Cisco Dynamic Fabric Automation Configuration Guide

First Published: 2015-03-11

Last Modified: 2015-07-31

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

Preface

The Preface contains the following sections:

Audience

This publication is for experienced network administrators who configure and maintain Cisco Dynamic Fabric Automation.

Document Organization

This document is organized into the following chapters:

Chapter	Description
"Information About Cisco DFA"	Provides an overview of Cisco Dynamic Fabric Automation (DFA) and descriptions of the Cisco DFA building blocks.
"Migration to Cisco DFA"	Provides information about how to prepare for migration to Cisco DFA, including migration steps and migration configuration.
"Troubleshooting Migration"	Provides the verification steps on how to troubleshoot Cisco DFA migration.

Document Conventions

Note As part of our constant endeavor to remodel our documents to meet our customers' requirements, we have modified the manner in which we document configuration tasks. As a result of this, you may find a deviation in the style used to describe these tasks, with the newly included sections of the document following the new format.

Command descriptions use the following conventions:

Convention	Description
bold	Bold text indicates the commands and keywords that you enter literally as shown.
Italic	Italic text indicates arguments for which the user supplies the values.
[x]	Square brackets enclose an optional element (keyword or argument).
[x | y]	Square brackets enclosing keywords or arguments separated by a vertical bar indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a vertical bar indicate a required choice.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.
`variable`	Indicates a variable for which you supply values, in context where italics cannot be used.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.

Examples use the following conventions:

Convention	Description
`screen font`	Terminal sessions and information the switch displays are in screen font.
`boldface screen font`	Information you must enter is in boldface screen font.
italic screen font	Arguments for which you supply values are in italic screen font.
< >	Nonprinting characters, such as passwords, are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

This document uses the following conventions:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Related Documentation for Cisco Dynamic Fabric Automation

The Cisco Dynamic Fabric Automation documentation is at the following URL: http://www.cisco.com/c/en/us/support/cloud-systems-management/dynamic-fabric-automation/tsd-products-support-series-home.html.

The Cisco Nexus 6000 Series documentation is at the following URL: http://www.cisco.com/c/en/us/support/switches/nexus-6000-series-switches/tsd-products-support-series-home.html.

The Cisco Nexus 7000 Series documentation is at the following URL: http://www.cisco.com/c/en/us/support/switches/nexus-7000-series-switches/tsd-products-support-series-home.html.

The Cisco Nexus 5500 and 5600 Series documentation is at the following URL: http://www.cisco.com/c/en/us/support/switches/nexus-5000-series-switches/tsd-products-support-series-home.html.

The Cisco Nexus 1000V switch for VMware vSphere documentation is at the following URL: http://www.cisco.com/en/US/products/ps9902/tsd_products_support_series_home.html. The documentation therein includes the following guides for Cisco DFA. Additional information pertaining to troubleshooting can be located in the Cisco Nexus 1000V documentation for Cisco NX-OS Release 4.2(1)SV2(2.2).

- *Cisco Nexus 1000V DFA Configuration Guide, Release 4.2(1)SV2(2.2)*
- *Cisco Nexus 1000V VDP Configuration Guide, Release 4.2(1)SV2(2.2)*

The Cisco Prime Data Center Network Manager (DCNM) documentation is at the following URL: http://www.cisco.com/en/US/products/ps9369/tsd_products_support_series_home.html. The Cisco Prime DCNM documentation for Cisco DFA includes but is not limited to the following guides:

- *Cisco DCNM 7.0 OVA Installation Guide.*
- *Cisco DCNM 7.0 Fundamentals Guide*
- *Cisco DCNM DFA REST 7.0 API Guide*

The Cisco Prime Network Services Controller (NSC) documentation is at the following URL: http://www.cisco.com/en/US/products/ps13213/tsd_products_support_series_home.html.

The OpenStack for Cisco DFA install documentation includes the following guide and documents:

- *Open Source Used In OpenStack for Cisco DFA 1.0* at the following URL: http://www.cisco.com/en/US/docs/switches/datacenter/dfa/openstack/opensource/OpenStack_for_Cisco_DFA_1.0_Open_Source_Documentation.pdf
- *OpenStack for Cisco DFA Install Guide Using Cisco OpenStack Installer* at the following URL: http://www.cisco.com/en/US/docs/switches/datacenter/dfa/openstack/install/guide/os-dfa-coi.pdf

- *OpenStack for Cisco DFA Install Guide for Using Pre-built OpenStack for Cisco DFA Images* at the following URL: http://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/dfa/openstack/install/guide/preblt-image.pdf
- *Quick Guide to Clonezilla* at the following URL: http://www.cisco.com/en/US/docs/switches/datacenter/dfa/openstack/install/guide/clonezilla-image-restore.pdf

Documentation Feedback

To provide technical feedback on this document, or to report an error or omission, please send your comments to: .

We appreciate your feedback.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*, at: http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.

CHAPTER 1

New and Changed Information

New and Changed Information

This chapter provides release-specific information for each new and changed feature in the *Cisco Dynamic Fabric Automation Configuration Guide*.

Cisco NX-OS Release Number	Platform Supported	New/Enhanced Features	Chapter/Topic Where Documented
7.2(0)D1(1)	Cisco Nexus 7000 Series Switches	Dynamic provisioning simplifies the management of the VRF and VLAN/BD configurations.	Dynamic Provisioning, on page 77
		Enhanced Forwarding functionality.	Unicast Forwarding, on page 97
		Multi-tenant data center handles the traffic segregation between different tenants.	Multi-tenancy, on page 47
		VN-Segment network can support up to 16 million virtual network segments.	VN-Segment, on page 51
		Single point of management support.	XMPP Client Configuration, on page 37
		DHCP relay configuration.	Segment ID Support for DHCP Relay, on page 59

Cisco NX-OS Release Number	Platform Supported	New/Enhanced Features	Chapter/Topic Where Documented
7.2(0)N1(1)	Cisco Nexus 5000 and 6000 Series Switches	VRF profile is updated on the leaf resulting in the loopback routable IP address being auto-configured under that VRF.	Configuring a Profile , on page 74
		Support common DHCP-Servers for IP address assignments within DFA.	Configuring Windows 2012 as DHCP Server, on page 62
		POAP failure can be detected with locator LED.	POAP Diagnostics, on page 26
		Enhanced syslogs are generated when profile apply, profile un-apply, and profile refresh is performed.	Auto-Configuration, on page 71
		Enable conversational learning on all leaf nodes.	Conversational learning on Border Leaf, on page 112
		IPv4 default route generation for default VRF towards fabric nodes.	Default Route Advertisement, on page 21

7.2(0)D1(1)
7.2(0)N1(1)
	Cisco Nexus 5000, 6000 and 7000 Series Switches	Border Leaf U-shape support.	Border Leaf Deployment Consideration, on page 24

CHAPTER 2

Cisco Dynamic Fabric Automation

Introduction

Cisco Dynamic Fabric Automation (DFA) configuration in this guide is organized based on the multi-tenancy segmentation that you use for various Cisco Nexus Series.

Multi-tenancy lite version supports:

- Cisco Nexus 5000 Series Switches
- Cisco Nexus 6000 Series Switches

Multi-tenancy full version supports:

- Cisco Nexus 7000 Series Switches

Overview of Cisco Dynamic Fabric Automation

Cisco Dynamic Fabric Automation (DFA) optimizes data centers through integration. The Cisco DFA architecture eliminates the need for overlay networks that can hinder traffic visibility and optimization and reduce scalability when physical server and virtual machine environments are integrated. This architecture enables zero-touch provisioning and greater orchestration, while delivering more predictable performance and latency for large cloud networks. The following building blocks are the foundation of Cisco DFA:

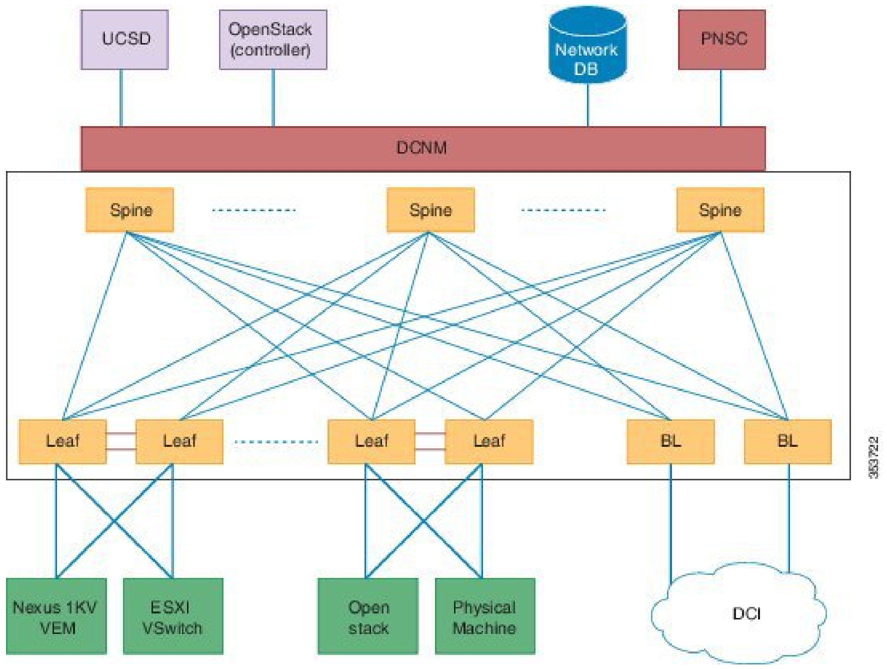
- Fabric Management — Simplifies workload visibility, optimizes troubleshooting, and automates fabric component configuration.
- Workload Automation — Integrates with automation and orchestration tools through northbound Application Programming Interfaces (APIs) and also provides control for provisioning fabric components by automatically applying templates that leverage southbound APIs and/or standard-based protocols. These automation mechanisms are also extensible to network services.
- Optimized Networking — Uses a simple distributed gateway mechanism to support any subnet, anywhere, concurrently. Existing redundancy models are also used to provide N+ redundancy across the entire fabric.

- Virtual Fabrics — Extends the boundaries of segmented environments to different routing and switching instances by using logical fabric isolation and segmentation within the fabric. All of these technologies can be combined to support hosting, cloud, and multi-tenancy environments.

Components in the Cisco DFA Network

The following figure depicts various components of Cisco Unified Fabric.

Figure 1: Components in the Cisco DFA Network



A leaf in a Cisco Unified Fabric is the node where virtual machines/physical machines are directly connected. A leaf is connected to all the spines in a Clos topology as shown in the figure above. The leaf manages VLAN allocation for physical and virtual machines wherever required. The leaf encapsulates tenant's traffic to the spine in a FabricPath encapsulation, either using segment ID or the VLAN based on the tenant traffic requirements. Cisco Unified Fabric leaf can also fetch network information via standard protocols like LDAP to automatically provision tenant networks.

A Border Leaf (BL) is a special purpose leaf typically required to reach the external world or provide connectivity for tenant networks that are spanning across a Cisco Unified Fabric. Any Cisco Nexus 5600 Platform Switches and Cisco Nexus 6000 and 7000 Series Switches can also be configured as a border leaf.

The spine nodes can load share the traffic between any leaf nodes and can also operate in a transit FabricPath mode essentially requiring no intervention in the spine configuration when a tenant is provisioned.

Cisco Prime Data Center Network Manager (DCNM) provides Power On Auto Provisioning (POAP), of the Cisco Unified Fabric nodes (spine, leaf and border leaf), cable-plan consistency, Unified Fabric Data Center Interconnect (DCI) automation and most importantly automated network provisioning with the help of northbound integration with both Virtual Machine Orchestrators and Services Orchestrators.

OpenStack DFA enabler seamlessly transforms and enables OpenStack compute nodes to be DFA capable. DFA enabler communicates network information to the Cisco Prime DCNM by invoking DCNM's rest APIs.

The compute nodes support Virtual Station Interface (VSI) Discovery and Configuration Protocol (VDP) to reliably pass the virtual machines network information to the leaf nodes.

