp>
            <ibgp>
              <policy>
                <out>
                  <enforce-modifications/>
                </out>
              </policy>
            </ibgp>

            <address-family>
```

```
            <vpnv4>
              <unicast/>
            </vpnv4>
            <vpnv6>
              <unicast/>
            </vpnv6>
          </address-family>
          <neighbor-group>
            <name>SvRR-L3VPN</name>
            <remote-as>{$BGP_ASN}</remote-as>
            <update-source>
              <Loopback>0</Loopback>
            </update-source>
            <address-family>
              <vpnv4>
                <unicast/>
              </vpnv4>
            </address-family>
          </neighbor-group>
          <neighbor>
            <id>10.0.0.102</id>
            <use>
              <neighbor-group>SvRR-L3VPN</neighbor-group>
            </use>
          </neighbor>
          <neighbor>
            <id>10.0.0.103</id>
            <use>
              <neighbor-group>SvRR-L3VPN</neighbor-group>
            </use>
          </neighbor>
        </bgp-no-instance>
      </bgp>
    </router>
  </config>
</device>
</devices>
```

View loaded templates

Use the **show packages package edge-fabric-manager templates** command on the router CLI to view the loaded templates.

```
Router-cisco@ncs#show packages package edge-fabric-manager templates
templates [ day1-leaf-l2vpn day1-leaf-l2vpn-v1 day1-leaf-l3vpn
delete-leaf-interface-connection fabric-base-isis fabric-base-loopback fabric-base-ptp
fabric-base-snmp fabric-generic leaf-interface-connection leaf-interface-connection-v1
spine-interface-connection ]
```

Example of Fabric and role definition

This example defines a fabric with the ID, *fab1*, with a base template, *fab1-base-template*, already defined.

The *xr-leaf-l2vpn* role and its properties are shown. Multiple templates and their associated variables are defined as part of the role definition. The role also defines an interface-template with its own variables which can be used to perform operations such as adding the interface to a specific ISIS instance or assigning a PTP profile.

```
fabrics fabric-id fab1
 fabric-description "Example Fabric"
 fabric-tags        "test;cisco"
```

```
 fabric-template-id fab1-base-template
 device-role device-role-name xr-leaf-l2vpn device-model N540-24Z8Q2C-M
  topology-role     leaf
  role-templates role-template-id leaf-l2vpn-bgp-1
   role-template-variables name BGP_ASN
   !
   role-template-variables name BGP_ROUTER_ID
   !
  !
  role-templates role-template-id leaf-l2vpn-isis-v1
   role-template-variables name BGP_LS_INSTANCE_ID
   !
   role-template-variables name ISIS_NET
   !
   role-template-variables name SR_SID_INDEX_ALGO_0
   !
   role-template-variables name SR_SID_ABS_ALGO_128
   !
  !
  interface-template interface-template-id fab1-interface-template
  interface-template interface-template-variables name MTU
  !
  interface-template interface-template-variables name PTP_PROFILE
  target-os-version 7.11.2
 !
!
```

Onboard Fabric

In this deployment we onboard the spine devices manually and onboard the leaf devices using ZTP. The leaf devices are automatically onboarded into the fabric with a specific role during the ZTP process.

Follow these steps to onboard Fabric.

Procedure

Step 1 Define spine fabric role.

Example:

```
fabrics fabric-id fab1
 device-role device-role-name xr-fabric-spine device-model NCS-5501-SE
  topology-role     spine
  role-templates role-template-id fabric-base-isis
   role-template-variables name BGP_LS_INSTANCE_ID
   !
   role-template-variables name ISIS_NET
   !
   role-template-variables name SR_SID_ABS_ALGO_128
   !
   role-template-variables name SR_SID_INDEX_ALGO_0
   !
  !
  role-templates role-template-id fabric-base-loopback
   role-template-variables name LOOPBACK_IP
   !
  !
  role-templates role-template-id fabric-base-ptp
  !
```

```
  role-templates role-template-id fabric-base-snmp
  !
  interface-template interface-template-id spine-interface-connection
  target-os-version 7.11.2
```

Step 2 Define DHCP device for the spine device.

Example:

```
devices dhcp-devices FOC2120R22S
 device-host-name   fab1-spine-2
 fabric-id          fab1
 geo-location       west
 nso-authgroup      fabric
 nso-device-group   ALL-ACCESS
 device-role-name   xr-fabric-spine
 mgmt-ip            192.168.1.192
 subnet-mask        24
 resource-pool-name fabric-lab
 device-model       NCS-5501-SE
 template-variable-values template-type role
  variables name BGP_LS_INSTANCE_ID
   value 100
  !
  variables name ISIS_NET
   value 49.0001.0105.0000.0002.00
  !
  variables name LOOPBACK_IP
   value 10.0.0.2
  !
  variables name SR_SID_INDEX_ALGO_0
   value 2
  !
  variables name SR_SID_ABS_ALGO_128
   value 17002
 !
!
```

These are some of the attributes in this example:

- Device type: NCS-5501-SE
- Device serial number: FOC2120R22S
- DCHP device: added with the key being the device serial number
- Four base templates applied to the spine device: PTP, IS-IS, SNMP, and the loopback interface.

  IS-IS and the loopback have device-specific variables defined. PTP and SNMP use hard-coded values in the template itself.

Step 3 Onboard the spine device.

a) Onboard the device into NSO.

Example:

```
devices device fab1-spine-2
 address   192.168.1.192
 authgroup fabric
```

```
 device-type cli ned-id cisco-iosxr-cli-7.52
 state admin-state unlocked
```

b) Onboard the device into spine role.

Example:

```
Router:cisco@ncs# edge-fabric-actions onboard-device-into-fabric-apply-day1 devices { device-name
 fab1-spine-2 device-model NCS-5501-SE fabric-id fab1 device-role-name xr-fabric-spine serial-number
 FOC2120R22S geo-location west }
devices {
    device-name fab1-spine-2
    fabric-id fab1
    status Completed
    response
                        Dry-run location is:
/home/nso/temp/dry-run-output/fabric-device-onboard-fab1_fab1-spine-2_2024-09-15T18:47:36.189545.txt
}
```

c) Verify spine device onboarding.

Example:

Interface loopback:

```
Router:fab1-spine-2#show run int lo0
Sun Sep 15 17:21:56.428 UTC
interface Loopback0
 description Loopback interface
 ipv4 address 105.0.0.2 10.0.0.2 255.255.255.255
!
```

SNMP:

```
Router:fab1-spine-2#show run snmp
Sun Sep 15 17:26:20.468 UTC
snmp-server host 105.0.0.250 traps version 2c public udp-port 1062
snmp-server community cisco RW
snmp-server community public RO
snmp-server community private RW
snmp-server location "Fabric-A Location"
snmp-server packetsize 4096
snmp-server trap-source MgmtEth0/RP0/CPU0/0
snmp-server ifmib ifalias long
snmp-server ifindex persist
snmp-server ifmib stats cache
```

IS-IS:

```
Router:fab1-spine-2#show run router isis
Sun Sep 15 17:25:57.621 UTC
router isis CORE
 is-type level-2-only
 net 49.0001.0105.0000.0002.00
 distribute link-state instance-id 100
 log adjacency changes
 log pdu drops
 lsp-refresh-interval 65000
 max-lsp-lifetime 65535
 address-family ipv4 unicast
  metric-style wide
  maximum-paths 32
```

```
 segment-routing mpls
!
address-family ipv6 unicast
 metric-style wide
 maximum-paths 32
!
flex-algo 128
 metric-type delay
 advertise-definition
!
interface Loopback0
 passive
 address-family ipv4 unicast
  prefix-sid index 2
  prefix-sid algorithm 128 absolute 17002
```

PTP:

```
Router:fab1-spine-2#show run ptp
Sun Sep 15 17:25:20.506 UTC
ptp
 clock
  domain 24
  profile g.8275.1 clock-type T-BC
 !
 profile timeReceiver
  multicast target-address ethernet 01-80-C2-00-00-0E
  transport ethernet
  announce timeout 5
  announce frequency 8
  delay-request frequency 16
 !
 profile timeTransmitter
  multicast target-address ethernet 01-80-C2-00-00-0E
  transport ethernet
  sync frequency 16
  clock operation two-step
  announce frequency 8
  delay-request frequency 16
 !
 physical-layer-frequency
 log
  servo events
  best-master-clock changes
```

Step 4 Define L2VPN leaf fabric role.

Example:

```
fabrics fabric-id fab1
 device-role device-role-name xr-fabric-leaf-l2vpn device-model N540-12Z20G-SYS
  topology-role     leaf
  role-templates role-template-id fabric-base-isis
   role-template-variables name BGP_LS_INSTANCE_ID
   !
   role-template-variables name ISIS_NET
   !
   role-template-variables name SR_SID_ABS_ALGO_128
   !
   role-template-variables name SR_SID_INDEX_ALGO_0
   !
  !
  role-templates role-template-id fabric-base-loopback
```

```
   role-template-variables name LOOPBACK_IP
   !
  !
  role-templates role-template-id fabric-base-ptp
  !
  role-templates role-template-id fabric-base-snmp
  !
  role-templates role-template-id fabric-leaf-l2vpn-bgp
   role-template-variables name BGP_ASN
   !
   role-template-variables name BGP_ROUTER_ID
   !
  !
  interface-template interface-template-id spine-interface-connection
  target-os-version 7.11.2
 !
!
```

Step 5 Define DHCP device for L2VPN leaf node.

Example:

```
devices dhcp-devices FOC2430PL4Z
 device-host-name   fab1-l2vpn-1
 fabric-id          fab1
 geo-location       west
 nso-authgroup      fabric
 nso-device-group   ALL-ACCESS
 device-role-name   xr-l2vpn
 mgmt-ip            192.168.1.64
 subnet-mask        24
 resource-pool-name fabric-lab
 device-model N540-12Z20G-SYS
 template-variable-values template-type role
  variables name BGP_ASN
   value 100
  !
  variables name BGP_ROUTER_ID
   value 10.0.0.3
  !
  variables name BGP_LS_INSTANCE_ID
   value 100
  !
  variables name ISIS_NET
   value 49.0001.0105.0000.0003.00
  !
  variables name LOOPBACK_IP
   value 10.0.0.3
  !
  variables name SR_SID_INDEX_ALGO_0
   value 3
  !
  variables name SR_SID_ABS_ALGO_128
   value 17003
 !
!
```

This step is same as the previous example of spine device with the addition of the BGP template.

The DCHP device is added with the key being the device serial number. The role template variables are populated with device-specific values

Step 6 Configure DHCP day0 configuration.

Example:

```
!! IOS XR
hostname ###HOST_NAME###
logging console disable
!
username lab
 group root-lr
 group cisco-support
 secret 5 $1$CcGF$EzBAkyycnbZFt4QRF16I20
!
grpc
 no-tls
 port 57344
 address-family ipv4
!
interface MgmtEth0/RP0/CPU0/0
 ipv4 address ###MGMT_IP### ###MGMT_MASK###
 description To MGMT network
 no shutdown
!
router static
 address-family ipv4 unicast
  0.0.0.0/0 192.168.1.1
 !
!
line console
 exec-timeout 0 0
!
line default
 exec-timeout 0 0
!
fpd auto-upgrade enable
netconf-yang agent
 ssh
!
lldp
!
ssh client source-interface MgmtEth0/RP0/CPU0/0
ssh server logging
ssh timeout 120
ssh server rate-limit 600
ssh server session-limit 100
ssh server v2
ssh server vrf default
ssh server netconf vrf default
```

The DHCP day0 configuration applies baseline configuration to the device so that further configuration can take place.

This file is located on the DCHP server and referenced as part of the DHCP role settings in *dhcpd.conf*. The variables are populated by the CX ZTP process.

Step 7 Initiate ZTP

Example:

```
Router:fab1-l2vpn-1#ztp initiate debug dhcp4 management verbose
```

How zero touch provisioning works

These stages describe the zero touch provisioning (ZTP) process.

1. The system initiates DHCP request on the management interface.
2. DHCP responds with interim management IP address, location of Day 0 configuration for the device, and the CX ZTP Python script.
3. The system applies Day 0 configuration to the device for management connectivity and user setup.
4. The user executes CX ZTP onboarding script on the router which in turn performs these operations:
   a. Assigns permanent management IP from NSO resource pool
   b. Onboards device into NSO with the specific hostname and allocated IP address
   c. Executes automated fabric onboarding to onboard the device into the specified fabric with the appropriate role templates applied

Verify Fabric onboarding

Procedure

Verify fabric onboarding.

Example:

```
Router:cisco@ncs# show fabrics devices
FABRIC                                                                                                  GEO
 OPER
ID       DEVICE NAME    DEVICE ROLE NAME      DEVICE MODEL       INTERFACE TEMPLATE ID    LOCATION
 STATE  ROLE TEMPLATE ID
-------------------------------------------------------------------------------------------------------------------
fab1     fab1-l2vpn-1   xr-fabric-leaf-l2vpn  N540-12Z20G-SYS-A  interface-template       west
 READY fabric-base-isis

       fabric-base-loopback

       fabric-base-ptp

       fabric-base-snmp

       fabric-leaf-l2vpn-bgp
         fab1-spine-2   xr-fabric-spine       NCS-5501-SE        interface-template       west
 READY fabric-base-isis

       fabric-base-loopback

       fabric-base-ptp

       fabric-base-snmp
```

Create Fabric connections

Once devices are onboarded into NSO and the fabric, you can then create device connections between them.

Connections contain endpoints. A single endpoint is useful for configuring a spine device performing in-band ZTP for downstream leaf devices.

Procedure

Step 1 Create connection between interfaces on both the routers.

Example:

```
Router-cisco@ncs#edge-fabric-actions populate-fabric-connections fabric-connection-details { fabric-id
 fab1 fabric-connections { connection-id spine1_leaf1 connection-type fabric endpoints { endpoint
fab1-spine-2 interface TenGigE0/0/0/32 ip-address 10.120.1.1/24 } endpoints { endpoint fab1-l2vpn-1
 interface TenGigE0/0/0/32 ip-address 10.120.1.50/24 } } } commit-type commit
```

In this example, we create a connection between TenGigE0/0/0/22 on both the *fab1-spine-2* and *fab1-l2vpn-1* routers.

The default commit-type is a **dry-run** to show the configuration without deploying to the device. Use the **commit** commit-type to commit the configuration to NSO and deploy to the devices.

Step 2 Verify fabric connections.

Example:

```
Router-cisco@ncs# show fabrics fabric-connections

FABRIC   CONNECTION ID       TYPE        ENDPOINT       INTERFACE        IP ADDRESS

-------------------------------------------------------------------------------------------

fab1     spine1_leaf1        fabric      fab1-l2vpn-1   TenGigE0/0/0/32  10.120.1.50/24

                                         fab1-spine-2   TenGigE0/0/0/32  10.120.1.1/24
```

Metro CX Edge Fabric software life cycle management

This section highlights applying the steps required to perform a software upgrade utilizing the Crosswork Workflow Manager workflows which determine whether the device complies with the intended target OS version. If the device is not in compliance, we remediate the condition by upgrading the device software.

For a brief on software life cycle management, see Fabric software lifecycle management, on page 44.

Perform prerequisite configurations for software upgrade

In Metro Release 1.0 software upgrades are performed using the CX OS Upgrade package. This is a comprehensive NSO-based service used to handle device upgrades for NX OS, IOS XE, and IOS XR OS.

Before you begin

You must define the device types and supported versions in the OS Upgrade package prior to performing upgrades.

In this example, we are upgrading a device of type NCS540L

Procedure

Step 1 Define the supported upgrade paths between software versions.

Example:

```
os-upgrade-service lookup-data upgradePathLookup image-version-mapping cisco-ios-xr
 entries NCS540L 7.11.1 24.4.1.37I
 !
 entries NCS540L 24.4.1.37I 7.11.1
 !
!

os-upgrade-service lookup-data platform-lookup platform-mapping cisco-ios-xr
 platform NCS540L
  model-keywords NCS540L
   device-model  NCS540L
   upgrade-reload-time 600
  !
  firmware-upgrade-enabled false
 !
!

os-upgrade-service lookup-data os-upgrade-vars image-vars image-version cisco-ios-xr NCS540L 24.4.1.37I

 vars SYSTEM_IMAGE
  var-value ncs540l-x64-24.4.1.37I.iso
 !
 vars target-version
  var-value 24.4.1.37I
 !
!

os-upgrade-service lookup-data version-image-lookup image-version-mapping cisco-ios-xr
 entries NCS540L 7.11.1
  image-filename [ ncs540l-x64-7.11.1.iso ]
 !
 entries NCS540L 24.4.1.37I
  image-filename [ ncs540l-x64-24.4.1.37I.iso ]
 !
!
```

In this example, we are upgrading from 7.11.1 to 24.4.1.37I.

Step 2 Define Fabric Manager role.

Example:

```
fabrics fabric-id metro10-fabric1
 device-role device-role-name xr-leaf device-model N540-24Q8L2DD-SYS
  topology-role     leaf
  target-os-version 24.4.1.37I
```

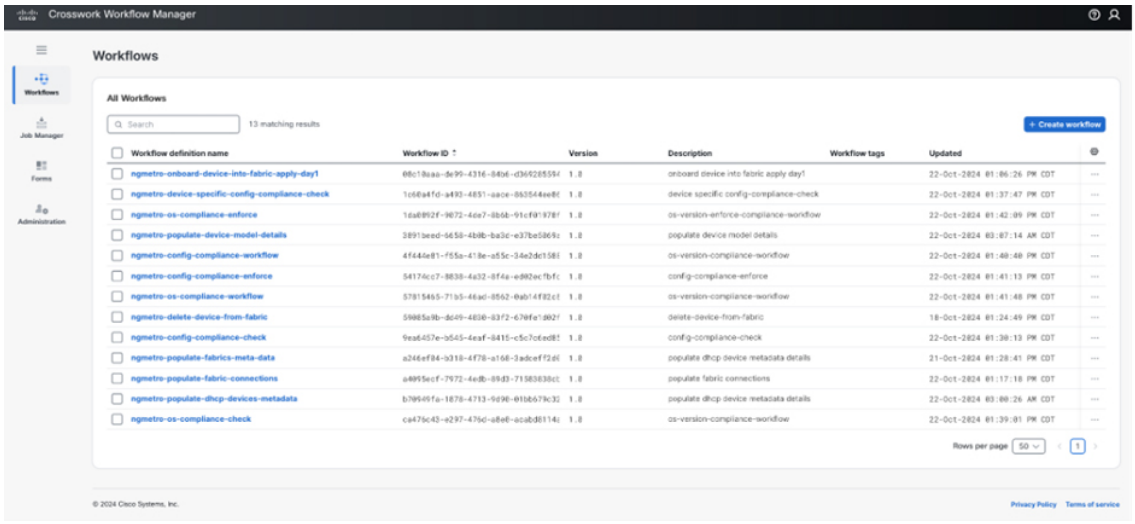
!
!

The intended version of the operating system is defined in the role definition. The device-model is used as a key and as an input to the CX OS upgrade service to determine the correct files and methodology for upgrading the device.

Crosswork Workflow Manager

Crosswork Workflow Manager is a flexible tool that is used to create customized network workflows. Workflows can be hierarchical in nature, with parent workflows with specific inputs used to drive child workflows to both collect data as well as execute actions against resources. In this use case, we are using CWM to execute flows to check the current software version of the device against an intended version defined by the fabric role.

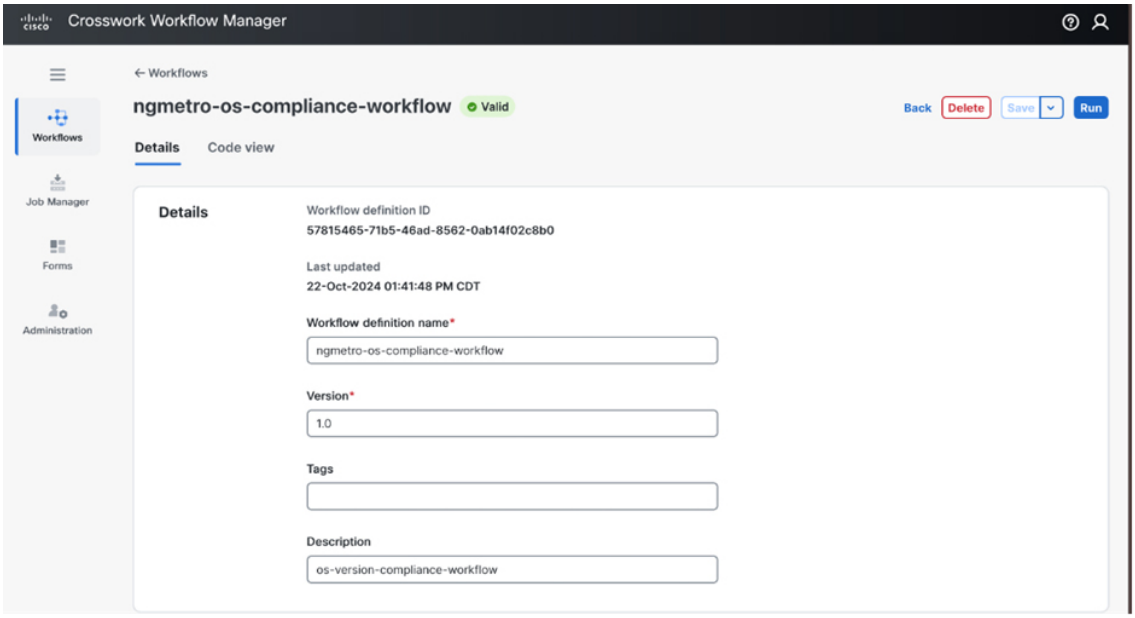
Figure 25: Crosswork Workflow Manager



Device OS compliance workflow

We use `ngmetro-os-compliance-workflow`, a compound workflow for performing device OS compliance check and remediation.

Figure 26: Device OS compliance workflow



These stages describe the device OS compliance workflow.

1. The compound workflow, `ngmetro-os-compliance-workflow`, first calls the `ngmetro-os-compliance-check` workflow to check for non-compliant devices.

   The `ngmetro-os-compliance-check` workflow uses these attributes as the input:

   ```
   ngmetro-os-compliance-check
       input:
       {
        "fabricId" : "metro10-fabric1",
        "nsoResource" : "METRO-NSO"
       }
   ```

2. The `ngmetro-os-compliance-check` workflow induces the OS compliance check function that is part of the Fabric Manager service.

3. When that task is executed, the `ngmetro-os-compliance-check` workflow returns the status of the devices in the fabric.

   In this case, the device `fabric-leaf3` is found as non-compliant.

   ```
   {
       "eventId": "21",
       "eventTime": "2024-10-22T18:39:16.152371499Z",
       "eventType": "WorkflowExecutionCompleted",
       "taskId": "2097494",
       "workflowExecutionCompletedEventAttributes": {
         "result": {
           "payloads": [
             {
               "metadata": {
                 "encoding": "json/plain"
               },
   ```