Cisco Unified Computing System Director (UCSD) integrates perfectly with the Cisco Prime DCNM, making virtual machine network information available to the unified fabric leaf nodes. This clean and tighter integration provides for not only touchless automated tenant network provisioning but also a flexible choice of features that can be applied during the provisioning.

Cisco Prime Network Service Controller (PNSC) provides a way to orchestrate services for tenants in a touchless manner on the Cisco Unified Fabric. Cisco Prime DCNM network information is available within the PNSC to identify services required (edge firewall, load balancer) for a particular tenant network.

Cisco Nexus 1000V Series Switch compliments automated network provisioning of the tenant via the VSI discovery protocol (VDP) by reliably communicating virtual machine network information to the leaf via standards based implementation of 802.1QBG.

CHAPTER 3

Fabric Control Segment

FabricPath IS-IS Topology Discovery and Maintenance

FabricPath IS-IS is used as the link-state routing protocol in the DFA fabric. A single instance of FabricPath IS-IS runs in the fabric and carries switch reachability information for supporting multiple address families by leveraging MP-BGP. The native functionality supported by FabricPath IS-IS is leveraged for use in the DFA fabric. The primary functions include:

- The computation of shortest, equal-cost multipaths between any two switches in the fabric.
- The computation of multiple rooted distribution trees for the efficient and loop-free replication of multi-destination traffic within the fabric.

Switch IP address Resolution

The version 7.1(0)N1(1) of FabricPath IS-IS propagates the fabric-facing IPv4 and MAC address assigned through the fabric to each individual nodes within FabricPath IS-IS label switched paths (LSP). It thus provides IPv4 and IPv6 reachability among all the nodes within the fabric. It supports address resolution for the switch IP addresses, MAC address, and FabricPath switch ID which is used as encapsulation for switching data. It provides resilient multi-hop connectivity in the face of individual link or node failures, for both unicast paths and multi-destination trees.

Leaf, Border Leaf and Spine Common Commands

1. For FabricPath, each switch needs a unique switch ID. FabricPath IS-IS requires the configuration of unique switch ID. The switch ID is a 12-bit value that identifies the switch in the FabricPath network.

```
fabricpath switch-id $$SWITCH_ID$$
```

2. Enable the FabricPath IS-IS feature along with the DFA fabric feature:

```
install feature-set fabricpath
install feature-set fabric
feature-set fabricpath
feature-set fabric

feature fabric forwarding
feature interface-vlan
fabric forwarding identifier 1  //provides unique identification of a DFA fabric
```

3. Setting up the fabric facing FabricPath interfaces. Change the default timers of scale for FabricPath IS-IS interface parameter. The BFD option for FabricPath can be enabled only through the BFD feature. For example (note that BFD is not mandatory):

```
interface EthernetX/Y
switchport mode fabricpath
  fabricpath isis hello-interval 100
  fabricpath isis bfd
  fabricpath isis retransmit-interval 10
  fabricpath isis retransmit-throttle-interval 200
```

4. **Switch IP address Resolution**

   FabricPath IS-IS propagates the fabric-facing IPv4 and MAC address assigned through the fabric to each individual nodes within FabricPath IS-IS label switched paths (LSP). It thus provides IPv4 and IPv6 reachability among all the nodes within the fabric. It supports address resolution for the switch IP addresses, MAC address, and FabricPath switch ID which is used as encapsulation for switching data. It provides resilient multi-hop connectivity in the face of individual link or node failures, for both unicast paths and multi-destination trees.

   Set up the control-segment for the fabric to host the MP-BGP session. This segment is referred as backbone VLAN. Enable the interface VLAN feature, as this enables to assign an IP address. The command **fabric forwarding control-segment** signals to FabricPath IS-IS to distribute the IP and MAC addresses binding of the switch.

```
vlan $$BACKBONE_VLAN$$
  mode fabricpath

interface vlan $$BACKBONE_VLAN$$
  no shutdown
  ip address $$BACKBONE_IP$$/$$BACKBONE_PREFIX$$
  ipv6 forward
  mtu 9192
  fabric forwarding control-segment
```

   For example:

```
!control SVI
vlan 2
  mode fabricpath

interface Vlan 2
  no shutdown
  mtu 9192
  ip address 44.2.3.33/22
  ipv6 forward
  fabric forwarding control-segment
```

Spine Specific Configuration

Spine should be in transit mode. In transit mode the incoming and outgoing encapsulation is VN-Segment. Any change in transit mode requires a switch reload.

```
fabricpath mode transit
```

Leaf and Border Leaf Configuration

Multi-tenancy lite version

VN-Segment feature to support the switch global mapping of a VLAN to a VN-Segment ID and vice-versa.

For example:

```
feature vn-segment-vlan-based
```

Multi-tenancy full version

VN-Segment feature to support mapping of VN-Segment ID or VNI to a bridge-domain and vice-versa.

For example:

```
feature vni
```

Note Cisco NX-OS host attachment with auto-configuration at the border leaf is not supported.

The bridge-domain manager provides a CLI command to configure a bridge-domain as fabric control bridge-domain. The command used is **fabric-control** and has the following properties:

- Allocated from carved out bridge-domain range
- Only one bridge-domain or a VLAN can be of type fabric control
- Created as type fabric control and mode FabricPath by default
- Brought up on all FabricPath core ports in base topology and no explicit topology to VLAN configuration is needed

Note Use of VLAN 1 as fabric control is not allowed.

You can delete fabric control bridge-domains. When you delete fabric control bridge-domains, all iBGP adjacencies will go down. In order to change, you must revert current fabric control bridge-domain to type USER and then convert existing bridge-domain to fabric control.

Following is an example of how a bridge-domain can be converted to fabric control:

```
Switch(config)# bridge-domain 50
Switch(config-bd)# fabric-control

Switch(config)# bridge-domain 60
Switch(config-bd)# fabric-control

ERROR: bridge-domain 50 is already fabric-control. Cannot make bridge-domain 60 as
fabric-control.
```

Following is an example of how a VLAN can be converted to fabric control:

```
Switch(config)# vlan 10
Switch(config-vlan)# fabric-control

Switch(config)# vlan 20
Switch(config-vlan)# fabric-control

ERROR: vlan 10 is already fabric-control. Cannot make vlan 20 as fabric-control.
```

Fabric Control Segment Overview

DFA solution requires a transport protocol to help carry the IP to MAC mapping of the leaf switches to avoid having to use ARP. This information exchange is achieved using the FabricPath IS-IS protocol. The FabricPath IS-IS protocol continues to offer the Layer-2 address family functionality it offers for the FabricPath/TRILL solution. This functionality entails creating a mapping of switch ID to MAC addresses. Along with this FabricPath IS-IS also offers the Layer-3 address family functionality needed for BGP peers to successfully exchange peer information across the DFA fabric.

The IP address carried by FabricPath IS-IS is the one configured on the SVI referred to as control VLAN. The configuration is given below. In this case IP address 44.2.3.33 is distributed as the SVI is marked with the command **fabric forwarding control-segment**.

This IP address is used as the BGP end point when establishing the MP-BGP peering with the route reflector. It is also the BGP next hop of all routes advertised by this node:

```
!control SVI
vlan 2
  mode fabricpath

interface Vlan 2
  no shutdown
  mtu 9192
  ip address 44.2.3.33/22
  ipv6 forward
  fabric forwarding control-segment
```

CHAPTER 4

BGP Control Plane

Feature Information for BGP Control Plane

Table 1: Feature Information for BGP Control Plane

Feature	Releases	Feature Information
PoAP diagnostics	7.2(0)N1(1)	Included a new section on *POAP Diagnostics*. POAP failure can be detected with locator LED.
Default Route Advertisement	7.2(0)N1(1)	Included a new section on *Default Route Advertisement*.

Border Leaf U-shape support	7.2(0)D1(1)
	7.2(0)N1(1)
		Included a new section on *Border Leaf Deployment Consideration* to support U-shape connectivity.

BGP Control Plane Setup

Multi-Protocol BGP (MP-BGP) is the primary protocol for exchanging host, subnet and default routes for IPv4 and IPv6 address families. MP-BGP based Control-Plane using EVPN NLRI (Network Layer Reachability Information) to transport end host information (IP and MAC) is used to transport the EVPN address family.

The following sections describe the reason for the POAP setting for BGP. The BGP configuration is same on all leaf nodes in the fabric. There are some additional knobs for the leaf node that are in the role of border leaf, we recommend that there be more than one border leaf in the fabric for redundancy reason. There are one or more switches that act as route reflectors (RR) that are configured on the spine, they have configuration related to being route reflectors. The following sections explain the general settings that apply to all leaf nodes, then knobs specific to border leaf and finally the spine knobs to act as a route reflector specific knob.

General BGP Configuration

Route-Target

Auto generated at the leaf and border leaf by combining the fabric ASN and Layer-3 Virtual Network Identifier (VNI).

The BGP route-target extended community is a path attribute shared by one or more routes in an UPDATE Message. Routes can be imported by using route-target as filter. Here, route-target carries a 2-byte ASN and a 4-byte VNI.

FABRIC ASN: VNI

Route Distinguisher

Auto generated at the leaf and border leaf by combining router ID and VRF ID. By making same route originated from different switches have a different Route Distinguisher (RD), the routes become unique. In MP-BGP, each route is uniquely qualified by a 8-byte RD. Here, the RD carries a 4-byte router ID and a 2-byte VRF ID.

Note The router ID is the same as the IP address configured on the backbone VLAN/SVI for BGP peering.

Switch router ID: local vrf id

Following is a sample configuration:

```
vrf context CiscoLive:Part4
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target both auto evpn
  address-family ipv6 unicast
    route-target both auto
    route-target both auto evpn
```

If the router ID is 220.1.1.1, local VRF ID is 4, Fabric AS is 65000, VNI is 65004 then RD = 220.1.1.1:4 and RT = 100:65004.

Add Path Support

The use of 'Add Path' is to allow one or more paths on a leaf and border leaf node to reach a given host. This facilitates Equal Cost Multipath (ECMP), faster convergence and host moves.

For example, spine acting as a router reflector:

```
route-map ALL-PATHS permit 10
  set path-selection all advertise

router bgp 65000

  address-family ipv4 unicast
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths selection route-map ALL-PATHS
```

At the leaf and border leaf:

```
route-map ALL-PATHS permit 10
  set path-selection all advertise
```

```
router bgp 65000

  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map bgp_next_hop_filter
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
```

General BGP settings for all Leaf nodes including Border Leaf

1. Feature BGP: Enables the feature on the box, needed on all leaf, border leaf and any spine that acts as route reflector.
2. BGP Router Autonomous System Number: All the nodes belong to one AS, this variable defines the AS value for the whole fabric.
    1. Every leaf node is connected to one or more route reflector neighbors. We recommend that you configure two route reflectors for redundancy purpose. At least one route reflector is needed in the fabric.

```
router bgp 65103

 router-id 192.0.2.1
    address-family ipv4 unicast
    address-family l2vpn evpn
 neighbor 192.0.2.10 remote-as 65103
    update-source loopback0
    address-family l2vpn evpn
      send-community both
 neighbor 192.0.2.20 remote-as 65103
    update-source loopback0
    address-family l2vpn evpn
      send-community both
```

- Due to simplified topology, typically most nodes except the route reflector have not more than two BGP sessions, thus more aggressive BGP timers can be used to speed up convergence. The following command is used to speed up convergence in the fabric for node failures. The timer depends on the speed of convergence of FabricPath IS-IS and the removal of the BGP next hop IP address leading to withdrawal of the propagation of VRF prefixes. For more information, see examples in the following routing policy section.

Note The convergence time for FabricPath IS-IS is in subseconds as the number of nodes and label switched path (LSP) is less.

Usage example:

```
router bgp 65000

   address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map bgp_next_hop_filter
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
```

- Default ECMP in unified fabric is set to 2. It is a balance between redundancy and saving hardware resources. This satisfies the common case of a vPC pair generating the same route and multiple border leaf nodes hosting the same VRF. For more information, see examples in the following routing policy section.

BGP Routing Policy

Cisco NX-OS operating system requires, that any route distribution passes through a route-map, to filter the distribution. Here are the policy statements that are configured by default through POAP. These are the common needs of unicast forwarding for hosts. This policy is configured on leaf and border leaf nodes.

1. Match any IPv4 address.

```
ip access-list HOSTS
  10 permit ip any any
```

2. Match any IPv6 address.

```
ipv6 access-list V6HOSTS
  10 permit ipv6 any any
```

3. The following route-maps allow the redistribution of all routes (IPv4 and IPv6 respectively) except for those learned over the control VLAN interface (backbone VLAN is used to set up the BGP topology). These route-maps are generally used for host redistribution via the HMM protocol.

```
route-map FABRIC-RMAP-REDIST-HOST deny 10
  match interface Vlan $$BACKBONE_VLAN$$
route-map FABRIC-RMAP-REDIST-HOST permit 20
  match ip address HOSTS

route-map FABRIC-RMAP-REDIST-V6HOST  deny 10
  match interface Vlan $$BACKBONE_VLAN$$
route-map FABRIC-RMAP-REDIST-V6HOST permit 20
  match ip address V6HOSTS

router bgp 65000

address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map bgp_next_hop_filter
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
```

4. The following route-map is used to redistribute server facing subnets. If the subnet route is tagged with the special value of 12345 then it will be redistributed. The same route-map works for IPv6 and IPv4 routes. Generally the default host facing configuration profiles will tag the subnet with this tag. It is highly

recommended that it should not be changed. If there is a subnet that does not require redistribution via BGP, then this tag should not be placed on it. There are several reasons for redistributing subnet address.

1. It enables border leaf to implement filtering policy for extended subnets.
2. It enables fabric to optimize Forwarding Information Base (FIB) usage in certain scenarios.

   Multi-tenancy lite version

```
route-map FABRIC-RMAP-REDIST-SUBNET permit 10
  match tag 12345

interface Vlan3509
  no shutdown
  vrf member CiscoLive:Part4
  no ip redirects
  ip address 17.1.0.1/24 tag 12345
  no ipv6 redirects
  fabric forwarding mode proxy-gateway

route-map FABRIC-RMAP-REDIST-SUBNET permit 10
  match tag 12345

router bgp 65000

vrf CiscoLive:Part4
    address-family ipv4 unicast
      redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
      redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
      maximum-paths ibgp 2
```

   Multi-tenancy full version

```
route-map FABRIC-RMAP-REDIST-SUBNET permit 10
  match tag 12345

interface bdi3509
  no shutdown
  vrf member CiscoLive:Part4
  no ip redirects
  ip address 17.1.0.1/24 tag 12345
  no ipv6 redirects
  fabric forwarding mode proxy-gateway

route-map FABRIC-RMAP-REDIST-SUBNET permit 10
  match tag 12345

router bgp 65000

vrf CiscoLive:Part4
    address-family ipv4 unicast
      redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
      redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
      maximum-paths ibgp 2
```

5. BGP next hop filter: For fast convergence, block the next hop resolution via the control subnet address. Generally all the BGP speakers within the fabric are on the same subnet. FabricPath IS-IS distributes the 32-bit local address for each switch via Link State Packet (LSP). For example, if the control subnet is 10.1.0.0/16, switches may have local IP addresses as 10.1.1.1, 10.1.1.2, 10.1.1.3 and so on. Essentially

each switch has the control backbone VLAN subnet 10.1.0.0 in the global routing table due to local configuration. It also has the 32-bit addresses learnt by FabricPath IS-IS. Suppose due to vPC pair or multiple border leaf there exists an ECMP path to prefix X. Consider if the path is advertised by 10.1.1.1 and 10.1.1.2. In steady state every other leaf in fabric will have an ECMP path to this prefix with the BGP next hop resolved via 10.1.1.1 and 10.1.1.2, now suppose switch 10.1.1.1 gets reloaded for upgrade, FabricPath IS-IS removes 10.1.1.1 immediately (subsecond) from all leaf nodes and withdraws the prefix propagation immediately. If **bgp_next_hop_filter** is not configured then convergence is delayed, as the next hop is resolved via the control subnet and route is not removed till BGP session timeout of route reflector with border leaf with IP address 10.1.1.1.

Note The Cisco NX-OS is an event trigger that uses next hop tracking and does not wait for BGP scan time.

In order to speed up convergence to subsecond, the following route-map is configured. It means if the route's next hop is resolved within control subnet, then it does not allow the subnet route to be used to resolve the next hop. For example, if FabricPath IS-IS removed the 32-bit next hop in subsecond but without the filter the route gets resolved through control subnet. Hence, route is not removed on FabricPath IS-IS event. It gets removed only when BGP session between RR and leaf and border leaf gets removed.

```
ip prefix-list control-subnet seq 100 permit $$BGP_CLIENT_SUBNET$$

route-map bgp_next_hop_filter deny 100
  match ip address prefix-list control-subnet
route-map bgp_next_hop_filter permit 200
  match ip address HOSTS

ip prefix-list control-subnet seq 100 permit 44.2.0.0/22
ip access-list HOSTS
  10 permit ip any any
route-map bgp_next_hop_filter deny 100
  match ip address prefix-list control-subnet
route-map bgp_next_hop_filter permit 200
  match ip address HOSTS
```

Usage example:

```
router bgp 65000

  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map bgp_next_hop_filter
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
```

BGP Settings for Border Leaf

The previous configurations and POAP settings apply to all leaf nodes including border leaf and some specific to route reflector. The following are settings that apply only to the border leaf.

1. **Default route generation from Border Leaf**: There are two options to do this:

- Option 1 is to advertise a default originate for all tenants by using a special route-target (RT) value. All the tenants that wants to use this border leaf will put this RT in the respective RT import statements as shown below, use this option if the number of VRFs is lesser than the maximum VRFs supported by border leaf and the total number of routes is also within the capability of the device, then use this variable to set up a default route for all VRFs. Option 1 is the default setting in the border leaf POAP template.
- Use one default route for all VRFs:

```
address-family vpnv4 unicast
   default-information originate always rd $$BACKBONE_IP$$:$$BGP_AS$$ route-target
 $$BGP_AS$$:$$BGP_RT_VNI$$

router bgp 65000

  address-family vpnv4 unicast
    default-information originate always rd 192.16.1.113:3 route-target 65000:9999
```

  At interior leaf nodes, within every tenant:

```
vrf context CiscoLive:Part4
  vni 65004
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target import 65000:9999

vrf context CiscoLive:Part3
  vni 65005
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target import 65000:9999
BGP_RT_VNI, defaultValue=9999;
```

- Option 2 is to advertise VRF specific default originate from the border leaf. With this method, tenants can be load shared by different border leaf nodes by advertising only those tenants that border leaf is hosting. The border leaf scale is achieved by using per VRF default route generation. In that case, omit the default route generation by omitting this variable. The border leaf auto-configuration generates per VRF default route. If not using that feature then configure manually.
- per VRF default originate: This does not come with POAP, either to be entered manually or through border leaf auto-configuration for Layer-3 extension:

```
address-family vpnv4 unicast
         default-information originate rd router ID:Local VRF ID route-target
Fabric ASN:L3 Segment ID (VNI)
```

  There is no need for a special route-target import statement at interior leaf. At the border leaf add the default information originate command per tenant under vpnv4 address family. The RD is constructed with border leaf BGP Router ID: Integer

  The integer is a two by value, which is unique per tenant. The local VRF ID obtained by show VRF <vrf name> detail command can be used as the integer value 4 in the following example show VRF vpn1 in detail:

```
VRF-Name: vpn1, VRF-ID: 4, State: Up
```

```
    VPNID: unknown
    RD: 220.1.1.1:4
    VNI: 65004
    Max Routes: 0  Mid-Threshold: 0
    Table-ID: 0x80000003, AF: IPv6, Fwd-ID: 0x80000003, State: Up
    Table-ID: 0x00000003, AF: IPv4, Fwd-ID: 0x00000003, State: Up
```

The route-target is made up by Fabric ASN: VNI

At leaf node:

```
vrf context CiscoLive:Part4
  vni 65004
  rd auto
  address-family ipv4 unicast
    route-target both auto

vrf context CiscoLive:Part3
  vni 65005
  rd auto
  address-family ipv4 unicast
    route-target both auto
```

At border leaf:

```
router bgp 65000

  address-family vpnv4 unicast
   default-information originate always rd 192.16.1.113:4 route-target 65000:65004

   default-information originate always rd 192.16.1.113:5 route-target 65000:65005
```

2. **Fabric Site of Origin (SOO)**: Border leaf generates a fabric SOO and attaches it to routing updates going from outside the fabric to inside and inside the fabric to the outside. Fabric SOO is constructed by joining fabric ID and fabric AS. Interior leaf nodes use the fabric forwarding identifier and the local AS value to determine about fabric SOO. Only border leaf inserts SOO attribute in BGP updates.

```
fabric-soo $$BGP_AS$$:$$FABRIC_ID$$
```

POAP.FABRIC_ID: Fabric Identifier is one per fabric, every fabric must be configured with a unique integer as fabric ID. This helps to troubleshoot, which fabric a route is originated from and also to prevent loop. We recommend to start with 1 for first fabric and increment monotonically. The vPC leaf nodes insert SOO for dually homed hosts. They construct it with the following values:

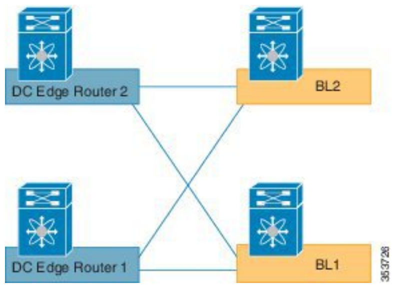
```
vPC Domain ID: Fabric Identifier

fabric forwarding identifier 1
router bgp 65000
  router-id 44.2.3.63
  fabric-soo 65000:1
```

3. (Optional) For release 7.1(0)N1(1), POAP offers port-channels as only interface option between border leaf and DC edge routers. If you do not want to use port-channel, skip this step. Other types of interfaces can be configured manually. This limitation will be fixed in the future releases. Ensure that port admin is up and is not a switch port.

For border leaf/edge router select the port-channel/interface ID as well as the interface range port-channel(s) towards DC edge router: The border leaf POAP optionally provides user to configure a port-channel towards each of the DC edge routers it is neighbored with. We recommend that you configure this interface as port-channel even if there is only one member port. This should not be configured as a switch port so that Layer-3 sub interfaces can be configured on this for Layer-3 extension via sub interfaces. This is port-channel on which the border leaf auto-configuration will deploy sub interfaces for Layer-3 extension outside the fabric. It should match the value configured in Cisco Prime DCNM when pairing border leaf with DC edge router.

The following is the recommended topology for full redundancy. This POAP section is to enable it. One or two DC edge routers should be connected to border leaf. Two is the recommended number.

Figure 2: Recommended Topology for Full Redundancy



Note As there are multiple links to two or more edge routers, even if a link to an edge router goes down, it can still advertise the default route into the fabric without blackholing the traffic.

4. For border leaf/edge router select the port-channel/interface for default VRF peering: Global routing table peering with DC edge box: Border leaf provides user with prompt for configuring a sub interface on the port towards DC edge box and also the corresponding BGP session parameters. This is optional depending upon customer topology, need for default table routing and model used for internet access.