```
            "data": {
              "Data": {
                "checkComplianceResult": [
                  {
                    "device-name": "fabric-leaf3",
                    "os-compliant-status": false,
                    "response": " Device fabric-leaf3 OS version is not compliant.
Existing version is: 7.11.2.17I and target version is: 24.4.1."
                  }
                ],
                "fabricId": "metro10-fabric1",
                "nsoResource": "METRO-NSO"
              }
            }
          }
        ]
      },
      "workflowTaskCompletedEventId": "20"
    }
  }
]
```


4. The compound workflow runs the `ngmetro-os-compliance-enforce` workflow to perform the proper upgrades based on the target software version defined in the fabric roles.

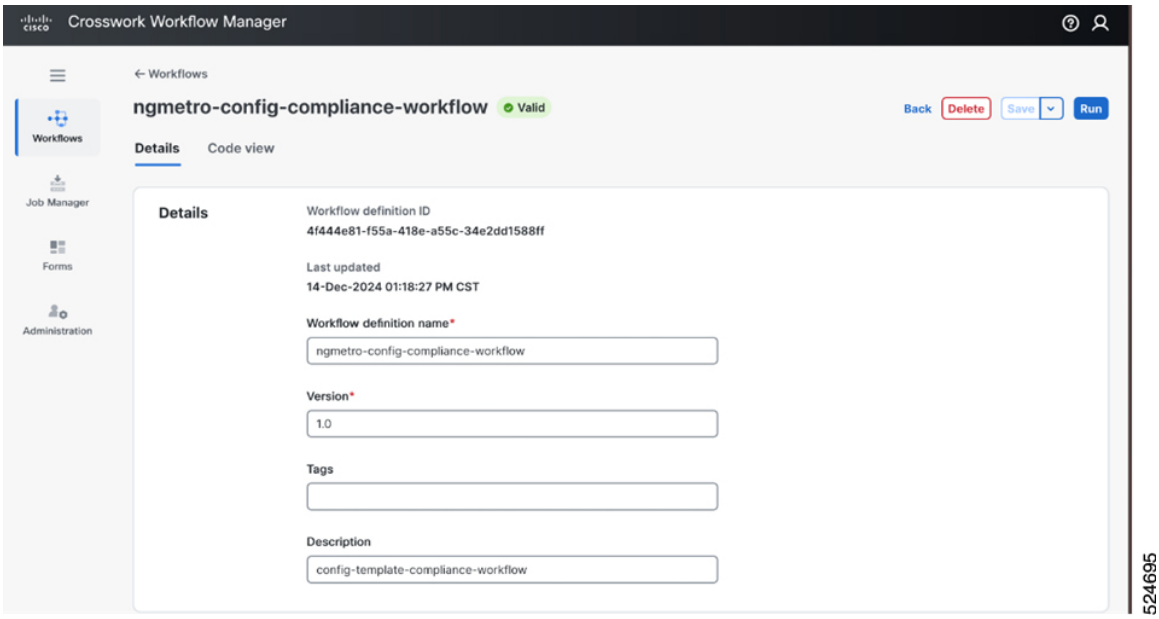
What's Next

The next step in the workflow is to remediate the device and upgrade it to the target version specified. For conciseness we are omitting the details of that step, but it utilizes the CX OS upgrade configuration mentioned earlier to upgrade the device matching type NCS540L with the appropriate Cisco IOS XR software version 24.4.1.

Metro CX Edge Fabric configuration template compliance

Perform configuration template compliance check and remediation

Figure 27: Crosswork Workflow Manager: configuration compliance workflow



Similar to the OS compliance workflow, we use a compound workflow in CWM to perform a configuration template compliance check and remediation:

- `ngmetro-config-compliance-workflow` workflow: to check the current assigned template in the Fabric Manager definition against what has been deployed on the node, and
- `ngmetro-config-compliance-enforce-workflow`: for remediation.

For a brief on configuration template management, see the Configuration compliance, on page 48 section.

Follow these steps to run the compound workflow for configuration template compliance check and remediation.

Procedure

Step 1 Run the `ngmetro-config-compliance-check` workflow with the appropriate fabric as input.

Example:

```
"data": {
              "status": 200,
              "data": {
                "edge-fabric-manager:output": {
                  "fabrics": [
                    {
```

```
                        "devices": [
                          {
                            "device-name": "fabric-leaf3",
                            "interface-compliant-status": false,
                            "response": "Device fabric-leaf3 Role-template is compliant. Device
fabric-leaf3 Interface-template is not compliant. Existing Interface-template is:
leaf-interface-connection and latestInterface-template is: interconnect-template.",
                            "role-compliant-status": true
                          }
                        ],
                        "fabric-id": "metro10-fabric1"
                      }
                    ]
                  }
                }
```


This step checks all the nodes in the Fabric against their intended templates vs. the runtime templates and returns whether the templates defined for the device and role match the runtime templates.

In this example the role template is compliant, but the interface template has been changed from `leaf-interface-connection` to `interconnect-template` and is no longer compliant.

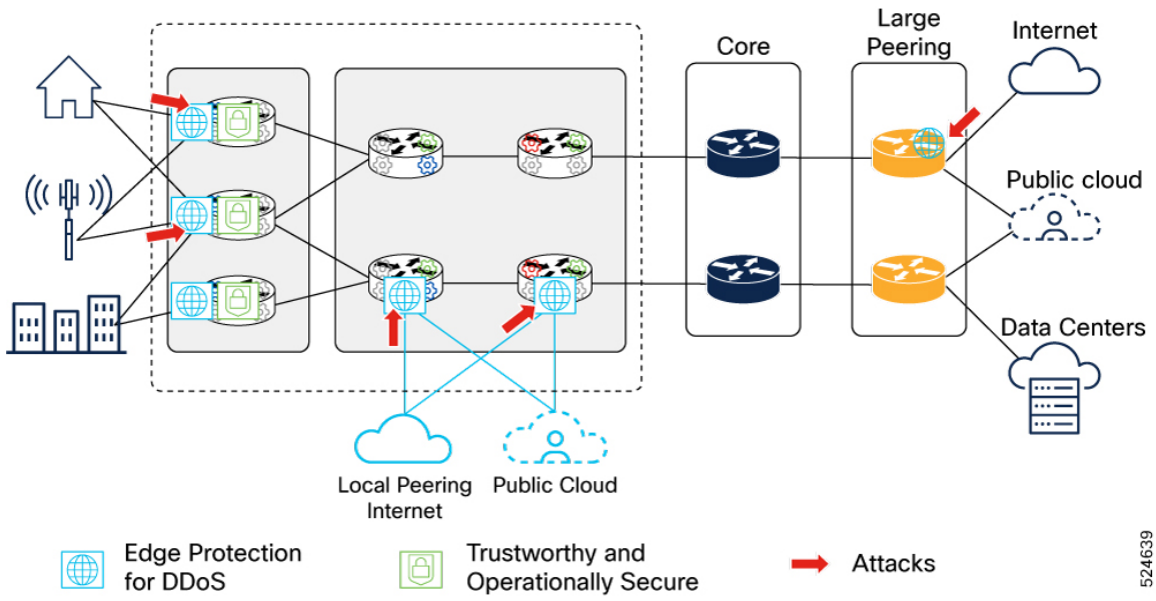
Step 2 Once you find devices with non-compliant templates, run the remediation workflow—`ngmetro-config-compliance-enforce-workflow`—with the appropriate fabric as input.

This step applies the proper configuration templates to devices which have been changed.

Edge Protect DDoS deployment

Once the Edge Fabric is deployed, you can deploy additional value-added services to the Fabric. Edge Protect DDoS is an innovative solution allowing service providers to deploy DDoS protection at the very edge of the network. This helps not only protect end services from DDoS and other security threats but also helps detect and mitigate attacks originating from within a service provider network.

Figure 28: High-level view of Edge Protect DDoS deployment topology



Edge Protect components

The Edge Protect solution consists of these components:

- Edge DDoS Protection Controller: This component runs as a VM on a virtualization infrastructure. The controller is responsible for configuring and deploying detectors, configuring protected objects, and monitoring detected attacks and active mitigations.