5. The border leaf has to be configured with switch role border.

```
fabric forwarding switch-role border
```

6. Set up the LDAP connection to the BL-DCI table. This is the table that enables auto-configuration of border leaf Layer-3 extension to the DC edge router. This is only done at border leaf in addition to the other LDAP tables set up at leaf nodes.

```
fabric database type network
server protocol ldap host rio-dcnm101a.cisco.com vrf management
db-table ou=networks,dc=cisco,dc=com key-type 1
db-security user cn=reader,dc=cisco,dc=com password 7 iwfw1c
fabric database type profile
server protocol ldap host rio-dcnm101a.cisco.com vrf management
db-table ou=profiles,dc=cisco,dc=com
db-security user cn=reader,dc=cisco,dc=com password 7 iwfw1c
fabric database type partition
server protocol ldap host rio-dcnm101a.cisco.com vrf management
db-table ou=partitions,dc=cisco,dc=com
db-security user cn=reader,dc=cisco,dc=com password 7 iwfw1c
fabric database type bl-dci
server protocol ldap host rio-dcnm101a.cisco.com vrf management
db-table ou=bl-dcis,dc=cisco,dc=com
db-security user cn=reader,dc=cisco,dc=com password 7 iwfw1c
```

Usage example:

```
fabric database type bl-dci
    server protocol ldap host ldap-server1.cisco.com vrf management
    db-security user cn=reader,dc=cisco,dc=com password1

db-security user admin password cis
  server protocol ldap host ldap-server2.cisco.com vrf management
    db-table ou=bl-dcis,dc=cisco,dc=com
   db-security user cn=reader,dc=cisco,dc=com password1
```

7. Border leaf specific tenant profile: Border leaf supports border leaf Layer-3 extension auto-configuration. Thus it needs a different profile than what is used by interior leaf nodes. The LDAP only allows one profile per tenant as the lookup key is only tenant name. The following command is used to override this locally at border leaf:

```
fabric database override-vrf-profile vrf-common-universal-bl
```

8. Border leaf should not accept default route from other border leaf nodes in the same fabric. This breaks ASBR function of border leaf and also leaks default route outside the fabric. The following commands are used to filter default route coming from route reflector neighbor.

Note The route-map 'deny-default-route' is required only when you run the previous versions of Cisco NX-OS 7.2(0)N1(1). From Cisco NX-OS 7.2(0)N1(1) or later, the import of default route advertised from the other border leaf node in the same fabric is supported.

```
ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
route-map deny-default-route deny 100
  match ip address prefix-list default-route
route-map deny-default-route permit 200
  match ip address HOSTS

router bgp 65000

!Peering to the first RR
neighbor 44.2.0.101 remote-as 65000
    address-family ipv4 unicast
      send-community both
      route-map deny-default-route in
      next-hop-self
    address-family ipv6 unicast
      send-community extended
    address-family vpnv4 unicast
      send-community extended
      route-map deny-default-route in
    address-family vpnv6 unicast
      send-community extended
    address-family ipv4 mvpn
      send-community extended
    address-family ipv6 mvpn
      send-community extended
!Peering to the second RR
  neighbor 44.2.0.144 remote-as 65000
    address-family ipv4 unicast
      send-community both
      route-map deny-default-route in
      next-hop-self
    address-family ipv6 unicast
      send-community extended
    address-family vpnv4 unicast
      send-community extended
      route-map deny-default-route in
    address-family vpnv6 unicast
      send-community extended
    address-family ipv4 mvpn
      send-community extended
    address-family ipv6 mvpn
```

9. Host based auto-configuration is disabled at border leaf. The feature evb along with the VDP configuration is missing on border leaf template for the same reason.

```
platform fabric database dot1q disable
```

Note Cisco NX-OS host attachment with auto-config at the border leaf is not supported.

10. For example for extension of tenant towards DC edge, see DC edge router on Appendix, on page 135.

Default Route Advertisement

Default route advertisement for the default VRF from the border leaf

In case if the interior leaf nodes need to use the default VRF, the border leaf can advertise a default route towards the fabric.

There are several ways to do this, two are explained below.

Default route advertisement using redistribution of static route

Advantage

- Use this approach to withdraw static route from the fabric when external interfaces goes down.

Disadvantage

- The default route points towards external neighbors. Even in the presence of external default route.
- The static route is preferred over the external default route.

Default route advertisement using the 'default-originate' command

Default route advertisement using the **default-originate** command under the peer neighbor configuration context for fabric route reflector.

Advantage

- Simple to configure.

Disadvantage

- Does not withdraw default route even if external connectivity is lost.

Default route advertisement using redistribution of static route details

The recommended way to achieve this is implemented in the border leaf POAP templates. The POAP templates ensure that:

- The default route advertised by the Border Leaf does not leave the fabric by:
  - Attaching the well known community NO_EXPORT_COMMUNITY.
  - A deny route-map for default route on all external neighbors of border leaf.
- The LOCAL_PREFERENCE is set to 50, which is lower than the default preference of 100 when this route is received by other border leaf nodes. This ensures that the border leaf prefers the external default route.
- The admin distance of the static route at the border leaf is set to 254, so that the default routes learnt from external neighbors are always preferred over the locally configured static route.

Route Map and Prefix lists Configuration

```
ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
route-map DEFAULT-ROUTE-MODIFY permit 100
  match ip address prefix-list default-route
  set local-preference 50
  set community no-export
route-map DEFAULT-ROUTE-MODIFY permit 1000
route-map DEFAULT-ROUTE-MODIFY-V6 permit 100
  match ipv6 address prefix-list default-route-v6
  set local-preference 50
  set community no-export
route-map DEFAULT-ROUTE-MODIFY-V6 permit 1000

route-map DENY-DEFAULT-ROUTE deny 10
  match ip address prefix-list default-route
route-map DENY-DEFAULT-ROUTE permit 1000

route-map FABRIC-RMAP-REDIST-STATIC permit 10
  match ip address prefix-list default-route

route-map ALL-PATHS permit 10
  set path-selection all advertise
```

Configuration details

Configuration specific to default route origination is given below.

Two box border leaf solution.

```
!Dc Edge facing sub interfaces

interface Ethernet1/35.10
  encapsulation dot1Q 10
  ip address 30.1.1.1/24

interface Ethernet1/36.10
  encapsulation dot1Q 10
  ip address 40.1.1.1/24
```

```
!Static route towards DC Edge

ip route 0.0.0.0/0 30.1.1.2 254
ip route 0.0.0.0/0 40.1.1.2 254


!Relevant BGP configuration

router bgp 65000
  router-id 128.89.0.20
  fabric-soo 65000:1
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    redistribute static route-map FABRIC-RMAP-REDIST-STATIC
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map BGP_NEXT_HOP_FILTER
    default-information originate
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
/**RR neighbor**/
neighbor 128.89.0.100 remote-as 65000
    address-family ipv4 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY in
      next-hop-self

[SNIP]
/*external neighbors */
neighbor 30.1.1.2 remote-as 300
    peer-type fabric-external
    address-family ipv4 unicast
      send-community both
      route-map DENY-DEFAULT-ROUTE out
neighbor 40.1.1.2 remote-as 300
    peer-type fabric-external
    address-family ipv4 unicast
      send-community both
      route-map DENY-DEFAULT-ROUTE out
```

default-information originate

Allows default route to be redistributed. By default, the default route is not redistributed without explicitly allowing the redistribution through this command.

Two Box Border leaf

For the two box solution, point the static route next hop to the DC-EDGE router address for the sub interface. This will ensure that the default route is withdrawn when the interface goes down. This is automatically done if a POAP template is used to configure the interface and sub-interface towards DC-EDGE box.

BorderPe

There are two options for BorderPe based on your preference:

1. Point the default static route to MPLS VPN facing interfaces.
    - Will be withdrawn if external connectivity is lost.
    - POAP template uses this approach for IPv4.

2. Point the default static route to NULL0.
   - Will not be withdrawn if external connectivity is lost.

Default route using default originate commands under RR neighbors Details

Default route can be advertised towards fabric by default originate command as shown below.

Route Map and Prefix lists Configuration.

```
ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
ipv6 prefix-list default-route-v6 seq 5 permit 0::/0
route-map DENY-DEFAULT-ROUTE deny 10
  match ip address prefix-list default-route
route-map DENY-DEFAULT-ROUTE permit 1000
```

BGP configuration.

```
router bgp 65000
  router-id 128.89.0.20
  fabric-soo 65000:1
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map BGP_NEXT_HOP_FILTER
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
/**External neighbors */
neighbor 30.1.1.2 remote-as 300
    peer-type fabric-external
    address-family ipv4 unicast
      send-community both
      route-map DENY-DEFAULT-ROUTE out

  neighbor 40.1.1.2 remote-as 300
    peer-type fabric-external
    address-family ipv4 unicast
      send-community both
      route-map DENY-DEFAULT-ROUTE out
/**RR neighbor**//

  neighbor 128.89.0.100 remote-as 65000
    address-family ipv4 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY in
      default-originate
      next-hop-self
```

Border Leaf Deployment Consideration

The previous configurations and description applies to border leaf topologies with full-meshed peering with the DC edge routers. Full-meshed connectivity between border leaf and DC edge router along with node redundancy is recommended, as this topology supports the highest level of redundancy. However there are certain deployments, where the full-meshed approach cannot be deployed.

For example, consider a case where the physical installation of the border leaf nodes and DC edge routers are in different buildings and you only have limited fiber available. Such topologies, where each border leaf has only a single link to the DC edge router is often called U-shape.

By default, the border leaf when using default route configuration according to option 1 (default route injection for all VRF with default-information originate always configured under the VPNv4/6 address family) injects the default route in the fabric independent if the external link towards the DC edge router is down or up.

In U-shape topologies this could cause blackholing for certain flows, as the traffic sourced on the leaf switch is hashed along the two default routes (ECMP) injected by the two border leaf nodes.

Cisco NX-OS version 7.2(0)D1(1) or 7.2(0)N1(1) or later is required on the border leaf as this software version supports the U-shape topology. The DCNM 7.2(1) with the V3 POAP templates (for example, Fabric_N5600_N6K_BorderLeaf_v3) provides the required configuration.

The border leaf POAP template provides the required configuration with the specific route-maps to avoid blackholing as shown below.

```
ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
ipv6 prefix-list default-route-v6 seq 5 permit 0::/0

route-map DEFAULT-ROUTE-MODIFY permit 100
  match ip address prefix-list default-route
  set local-preference 50
route-map DEFAULT-ROUTE-MODIFY permit 1000

router bgp 65000

!Peering to the first RR
neighbor 44.2.0.101 remote-as 65000
    address-family ipv4 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY in
      next-hop-self
    address-family ipv6 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY-V6 in
      next-hop-self
    address-family vpnv4 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY in
    address-family vpnv6 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY-V6 in
    address-family ipv4 mvpn
      send-community both
    address-family ipv6 mvpn
      send-community both

!Peering to the second RR
neighbor 44.2.0.144 remote-as 65000
    address-family ipv4 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY in
      next-hop-self
    address-family ipv6 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY-V6 in
      next-hop-self
    address-family vpnv4 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY in
    address-family vpnv6 unicast
      send-community both
      route-map DEFAULT-ROUTE-MODIFY-V6 in
    address-family ipv4 mvpn
      send-community both
    address-family ipv6 mvpn
      send-community both
```

Note The route-map 'deny-default-route' as shown in the previous section is replaced by the route-map 'DEFAULT-ROUTE-MODIFY'.

The route-map "DEFAULT-ROUTE-MODIFY" along the additional BGP route-target import statement 65000:9999 (same as on the interior leaf) will re-import the default route advertised by the other border leaf. This default route is imported per specific tenant with a lower local preference and is only installed in the forwarding table when the external learnt default route from the DC edge router is unavailable.

At border leaf, within every tenant:

```
vrf context CiscoLive:Part4
  vni 65004
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target import 65000:9999

vrf context CiscoLive:Part3
  vni 65005
  rd auto
  address-family ipv4 unicast
    route-target both auto
    route-target import 65000:9999
```

POAP Diagnostics

POAP failure can be detected with locator LED. When the POAP process starts, the locator-LED will flash the pattern 21 (flashing twice, short pause, flashing once, long pause) to indicate that POAP is in progress.

The device has several LEDs such as chassis LED, status LED, port LED, and so on. For PoAP diagnostics, you must follow the chassis (beacon) LED flashing in pattern 21.

Given below are the LED flashing in a pattern that is recognizable and visible to human eyes:

Table 2: Chassis LED Pattern

LED Pattern	Description
Blue LED - two long flashes, pause, one short flash, long pause	PoAP is currently running. If this pattern is flashing and not turned off after a considerable amount of time lapse, it indicates PoAP is in error condition such as DHCP discovery failure or script execution failure.
No light	PoAP is not running.

Router Reflector Configuration

Use subnet for neighbor address so that each neighbor does not have to be explicitly configured.

For example, if control subnet is 192.168.99.0/24 and fabric as is 65101.

```
router bgp 65101

 router-id 192.168.99.1
 address-family ipv4 unicast
   maximum-paths ibgp 2
   nexthop trigger-delay critical 250 non-critical 10000
   additional-paths send
   additional-paths selection route-map ALL-PATHS
 address-family ipv6 unicast
   maximum-paths ibgp 2
   nexthop trigger-delay critical 250 non-critical 10000
   additional-paths send
   additional-paths selection route-map ALL-PATHS
 address-family vpnv4 unicast
   nexthop trigger-delay critical 250 non-critical 10000
   additional-paths send
   additional-paths selection route-map ALL-PATHS
 address-family vpnv6 unicast
   nexthop trigger-delay critical 250 non-critical 10000
   additional-paths send
   additional-paths selection route-map ALL-PATHS
 address-family ipv4 mvpn
   nexthop trigger-delay critical 250 non-critical 10000
   additional-paths send
   additional-paths selection route-map ALL-PATHS
 address-family ipv6 mvpn
   nexthop trigger-delay critical 250 non-critical 10000
   additional-paths send
   additional-paths selection route-map ALL-PATHS
   neighbor 192.168.99.0/24 remote-as 65101
 address-family ipv4 unicast
   send-community both
   route-reflector-client
 address-family ipv6 unicast
   send-community extended
   route-reflector-client
 address-family vpnv4 unicast
   send-community extended
   route-reflector-client
 address-family vpnv6 unicast
   send-community extended
   route-reflector-client
 address-family ipv4 mvpn
   send-community extended
   route-reflector-client
 address-family ipv6 mvpn
   send-community extended
   route-reflector-client
```

CHAPTER 5

Cable Management

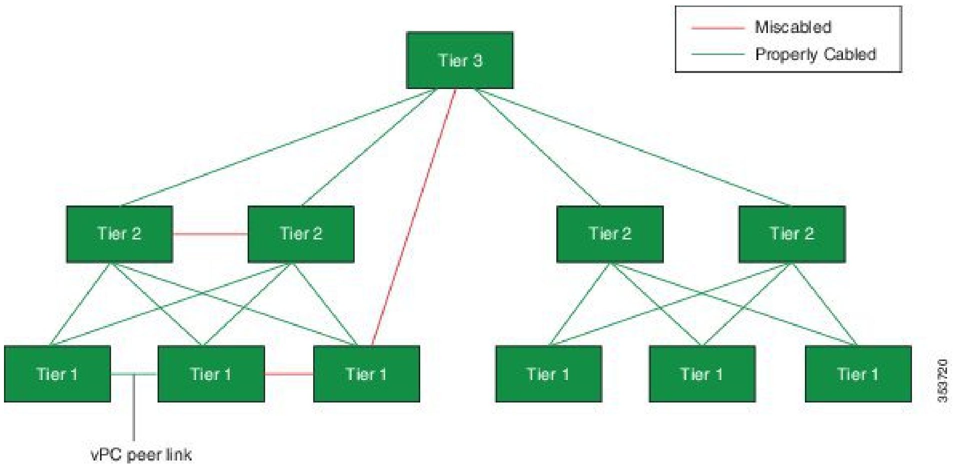
Cable Management

In a highly meshed network such as Clos topology based network fabric, miscabling can be a pragmatic problem leading to painful troubleshooting without sufficient support. The cable management feature calls out for two mechanisms to address the miscabling issues caused due to human errors. The first mechanism is based on the tier-based checks and the second mechanism is based on a user-defined cabling plan.

Tier-Based Miscabling Detection

The figure below shows the properly cabled (in green) and miscabled (in red) links in a hierarchical network designed based on Clos design principle. The cable verification rules based on tier-level is applicable only for Clos topologies. Each box represents a switch in the network.

Figure 3: Tier-Based Miscabling Detection



As shown in the figure, every switch in a stage is associated with the corresponding tier-level number assigned to the stage it is in. That means that all the leaf switches (in the lowest-level Clos stage) are provisioned with a tier-level of 1, the next higher-level stage switches (1st stage of spine switches) are provisioned with a tier-level value of 2, and the next higher-level stage switches (2nd stage of spine switches) are provisioned with a tier-level of 3, so on.

As an example the tier level configuration on a level 2 spine switch is performed as follows:

```
Switch(config)# fabric connectivity tier 2
```

Note You can configure this automatically using the POAP template on DCNM when the switch is designated as a spine or leaf.

You have to enable features 'cable-management' and 'LLDP' and this command enables tier-based checks on the switch.

Every switch in the network advertises its tier-level number (only if one is configured with), as part of the link layer PDUs (of LLDP) in addition to the chassis-id, and physical port-id to their adjacent peers.

The receiver of the link layer PDU performs the following algorithm checks to validate link connections with sender per Clos network design rules:

1. Performed only at leaf switches (switches assigned with a tier-level of 1)
    - The received tier level from adjacent link partners must be 2. There will be an exception to this rule. If the receiving link is vPC peer link (detected automatically via LLDP), a remote tier-level of 1 will be allowed.
2. Performed only at spine switches (switches assigned with tier-level > 1)
    - The received tier level from adjacent link partners must be either (my_tier_level + 1) or (my_tier_level - 1), where my_tier_level is the tier level assigned to the spine switch performing the check.

If any cabling inconsistency is detected while validating the remote link information from the adjacent switches, then the miscabling action (error disabling the port by default) is performed and a cabling mismatch error is logged. The logged error includes necessary information about the involved local chassis-id, local port-id, local tier-level, and remote switch details such as peer chassis-id, peer port-id, and peer tier-level.

The following example shows the fabric connectivity neighbors with tier details.

```
Switch# show fabric connectivity neighbors all

--------------------------------------------------------------------------------
Local System:
Device Tier Config:          Enabled       Device Tier Level:               1
Tier-mismatch Retry Config:  Disabled      Tier-mismatch Retry Timeout:     0
Cable Plan Check:            Enabled
DeviceID:  poap-integ-leaf1                ChassisID:  0003.0111.0008
--------------------------------------------------------------------------------
         Codes: (Ok) Normal, (ErrT) Tier error , (ErrC) Cabling Plan error,
                (Ok/VPC) VPC leaf-to-leaf connection, (S) Stale entry
                (Unkn) Unknown Tier, (Enp) Plan Entry not present
Neighbor Table:
--------------------------------------------------------------------------------
Local      DeviceID        PortID       Tier  Cable-Plan          Status
Intf

Eth2/1     spine0          Eth2/1       2     spine2,Eth2/1       ErrC
Eth2/6     spine5          Eth2/2       2     spine5,Eth2/2       Ok
```

```
Eth2/5     spine4          Eth2/2       2     spine0,Eth2/2       ErrC
Eth2/3     spine6          Eth2/2       2     Enp                 Ok
Eth2/4     spine3          Eth2/2       2     spine7,Eth2/1       ErrC

poap-integ-spine0# show fabric connectivity neighbors errors
--------------------------------------------------------------------------------
Local System:
Device Tier Config:          Enabled       Device Tier Level:                2
Tier-mismatch Retry Config:  Disabled      Tier-mismatch Retry Timeout:      0
Cable Plan Check:            Disabled
DeviceID:  poap-integ-spine0               ChassisID:  0003.0210.0008
--------------------------------------------------------------------------------
         Codes: (Ok) Normal, (ErrT) Tier error , (ErrC) Cabling Plan error,
                (Ok/VPC) VPC leaf-to-leaf connection, (S) Stale entry
                (Unkn) Unknown Tier
Neighbors with Miscabling Warning/Error:
--------------------------------------------------------------------------------
Local           DeviceID             ChassisID        PortID       Tier Status
Interface

Ethernet2/1     poap-integ-leaf0     0003.0110.0001   Eth2/1       2    ErrT,S
Ethernet2/2     poap-integ-leaf1     0003.0111.0001   Eth2/1       2    ErrT,S
Total entries displayed: 2
```

In case switch is configured with a tier level that is connected to another switch that does not support this feature and hence does not send tier level information, then the link will not be checked for miscabling.

Overview of Cable-Plan Miscabling Detection

Cable-plans also rely on LLDP to exchange information, for example switch ID, port number, between two ends of a cable connection. This information is carried in vendor specific LLDP TLVs.

You must first import a valid cable-plan to the switch and check against the information in these received TLVs. Cable-plans are XML files that fully or partially describe the topology of the data center.

Once a plan is imported, checks will happen to make sure that the imported plan matches the received TLV information. If it matches nothing will happen, however if there is a mismatch then the miscabling action will be taken on the port in question (err-disable by default). The user can change the actions taken on a miscabling event.

Administrators will then be warned via SysLog of any such miscablings, and they will take the appropriate action to fix the problem (in most cases like above, they will simply need to fix the miscabling to the correct configuration as described in the cable-plan, and clear the err-disable).

To enable miscabling detection, configure the following feature commands:

```
Switch(config)# feature lldp
Switch(config)# feature cable-management
```

Cable-Plan XML Schema

The following schema of cable-plan XML is for a data center with two spine switches, and three leaf switches:

```
<?xml version="1.0" encoding="UTF-8"?>
<CISCO_NETWORK_TYPES version="1.0" xmlns="http://www.cisco.com/cableplan/Schema2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cisco.com/cableplan/Schema2 nxos-cable-plan-schema.xsd ">
```

```
  <DATA_CENTER networkLocation="san-jose" idFormat="hostname">
    <CHASSIS_INFO sourceChassis="spine1" type="n7k">
      <LINK_INFO sourcePort="Eth2/1" destChassis="leaf1" destPort="Eth2/1"/>
      <LINK_INFO sourcePort="Eth2/2" destChassis="leaf2" destPort="Eth2/1"/>
      <LINK_INFO sourcePort="Eth2/3" destChassis="leaf3" destPort="Eth2/1"/>
    </CHASSIS_INFO>

    <CHASSIS_INFO sourceChassis="spine2" type="n7k">
      <LINK_INFO sourcePort="Eth1/1" destChassis="leaf1" destPort="Eth1/2"/>
      <LINK_INFO sourcePort="Eth1/2" destChassis="leaf2" destPort="Eth1/2"/>
      <LINK_INFO sourcePort="Eth1/3" destChassis="leaf3" destPort="Eth1/2"/>
    </CHASSIS_INFO>
  </DATA_CENTER>
</CISCO_NETWORK_TYPES>
```

Following is the breakdown of each line and the XML tags and attributes:

```
<?xml version="1.0" encoding="UTF-8"?>
<CISCO_NETWORK_TYPES version="1.0" xmlns="http://www.cisco.com/cableplan/Schema2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cisco.com/cableplan/Schema2 nxos-cable-plan-schema.xsd ">
```

The previously mentioned example headings are required for XML processing and Cisco specific headers required to denote it as a Cisco cable-plan. These must be exact as mentioned in the previous example for all cable-plans. Failure to adhere to the specified format will result in a rejected cable-plan.

- `CISCO_NETWORK_TYPES` - Parent tag for the entire XML cable-plan. The entire cable-plan must be within this tag.

```
<DATA_CENTER networkLocation="san-jose" idFormat="hostname">
```

- `DATA_CENTER` - Houses all the information for each chassis in the plan.
- `networkLocation` - States where the data center is for clerical purposes. No affect on miscabling.
- `idFormat` - States the format in which IDs will be presented in subsequent entries. As of 12/2/2014, the only supported format is "hostname". All cable-plans that do not use “hostname” as the ID format will be rejected.

```
<CHASSIS_INFO sourceChassis="spine1" type="n7k">
```

- `CHASSIS_INFO` - Describes a single chassis. All interfaces that belong to sourceChassis that administrators want to have checked by cable-plan must be within one, and only one of these tags.
- `sourceChassis` - The chassis that all subsequent interfaces (described in LINK_INFO tags) belongs to. In this case, all interfaces within the CHASSIS_INFO tags are said to belong to the chassis called "spine1".
- `type` - The type of chassis. Only the Cisco Nexus Switches are supported, so this attribute must read "n#k", such as "n7k" (non-case sensitive). All cable-plans that do not adhere to the "n#k" format will be rejected.

```
<LINK_INFO sourcePort="Eth2/1" destChassis="leaf1" destPort="Eth2/1"/>
<LINK_INFO sourcePort="Eth2/2" destChassis="leaf2" destPort="Eth2/1"/>
<LINK_INFO sourcePort="Eth2/3" destChassis="leaf3" destPort="Eth2/1"/>
```

- `LINK_INFO` - Fully describes an interface connection from sourceChassis (as mentioned previously) to destChassis. In this case, we are stating that spine1's port eth2/1 is connected to leaf1's eth2/1, spine1's eth2/2 is connected to leaf2's eth2/1, and so on.