- Edge DDoS Protection Detector: This component runs as a third-party application on IOS-XR routers, providing a distributed detection and mitigation engine. There is a single detector deployed to each router.

Figure 29: Detector lists

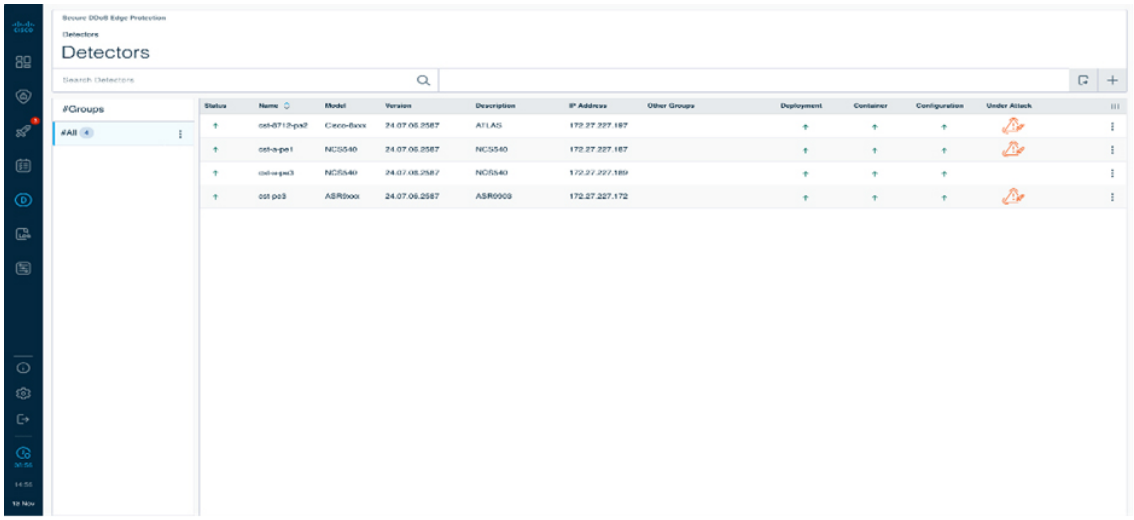
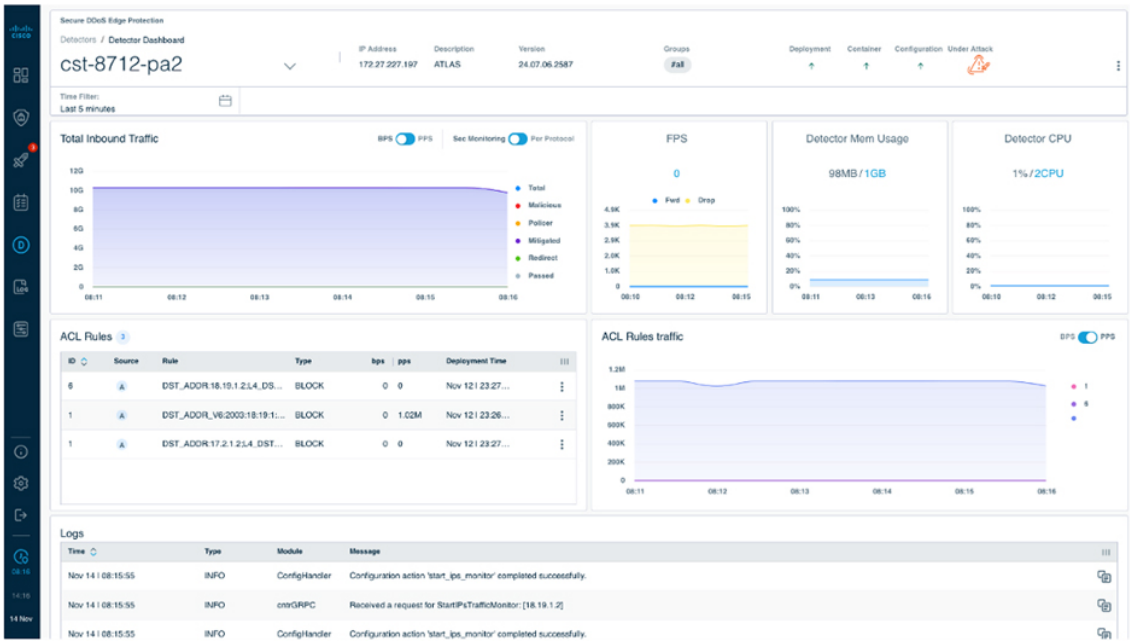


Figure 30: Detector details



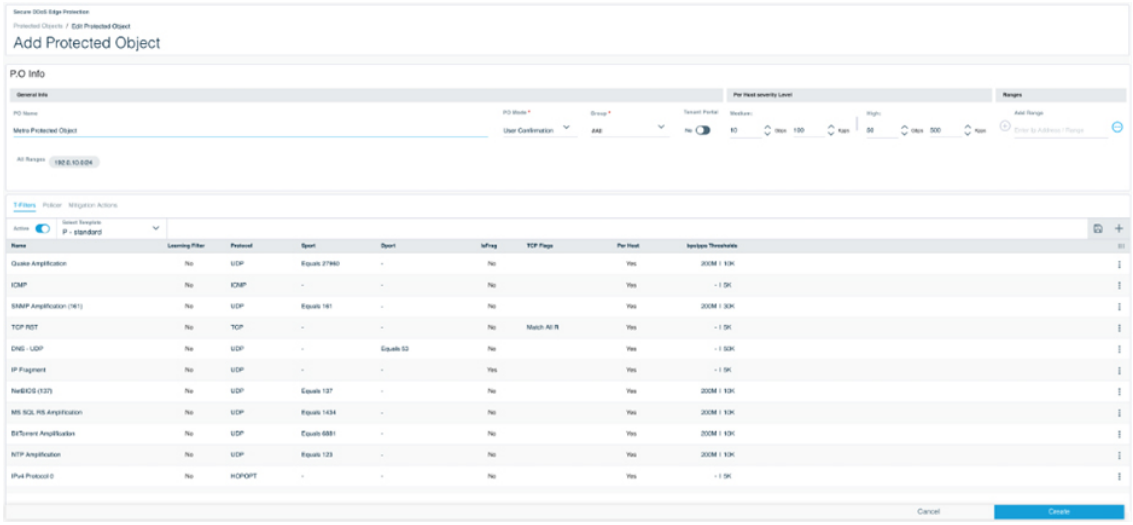
Edge Protect operation

The Edge Protect operation involves Edge Protect DDoS detection and mitigation. The detection phase uses advanced analytics and examinations to identify specific amplification and application layer attacks. The mitigation phase is performed by deploying either user-driven or automated ACLs to distributed nodes to limit the attack impact.

Edge Protect DDoS detection

Detecting attacks requires having flow-level visibility of traffic traversing the router. In the Edge Protect solution Netflow data is encoded into Google Protobuf format and then consumed by the on-board detection engine. Deployment of the appropriate Netflow configuration is done automatically by the controller when a router acting as a detector is onboarded. Standard or user-defined detection templates are used as a basis for detecting DDoS traffic patterns. These are applied to protected objects, which can apply to an entire device, or a subset of IP addresses based on configuration. Protected objects are then configured with policing and additional mitigation actions.

Figure 31: Edge detect projected objects

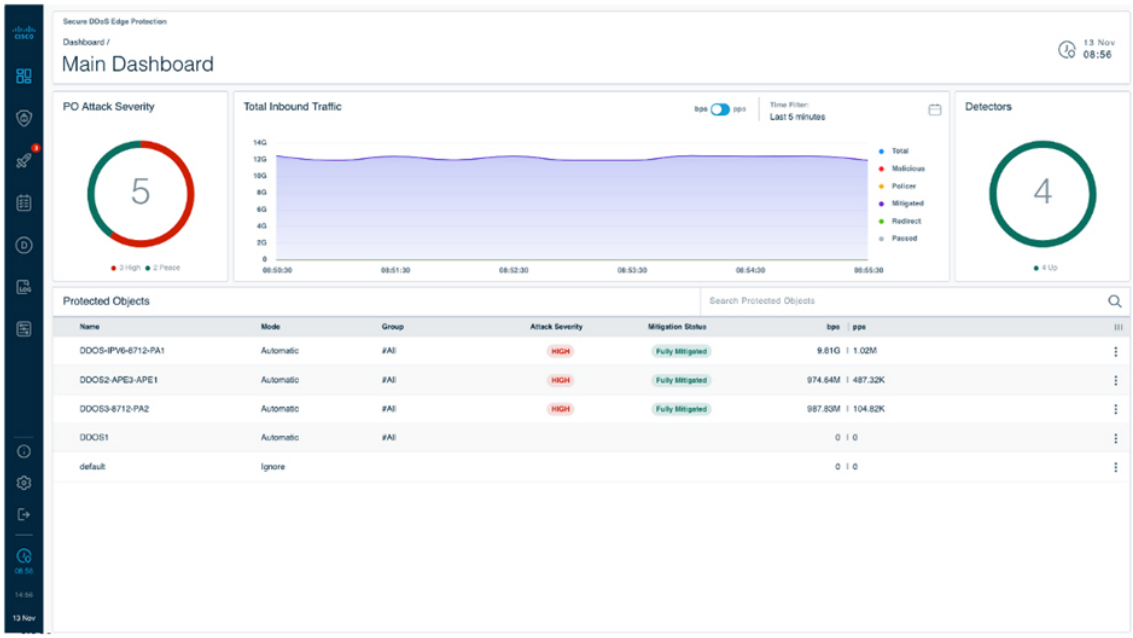


524698

Edge Protect DDoS mitigation

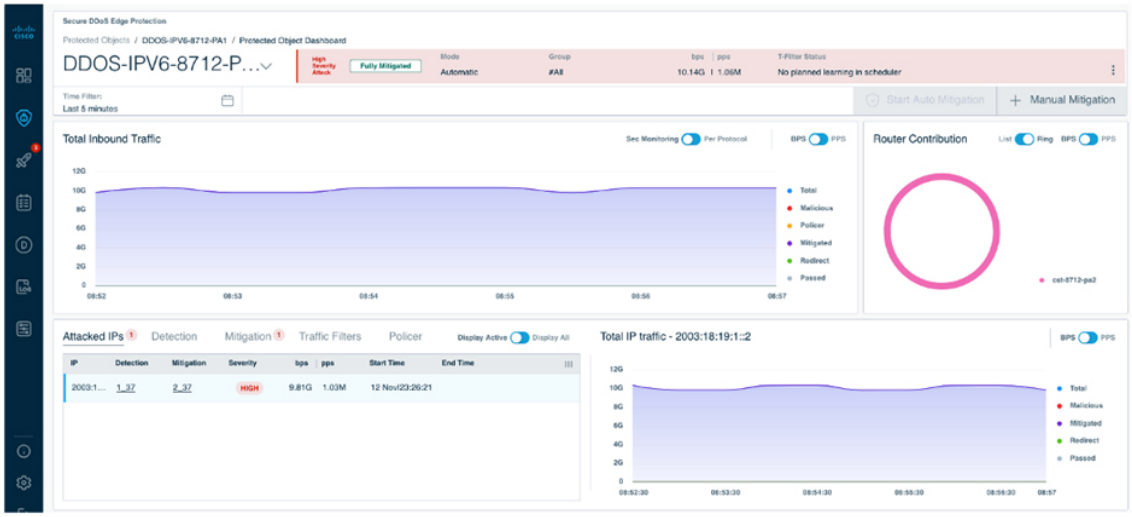
Mitigation on distributed nodes is performed using standard data plane ACLs. The ACLs contain granular information to mitigate only the attacks, and on platforms supporting User Defined Fields (UDF) can mitigate traffic by pattern matching components in the IP header. Active mitigations are shown both on the main launch screen for the Controller as well as under specific detectors.

Figure 32: Active mitigations



524699

Figure 33: Protected object mitigation details



For details on Edge DDoS Protection, see the Quick Start Guide for Edge DDoS Protection.

Example of Edge Protect deployed ACL

This is an example of an automated ACL used to mitigate an attack. In this case, the protected objects cover the 203.0.113.100 and 198.51.100.2 IP addresses. The ACL is as granular as possible to mitigate attacks in a targeted manner. ACL sequence number 1 blocks DNS packets with a specific length 228 and sequence 6 blocks an application layer attack against http.