- `sourcePort` - The sourcePort is the port on the sourceChassis (as mentioned previously). Source ports should be unique per chassis (that is, spine1 should not describe multiple connections coming from port eth2/1). While an import will not fail on cable-plans with non-unique ports, a warning will be printed and only the first entry will be read and checked by cable-plan.
- `destChassis` - The destination chassis that the sourceChassis is connected to.
- `destPort` - The port on the destination chassis that has the connection. Like sourcePort, this should be unique to destChassis.

```
</CHASSIS_INFO>

<CHASSIS_INFO sourceChassis="spine2" type="n7k">
<LINK_INFO sourcePort="Eth1/1" destChassis="leaf1" destPort="Eth1/2"/>
<LINK_INFO sourcePort="Eth1/2" destChassis="leaf2" destPort="Eth1/2"/>
<LINK_INFO sourcePort="Eth1/3" destChassis="leaf3" destPort="Eth1/2"/>
</CHASSIS_INFO>
```

Here you close the CHASSIS_INFO tag and we completely describe the connections on all interfaces that spine1 can have. You can open another tag to describe interfaces in the same way, on a different router that you want it checked. There is no limit to the number of unique CHASSIS_INFO tags you can describe in a cable-plan.

```
</DATA_CENTER>
</CISCO_NETWORK_TYPES>
```

Finish closing the tags to complete our XML file. At this point we have a valid XML cable-plan that completely describes a data center containing the following switches: spine1, spine2, leaf1, leaf2, leaf3.

Note that you need not repeat connections from each chassis' perspective. Since you describe that sourceChassis spine2 is connected to destChassis leaf1 via Eth1/1 - Eth1/2, you do not need an entry describing sourceChassis leaf1 and destChassis spine2. While the plan will not fail if you include this redundant information, it will be ignored and add unnecessary length to the plan.

Also ensure that other interfaces may be present that were not described in the cable-plan. Consider a third spine, spine3, connected to all the leaf nodes. Since this is not described in the cable-plan, LLDP TLVs with this information will not be checked and only an indicator warning administrators of its absence from the cable-plan will be noted. As long as the un-described spine is not interfering with ports already described in the cable-plan, then no actions will occur with the missing ports (that is, the new spine3 cannot be connected to leaf1's eth1/2, since you explicitly state that leaf1's eth1/2 is connected to spine2's eth1/1).

Cable-Plan Specific Commands

The cable-plan exec commands are under the **fabric connectivity cable-plan** commands. For example, to import the valid XML cable-plan we can use this CLI:

```
leaf1# fabric connectivity cable-plan import [means]:[location] {update} {verbose}
```

- means - Describes the way you may want to import. Currently the only local means for importing is via bootflash and USB. If you would like to import via remote location, FTP, SCP, SFTP and TFTP protocols are supported for this option.
- location - The location of the file. For instance "`scp://calinfor@171.77.77.7/nobackup/test.xml`" will scp the file located at /nobackup/test.xml on 171.77.77.7.
- update - (Optional) To add the entries in the importing cable-plan to the cable-plan entries that have already been imported in a previous import command. If there is no cable-plan already imported then

this keyword has no effect. If this keyword is not specified during the import command, then the previously imported cable-plan (if there is one) will be deleted and it will be replaced with the new cable-plan specified in the import command.

- verbose - (Optional) To print all errors associated with the import. If this option is not specified, only a one line description of success or failure will be printed. If the option is specified, detailed information on why the import failed will be printed to the console.

```
leaf1# fabric connectivity cable-plan export[cable-plan location]
```

The export CLI will take a previously imported cable-plan and write it to a file. Since cable-plans are stored in memory after they are imported, users may have lost or altered the previous XML file containing the cable-plan. This allows these users to export the current cable-plan stored in memory and regain their XML file.

```
leaf1# fabric connectivity cable-plan generate [cable-plan location] [name]
```

Auto generate a cable-plan based on the current LLDP neighbors. This creates a valid cable-plan based on the local switch's perspective. Outputs a time-stamped file in the bootflash when done. Useful for allowing users to quickly create a cable-plan out of current topology without writing it by hand. Users can then either import this cable-plan to enforce the current topology or users can use it as a template of sorts to create a more complex cable-plan. Note, if there are no LLDP neighbors present then only a template that will need to be edited will be created.

Now turn on cable-checks. If this is disabled (by default) then all TLVs received on this switch will not be checked against the cable-plan, regardless of whether or not a plan is imported:

```
leaf1# configure terminal
leaf1(config)# [no] fabric connectivity cable-plan enforce
leaf1(config)# exit
```

To check information regarding the new cable-plan:

```
leaf1# show fabric connectivity cable-plan
```

This command prints the information in the currently stored cable-plan. If no information is stored this will be blank. This will print the local and remote Chassis and Port IDs, and the cabling status of this connection. This is just a printing of the cable-plan and the status of each link.

Switch Links Cable State

The switch learns and stores all the neighbors and their respective ports and states. The following command can be used to display this information.

```
spine1# show fabric connectivity neighbors {errors | interface | tier}
```

- errors - (Optional) Displays only interfaces with errors
- interface - (Optional) Displays information pertaining to the specified interface
- tier - (Optional) Displays only interfaces with the specified tier level
- default (blank) - Displays all interfaces that have received an LLDP TLV since either tier or cable-plan checks have been toggled on. Unlike "show fabric connectivity cable-plan", the show neighbor command will show the actual data being received by the switch via LLDP TLVs

Local Chassis and Port IDs, remote Chassis and Port IDs, the tier levels of the remote Chassis, the expected cable-plan entry and all the status will be displayed.

Status codes include the following:

- Ok - Everything working as intended, the check succeeded
- Unkn - Unable to determine the status, most likely because cable-check and tier-check has been disabled
- ErrC - The port has been err-disabled due to a mismatch (that is the TLV received did not match the specific entry in the cable-plan)
- ErrT - A tier level mismatch error
- S - May be specified at the end of a status. This means the port is "stale" in that the switch has received a purge event from the remote peer. This usually happens when the port has gone err-disabled or when we have not seen LLDP events from this link in a long time.

Examples

The following is a sample output of fabric connectivity:

```
switch# show fabric connectivity neighbors

-------------------------------------------------------------------------------
Local System:
Device Tier Config:          Disabled      Device Tier Level:          Unknown
Mismatch Delay Config:       Disabled      Mismatch Delay Timeout:           0
Cable-Plan Enforce:          Enabled
DeviceID:  switch                          ChassisID:  f866.f2d6.37c4
-------------------------------------------------------------------------------
         Codes: (Ok) Normal, (ErrT) Tier error , (ErrC) Cable-Plan error,
                (V) VPC Peer connection, (S) Stale entry, (Unkn) Unknown,
                (Enp) Entry not present in Cable-Plan, (Tl) Tier level
Neighbor Table:
-------------------------------------------------------------------------------
Local      DeviceID              PortID     Tl   Cable-Plan             Status
Intf                                             Entry

Eth1/13    switch                Eth1/14    Unkn switch2,Eth1/14        ErrC,S
Eth1/14    switch                Eth1/13    Unkn switch9,Eth1/13        ErrC,S
```

When cable-plan is enabled, the cable-plan column will show the plan based validation status (Ok, Enp - Entry Not Present, or as above the expected value for errored links).

```
Spine11# show interface Eth2/1

Ethernet2/1 is down (errDisabled)
……
……

switch# show interface status err-disabled

----------------------------------------------------------------------------
Port         Name               Status             Reason
----------------------------------------------------------------------------
Eth2/1        --                Miscabled          Miscabled
```

Miscabling Error Disable Action Control

By default any tier based or cable-plan based mismatches trigger err disabling of the ports. After the miscabling or topological errors are corrected, the miscabled ports can be brought back live either by manually removing them out of error disabled state or automatically through a configuration CLI. The error disabling action can also be turned off or delayed. The following knobs can control these optional behaviors.

- To remove the default errdisable on error action:

  ```
  Switch(config)# no errdisable detect cause miscabled
  ```

- Miscabled error disabling action can be delayed for specified time (timeout) using the following configurations. Here action will be taken after <timeout> seconds have passed:

  ```
  Switch(config)# fabric connectivity mismatch action delay <timeout>
  ```

- Auto recovery of the miscabled ports out of error disabled state can be enabled using the following command. The below command will attempt to recover all miscabled ports at some set interval.

  ```
  Switch(config)# errdisable recovery cause miscabled
  ```

- To change the interval at which the errdisable recovery command attempt to recovery, the following configure command can be used:

  ```
  Switch(config)# errdisable recovery interval <time>
  ```

- To clear a single entry or all entries from the Clos neighbor cache immediately, the following clear command can be used:

  ```
  Switch# clear fabric connectivity neighbors [interface | stale]
  ```

  This CLI will allow the you to clear a single entry or all error entries or all entries from the Clos neighbor cache immediately. You should manually clear an already secured port in the neighbor cache if recabling is desired to remove old stale entry immediately. The entry will be automatically removed after the hold time otherwise. If you have multiple errors that you just fixed, performing a 'clear neighbors' on the affected switches is the easiest way to bring up all interfaces again if 'error recovery' is not enabled. Also, if you have a switch that was previously in the network, but has gone inactive (may be you removed it or it was taken down), performing a clear command is the only way to completely remove it from the neighbor cache (so it stops showing up in the **show neighbors** command mentioned above).

- To clear the cable-plan from this switch, the following command can be used:

  ```
  Switch# clear fabric connectivity cable-plan
  ```

  Clears the current cable-plan. If you had previously written a cable-plan to the startup config and you want this clearing to persist, use the **copy running-config startup-config** command.

CHAPTER 6

XMPP Client Configuration

Feature Information for XMPP

Table 3: Feature Information for XMPP

Feature	Releases	Feature Information
SPOM	7.2(0)D1(1)	Included as a part of the chapter *XMPP Client Configuration*. Single point of management support.

Overview of XMPP Client

Extensible Messaging and Presence Protocol (XMPP) is a communication protocol. XMPP clients set up TCP based XMPP connection to XMPP server. XMPP server forwards the messages from one client to another client or a group of clients based on the configuration and request.

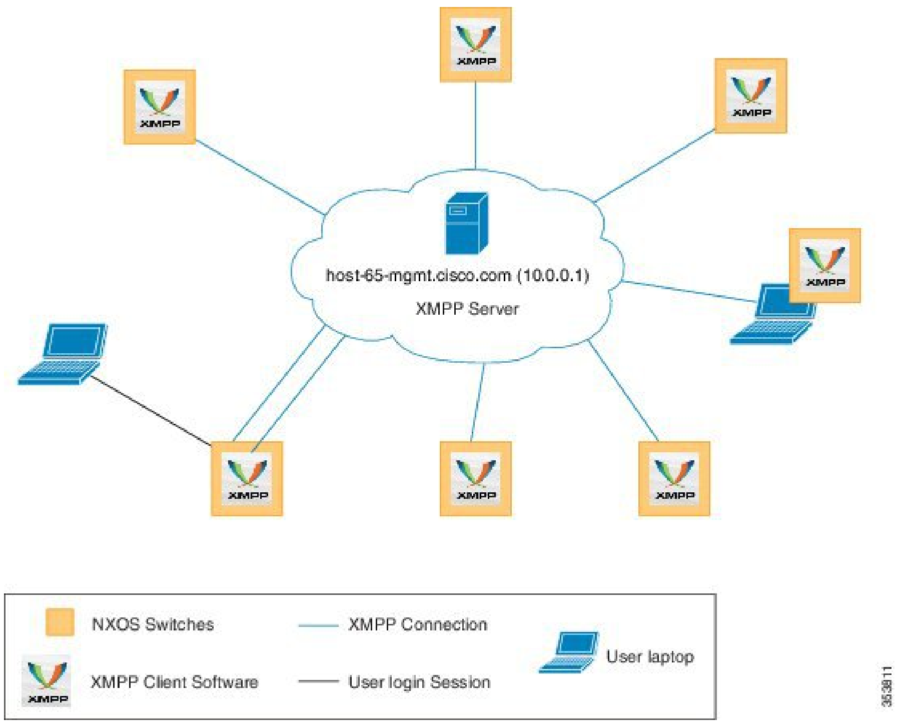
This XMPP protocol is adopted by DFA, so the administrator can manage (by issuing CLI commands) a device or group of devices in the network from the administrator's XMPP connection with a single point of management with no separate login required for each device. Each device is a XMPP client that can be configured to connect to XMPP server. The administrator issues the CLI command and the device receives the CLI commands. Device processes the CLI commands and sends CLI output back to the administrator XMPP client.

XMPP client support is added to the Cisco NX-OS operating system with DFA since 7.0(0)N1(1) for Cisco Nexus 5000/6000 Series Switches and from 7.2(0)D1(1) for Cisco Nexus 7000 Series Switches. The XMPP client library can be shared by different features which require XMPP client functionality. This chapter explains the usage of XMPP for managing Cisco NX-OS supported devices with the XMPP bus. The XMPP client on the switch provides a single point of access from any switch to the rest of the switches in the network. You can also use other XMPP clients like Pidgin.

Cisco Nexus Series switch command line shell (VSH) includes an integrated XMPP client. You can utilize this feature to send CLI commands to a single device or a group of devices through the XMPP bus.

The figure below illustrates switches in a network being managed by a user using the XMPP protocol. The user can either telnet or SSH to a Cisco NX-OS switch or run a third-party instant messenger such as Pidgin to manage other switches.

Figure 4: XMPP Client



XMPP Server

An XMPP server is required to establish the XMPP communication with the clients. DFA only supports the XMPP server bundled with the Cisco Prime Data Center Network Manager (DCNM) Open Virtual Appliance (OVA). For more information on XMPP server configuration, see Cisco DCNM OVA Installation Guide, release 7.x.

The XMPP server bundled with Cisco Prime DCNM OVA is XCP (Cisco internally developed product) with Version 2012.2.0.38425. Standalone XMPP server uses PostGreSQL database that is packaged in the same OVA. In a XMPP HA setup, an external Oracle database is required to be installed by user for both XMPP active and standby servers to share. Oracle 11g XE version 11.2.0 (oracle-xe-11.2.0-1.0.x86_64.rpm) is verified to work with XCP Version 2012.2.0.38425.

XMPP server is bundled with DCNM OVA. Log into Cisco Prime DCNM server via SSH and use the **appmgr** command to verify the XMPP status:

appmgr [start|stop|status|restart] xmpp

```
appmgr start xmpp
```

For XMPP user and group management use the following **appmgr** command to add, delete, and list the users and groups:

appmgr [add_user|delete_user|list_users|add_group|delete_group|list_groups] xmpp

```
appmgr add_user xmpp -u <user> -p <secret>
appmgr delete_user xmpp -u <user>
appmgr list_users xmpp
appmgr add_group xmpp -u <user> -p <secret> -g <groupname>
appmgr delete_group xmpp -u <user> -p <secret> -g <groupname>
appmgr list_groups xmpp
```

XMPP Client Configuration

The basic configurations required on an Cisco NX-OS switch to establish an XMPP session are the following:

- Enabling the XMPP Client (fabric access) feature
- Network path to the XMPP server
- XMPP server configuration to establish client-server connection
- XMPP group subscription

Enabling the XMPP Client Feature

The XMPP client feature on an Cisco NX-OS switch is a conditional service which is referred as **fabric access**. This feature can be enabled/disabled using the following configuration command:

```
[no] feature fabric access
```

Network Path to the XMPP server

The address information for an XMPP server is a Fully Qualified Name (FQN). Therefore, the switch requires either the static local IP-to-Name mapping or the capabilities to resolve the FQN via DNS to establish a client-server connection.

The following is an example for configuration of a locally configured DNS resolution for server host-65-mgmt.cisco.com with IP address 10.0.0.1.

```
switch(config)# ip host-65-mgmt.cisco.com 10.0.0.1
```

Note This name must match with the name of the XMPP server. In case of Cisco Prime DCNM XMPP server, the name must match with the printout of the **appmgr list_users xmpp** command.

XMPP Server Connection Configuration

XMPP client connects to a specific XMPP server with configured VRF route and JID information.

JID is a Jabber Identifier (JID), which uniquely identify individual entities in the XMPP (Jabber) network. For example, consider if the Fully Qualified Domain name of server is xmpp-server.cisco.com and the username is 'admin' (username can be obtained by issue 'where' cli on the device), JID for this user is admin@xmpp-server.cisco.com. For a device, by default device hostname is used as the ID to construct device JID. If device host name is leaf, JID for this device is leaf@xmpp-server.cisco.com.

Use the following command:

```
[no]fabric access server <fqn-name> [vrf <vrf-known-name>] password <password>

switch(config)# fabric access server  host-65-mgmt.cisco.com vrf management password
    pwd
```

For more information on XMPP commands, see XMPP Client Commands, on page 42.

XMPP Group Subscription

XMPP client can join multiple group chats. The user has to create a group by using the fabric access create group *group-name* command. Before running the command to create a group, the user has to log into the XMPP server and get an authentication.

Note If a group does not exist yet, the group subscription command will not be in effect, and the switch has to retry the group subscription periodically till it is successful.

The XMPP client can join a group using the **[no] fabric access group** <*group-name*> command.

```
switch(config)# fabric access group leaves
```

XMPP Client Configuration for Third-Party Instant Messenger

The basic configurations (high level) required on a third-party instant messenger to establish an XMPP session are the following:

Note The detailed procedure depends on the type of instant messenger application used, for example Pidgin. For more information, see https://www.pidgin.im/support/.

- An account with the XMPP server (if open registration is allowed; else the system administrator can set up an account on the XMPP server.)
- Establish connection with the XMPP server
- Establish connection with the Cisco NX-OS switches
- XMPP group subscription

Logging into an Account with the XMPP Server

The user has to log in to the XMPP server and get an authentication first to access the XMPP client on the Cisco NX-OS switch. The XMPP login username is inherited from the current login/telnet session username on the switch.

Note Whenever a user role or privilege of a user account is changed, the changed role shall come into effect for subsequent logins only.

```
Switch# fabric access login <user-password>
```

Note After successful authentication from the server, the user can use the XMPP client on the switch to perform XMPP client functionalities similar to those on the instant messenger.

Verifying the Network Path to the XMPP Server

The network path (through VRF) can be validated with a ping to ensure that the XMPP client on the Cisco NX-OS switch communicates with the XMPP server.

```
ping host-65-mgmt.cisco.com vrf management
```

Verifying the XMPP Server Connection

The following **show** command can be used to check if the XMPP client on the switch has been successfully authenticated to the server:

```
Switch# show fabric access connection

XMPP Ping:
  Status = Enabled
  Interval = 120 second(s)
  Response = 60 second(s)
  Retry = 3 time(s)
  Next Ping will be sent : in 24 second(s)
XMPP Payload CDATA-Encapsulated : Enabled
Device Connection :
  JID      = switch@host-65-mgmt.cisco.com/(fmgr-device)(FOX01010101)
  State    = AUTHENTICATED
```

Examples for XMPP Client Configuration

This is an example for basic configuration required on a Cisco NX-OS switch to enable the XMPP client and to establish the XMPP connection with the server.

```
feature fabric access
!Enables the XMPP Client feature

ip host host-2-mgmt.cisco.com 10.1.1.2
!DNS for XMPP server

fabric access server host-2-mgmt.cisco.com vrf management password pwd
!Establishes XMPP connection between the switch and the XMPP-server,
configures XMPP-server DNS name, and switch password.
```

This is an example for basic configuration required for an XMPP client to join multiple groups.

```
fabric access group global
!XMPP client joins group chat “global”
```

```
fabric access group leaves
!XMPP client joins group chat “leaves”
```

The following is a simple example of getting into XMPP client mode from Cisco NX-OS VSH CLI exec mode. Some prerequisite steps are needed to reach this stage, for example, setting up the XMPP server, have the Cisco Nexus Switch establish XMPP connection to the server. This specific example starts with attaching to the group "leaves" and at the same time entering the XMPP Client mode. Any commands entered in the client mode will be applied to all members of the "leaves" group. The **exit** command leaves the XMPP client mode and gets back to Cisco NX-OS VSH CLI.

```
Switch#
Switch# fabric access attach group leaves
Switch>leaves# copy running-config startup-config //will be applied to all members//
Switch>leaves# exit
Switch#
```

NETCONF contents in XMPP bus

Since 7.0 release, XMPP Client can process NETCONF XML. There is no specific "tag" introduced for NETCONF XML processing in the XMPP Client. XMPP Client has the capability to distinguish the incoming XMPP payload as XML/Netconf or raw CLI without a specific tag.

XMPP Client will use the following information to distinguish the XML/Netconf commands or raw CLI commands.

Opening xml tag "`<?xml …?>`"

Attribute "`urn:ietf:params:xml:ns:netconf:base:1.0`"

Closing `char string "]]>]]>"`

```
<?xml version="1.0"?>
<nc:hello xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nc:capabilities>
    <nc:capability>
        urn:ietf:params:xml:ns:netconf:base:1.0
    </nc:capability>
  </nc:capabilities>
</nc:hello>]]>]]>
```

XMPP Client Commands

Cisco NX-OS commands on XMPP Client

```
Switch(config)#[no] feature fabric access
```

- Enables/disables the fabric access feature.

```
Switch(config)#[no] fabric access server <dns-name> [device <name>] [vrf <vrf-name>] password
 <password>
```

- Configures XMPP server (must be DNS name) for the XMPP connections. This is for the switch to establish its own XMPP connection to the server.

- If no device <name> is configured, the default value will be the hostname of the device. The name can be upper/lower case in configuration, but it will be converted to lower case in the server always based on the standard.
- Password allows authentication of XMPP device connection. It can be input by clear or encrypted format. And also can be interactive input if no password is specified in the command line. In running-config or startup-config, it will only show encrypted password.

```
Switch(config)#[no] fabric access group <group-name>
```

- The group-name can be configured in uppercase or lowercase, but it will be always converted to lowercase in the server based on the standard.
- If the group does not exist in the server, the device will not be able to subscribe to the group and will retry periodically in the background. Device can only join the groups, not create the groups.

```
Switch(config)#[no] fabric access prepend-id
```

- To have the display of the replied result prepend the source device information for each line. The default is disabled.

```
Switch(config)#[no] fabric access ping [ interval <interval> | response <response> | retry
 <retry>]
```

- To have the device send XMPP PING packets periodically to the configured XMPP server. The default is enabled.
- Proactively validate the liveness of the server.
- Interval: how frequently send the XMPP PING packet.
- Response: how long will to wait for the response XMPP PING packet from server.
- Retry: how many times for retrying.
- Default values for those parameters and range can be found in the CLI help.

```
Switch(config)#[no] fabric access CDATA-encap
```

- To have the replied result encapsulated inside <CDATA> tag in the XMPP payload. The default is enabled.
- CDATA is used about text data that should not be parsed by the XML/XMPP parser. Characters such as "<" and "&" are illegal in XML elements.
- Different XMPP server may support parsing <CDATA> differently.

```
Switch#[no] fabric access create group <group-name>
```

- For the login user to create XMPP chat groups in the XMPP fabric access network.
- Creation will be failed if no successful user connection.

```
Switch#[no] fabric access login [<password>]
```

- The login user can connect to the configured XMPP server and build up the user connection.