```
RP/0/RP0/CPU0:cst-8712-pa2#show run ipv4 access-list myACL
ipv4 access-list myACL
 1 deny udp any eq 3000 host 203.0.113.100 eq domain packet-length eq 228
 6 deny tcp any eq 3400 host 198.51.100.2 eq www match-all -established -fin -psh +syn -urg
 packet-length eq 1178
 1301 permit ipv4 any any
```

Cisco Routed PON deployment

Components of Cisco Routed PON

The Cisco Routed Passive Optical Networking solution is based on Cisco uOLT PON SFP, and the Routed PON Management applications.

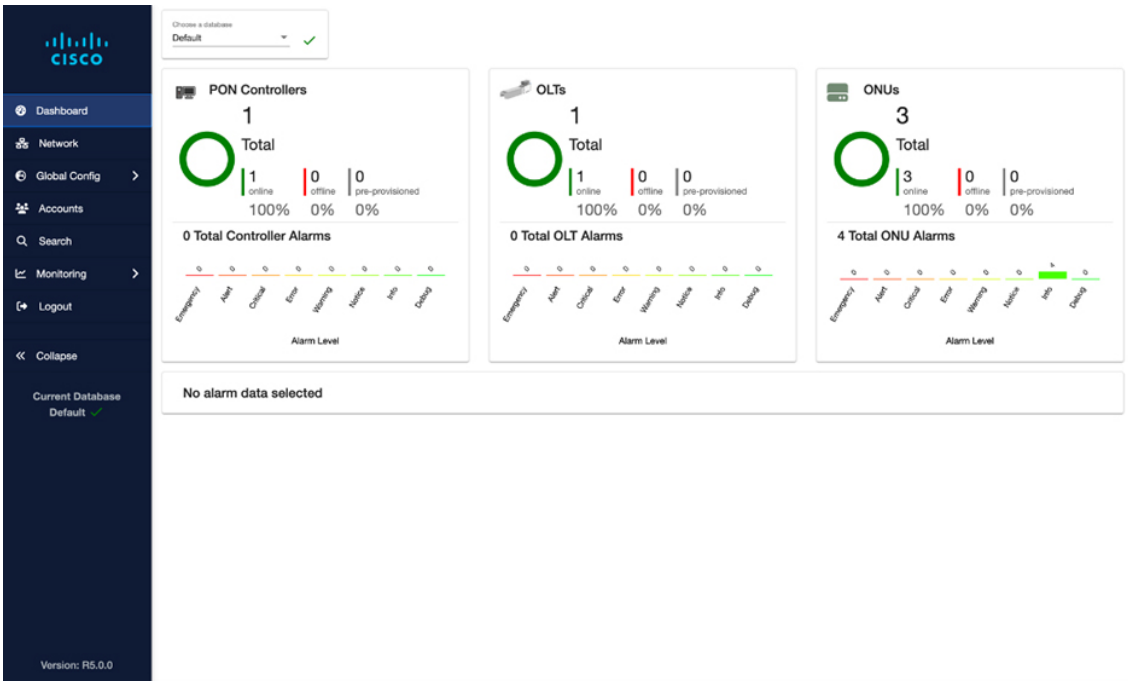
These are the major elements in manageability solution:

- Cisco Routed PON Manager
- Cisco Routed PON Controller

Cisco Routed PON Manager

Cisco Routed PON Manager is a single-page web application and an accompanying REST API that provides a graphical user interface for managing the Cisco Routed PON Network.

Figure 34: Cisco Routed PON Manager



524701

The key features of Cisco Routed PON manager include

- alarm management
- dashboard view with a summary of PON network conditions
- device monitoring and statistics
- device provisioning and management
- logging for diagnostics and troubleshooting
- PON Controller database management
- PON Manager user management
- Graphical ONU Management and Control Interface (OMCI) (and future 10G EPON OAM) service configuration tool, and
- service configuration, including VLANs, SLAs, 802.1X authentication, and DHCP Relay.

Cisco Routed PON Controller

Cisco Routed PON Controller is a stateless software that primarily act as intelligent relay to push or pull information from OLT micro plug or ONUs and transfer them to or from data store. The PON Controller is hosted as a third-party application container in router where the PON SPFs are hosted. The pseudo driver functions implemented in PON Controller encode and encapsulate requests in IEEE 1904.2 packets (L2) to

communicate with downstream devices. The Cisco PON Controller runs as a third-party application the Cisco router hosting Cisco PON pluggable OLT.

Operational data including device state, statistics, alarms, and logging, is collected and flows upstream through the management network and presented in Cisco Routed PON Manager.

Hardware support for Cisco Routed PON in Metro solution

The table lists the supported hardware for Cisco Routed PON in Metro solution.

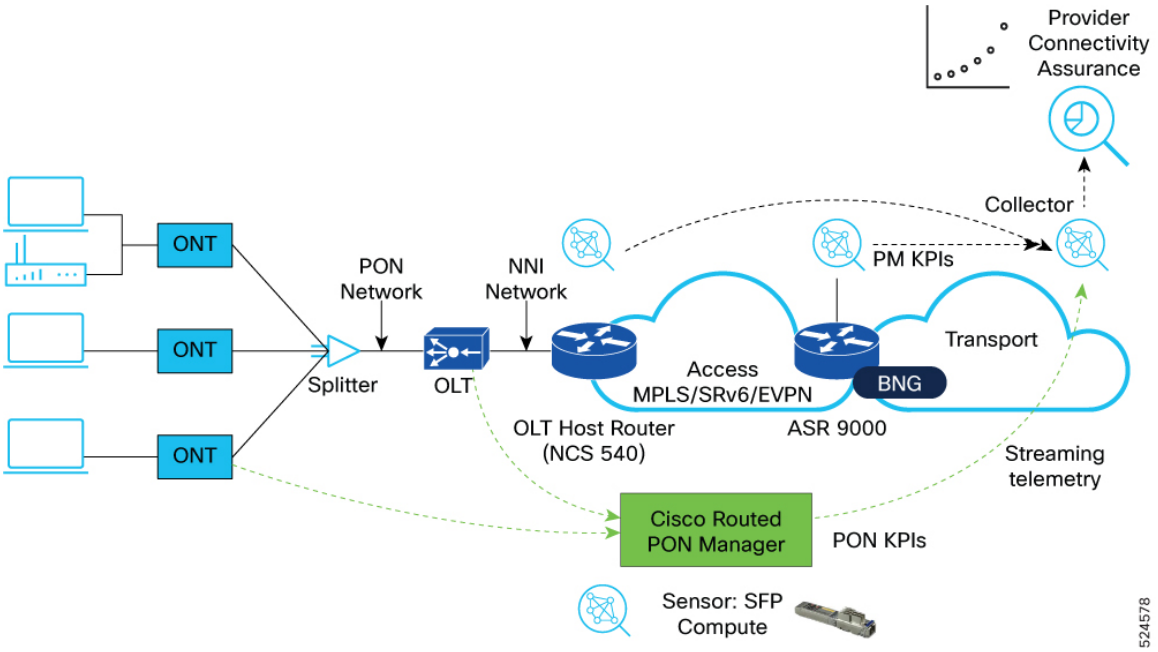
Table 5: Hardware support Matrix for Cisco Routed PON in Metro solution

Metro release	Product Id
Release 1.0	N540-24Z8Q2C-SYS
	N540-ACC-SYS
	N540-24Q8L2DD-SYS
	N540X-16Z4G8Q2C-D/A
	N540-28Z4C-SYS-D/A
	NCS-55A2-MOD-S
	NCS-57C1-48Q6D
	NCS-55A1-24Q6H-SS

Routed PON service assurance using Provider Connectivity Assurance

Provider Connectivity Assurance (PCA) provides assurance for Routed PON by combining data from the PON network, network fabric, and BNG subscriber data to create the end-to-end assurance view for PON attached endpoints.

Figure 35: Routed PON PCA integration

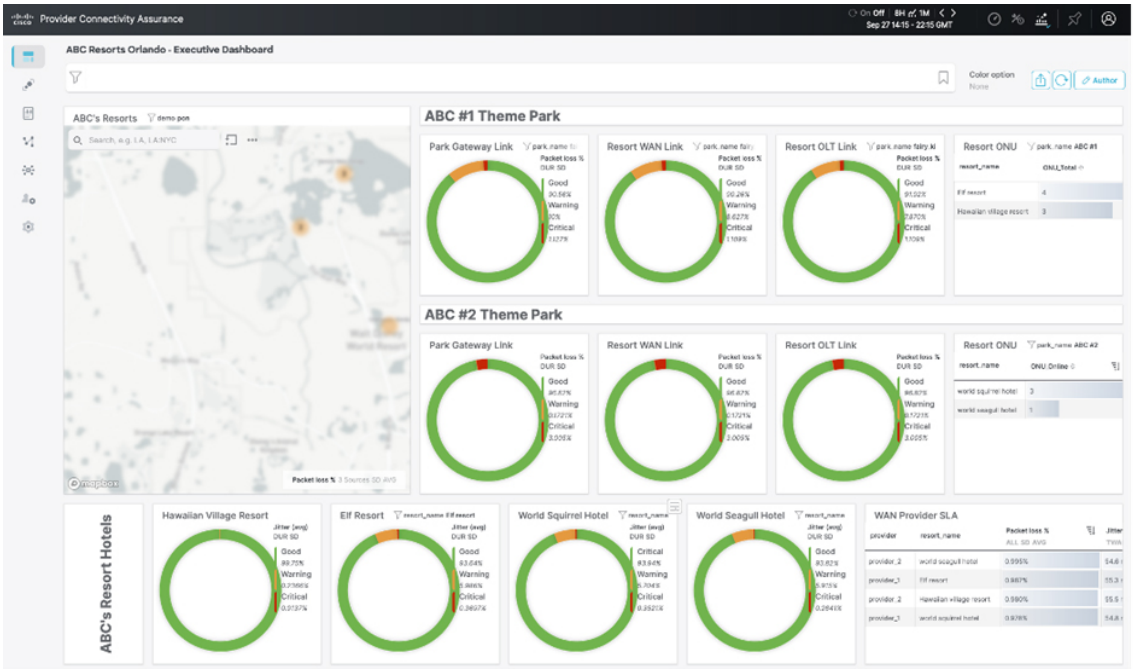


These are the PCA use cases specific to RPON service assurance:

- Integration with Cisco PON Manager to collect KPI data for the PON network
- Monitor network infrastructure data and correlate that with the PON network data. One use case is to monitor latency between PON endpoints and cnBNG user plane endpoints, correlating specific subscribers connected to a specific BNG user plane
- Integration with Cisco cnBNG user planes and control plane to provide additional per-subscriber assurance for the solution by combining BNG health and subscriber monitoring into the PCA solution

The figure depicts the Routed PON assurance dashboard.

Figure 36: Routed PON assurance dashboard



524702