- The login name can be in upper/lower case, but it will be converted to lower case in the server always based on the standard. Login name is generated automatically for this cli and sent, which is the same as the current user credentials logged into the privileged EXEC mode.
- Hence, it will be lower case in displaying runtime information.
- Password allows the authentication of XMPP for the user connection. It can be input by clear or encrypted format. And also can be interactive input if no password specified in the command line.
- Only when the login user creates the user connection to the XMPP server, the login user can access the fabric access (XMPP) network.

```
Switch#[no] fabric access local-help
```

- In the attach mode, the help command is executed in local device or remote device. Default is local.

```
Switch#[no] fabric access attach { group <group-name> | device}
```

- The login user can attach (join) XMPP chat group or device under attaching mode in the XMPP fabric access network.
- Attaching will be failed if there is no successful user connection.
- If the group or device do not exist in the XMPP server, the attachment will fail.

```
Switch# fabric access send { group <group-name> | device <device-name>+} <cli-command>
```

- The login user to send CLIs to group/device(s) in the XMPP fabric access network.
- Sending will fail if there is no successful user connection.
- If the group or device(s) does not exist in the XMPP server, the sending will fail.

```
Switch# show fabric access connections
```

- To show fabric access feature information and all the connections' status.

```
Switch# show fabric access group [{members <group-name>} | {device} | {user}]
```

- To show fabric access group information.
- Command will fail if there is no successful user connection.
- It can show all the created groups in the XMPP fabric access network, groups which the device has joined, groups which the login user has joined, and the group members of the specific group.

```
Switch# show fabric access statistics
```

- To show fabric access feature statistics information.
- The statistics information about which device did not reply before.

```
Switch# clear fabric access user <username>
```

- To clear a user connection in this device.
- Remove any specified user connection in this device in the XMPP fabric access network.

```
Switch# clear fabric access statistics
```

- To clear fabric access feature statistics information.
- The statistics information can be reset by this command.

CHAPTER 7

Multi-tenancy

Feature Information for Multi-tenancy

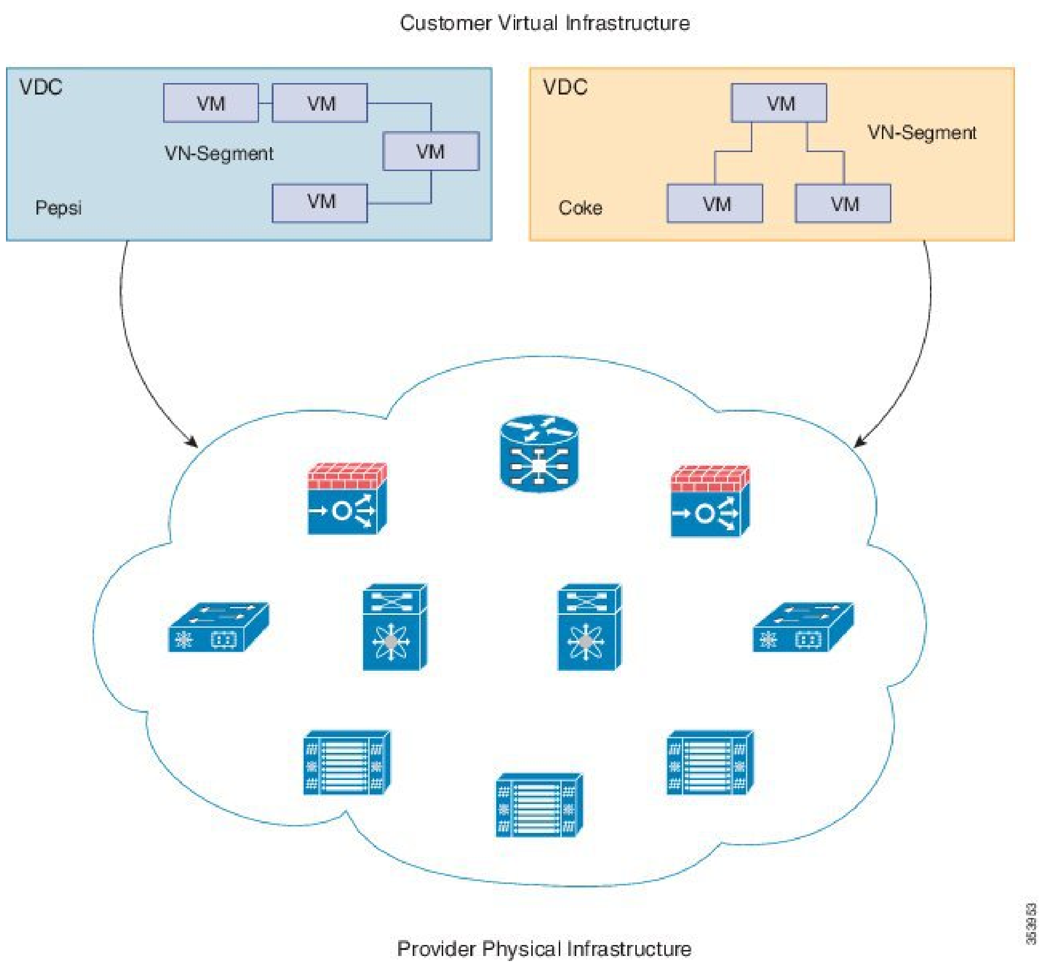
Table 4: Feature Information for Multi-tenancy

Feature	Releases	Feature Information
Multi-tenancy	7.2(0)D1(1)	Included a new chapter on *Multi-tenancy* . Multi-tenant data center handles the traffic segregation between different tenants.
Segment ID	7.2(0)D1(1)	Included a new section on *VN-Segment*. VN-Segment network can support up to 16 million virtual network segments.

Multi-tenancy

Multi-tenancy is a concept that refers to the logical isolation of shared virtual compute, storage, and network resources. In multi-tenant data center, tenants subscribe to virtual data center (VDC), and based on the services hosted by the tenants I within the virtual data center, each virtual data center can have multiple VN-Segments.

Figure 5: Multi-tenant Data Center


Customer Virtual Infrastructure
VDC
VM
VM
VN-Segment
VM
Pepsi
VM
VDC
VM
VN-Segment
Coke
VM
VM
Provider Physical Infrastructure
353953

The above figure depicts two virtual data centers assigned to different tenants. For example Coke and Pepsi, each virtual data center has virtual data center elements like virtual machines (VM), storage inter-connected by a VN-Segment.

Multi-tenant data center handles the traffic segregation between different tenants, and also within tenant traffic, for security and privacy. Data centers have deployed VLANs to isolate the machines of different tenants on a single Layer-2 network. This could be extended to the virtualized data centers by having the hypervisor encapsulate VM packets with a VLAN tag corresponding to the VM owners. This approach provides a Layer-2 abstraction to the tenants and, with VRF, it can completely virtualize the Layer-2 and Layer-3 address spaces. However, the VLAN is a 12-bit field in the VLAN header, limiting this to at most 4K tenants. Also, multi-tenant network should provide tenants with simple and flexible network abstractions, by completely and efficiently virtualizing the address space at both Layer-2 and Layer-3 for each tenant, without any restrictions on the tenant's choice of Layer-2 or Layer-3 addresses. Also, tenants might want to extend their IT services or storage network which uses non-IP protocols such as Fibre Channel over Ethernet (FCOE). These protocols may be important for tenants trying to move the existing applications into service provider data center (SPDC) and does not support in a network that has no Layer-2 abstraction. Similarly, these tenants will benefit from the SPDC that supports tenant-level broadcast or multicast trees. In order to maximize the benefits of resource sharing, which provides multiplexing to achieve better resource efficiency and cost saving, multi-tenant data centers must scale to larger size to accommodate more tenants and VMs. Maintaining such large multi-tenant data centers can be expensive and hence multi-tenant data centers require automated configuration and management tools to reduce the cost. Also with the large scale Layer-2 multi-tenant data center needs high bi-sectional bandwidth and this can be achieved by using Layer-2 multi-pathing short path bridging technologies like FabricPath and TRILL, which also addresses the MAC address scale issues required for per-tenant Layer-2 abstraction.

Another important requirement for multi-tenant data center is to support the mobility of VMs within and across SPDC, and also into enterprise data centers. Mobility within SPDC allows for dynamic tenant growth and maximizes resource utilization and sharing. For instance, if a tenant needs to add a VM to the existing SPDC POD but all the servers are overloaded then the VM for the tenant can be accommodated on another SPDC POD, which has the capacity and is available in server. This means that the VN-Segment must be able to extend virtually anywhere within and across multi-tenant data center.

Bridge-Domain

Note This section is applicable only for multi-tenancy full version.

A bridge-domain is a generic object that represents a Layer-2 broadcast domain on a device. Either a VLAN or a bridge-domain with the same number can exist. The bridge-domain range needs to be carved out from the 4096 VLAN range. The reserved VLANs cannot be used as a bridge-domain. All the carved out bridge-domain can be used as user/tenant bridge-domain.

The following is an example to carve out the bridge-domain range:

```
system bridge-domain 10-3000
```

Given above is the entire set of bridge-domains that can be used on the switch. For bridge-domain to be used for different VRFs you need to define a fabric bridge-domain range. Out of this range of user bridge-domains, a subset of bridge-domains can be designated as fabric bridge-domains. The corresponding BDIs will be reserved as fabric BDIs.

The following example shows allocating fabric bridge-domains:

```
system fabric bridge-domain 2001-3000
```

This will designate bridge-domains 2001-3000 to be used as fabric bridge-domains. Fabric bridge-domains are used as part of applying the vrf-tenant-profile. The remaining bridge-domains (10-2000) are user bridge-domains. They will be used to map tenant VNIs on the switch.

Note Do not create, delete, or edit a bridge domain in the fabric bridge domain range. These are created whenever a new VRF is created and is removed when the VRF is removed.

A fabric-control bridge-domain is configured from the range of user bridge-domains only (in this case 10-2000). The fabric control bridge-domain/VLAN needs to be defined for control traffic to propagate. There can only be one fabric control bridge-domain or a VLAN in the system.

Note Use of VLAN 1 as fabric control is not allowed.

Configuring Bridge-Domain

SUMMARY STEPS

1. **configure terminal**
2. **[no] system bridge-domain { bd-list | add bd-list | all | except bd-list | none | remove bd-list }**
3. **[no] system fabric bridge-domain { bd-list | add bd-list | all | except bd-list | none | remove bd-list }**
4. **[no] bridge-domain {bd-id | bd-range}**
5. **[no] fabric-control**
6. **show bridge-domain summary**
7. **show bridge-domain id**
8. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters configuration mode.

Step 2	**[no] system bridge-domain { bd-list | add bd-list | all | except bd-list | none | remove bd-list }**
	Example:
	`switch(config)# system bridge-domain add 100-200`
		Identifies the IDs that are available for bridge-domain configurations.
		• The valid range for the ID argument is from 2 to 3967.
		• (Optional) The id keyword and argument combination identifies the last ID in a range of contiguous IDs. The hyphen (-) is mandatory.
		• (Optional) The arguments like add, remove, all, except, none can be used for adding, removing, adding all, adding all except and removing all respectively.

Step 3	**[no] system fabric bridge-domain { bd-list | add bd-list | all | except bd-list | none | remove bd-list }**
	Example:
	`switch(config)# system fabric bridge-domain 151-200`
		Identifies the IDs that are available for fabric bridge-domain configuration. This command has same option as the previous command but the range it can act on is only the existing system bridge-domain carved out range.

Step 4	**[no] bridge-domain {bd-id | bd-range}**
	Example:
	`switch(config)# bridge-domain 100-110`
	`switch(config-bdomain)#`
		Enters bridge-domain configuration mode and configures a bridge-domain. The domain-ID argument is a unique identifier for the bridge-domain and underlying VLAN to be created. The valid range is defined by the system bridge-domain configuration.
		Note You can use the no form of this command to remove the bridge-domain configuration including port associations. Removing the bridge-domain configuration does remove the underlying VLAN and all the bridge-domain properties.

	Command or Action	Purpose

Step 5	**[no] fabric-control**
	Example:
	`switch(config)# bridge-domain 100`
	`switch(config-bdomain)# fabric-control`
		Make the bridge-domain as the fabric control bridge-domain. Only one bridge-domain or a VLAN can be configured as fabric control.

Step 6	**show bridge-domain summary** **Example:** `switch# show bridge-domain summary`	(Optional) To show the bridge-domain configuration. Similar to *show vlan summary*.
Step 7	**show bridge-domain id** **Example:** `switch# show bridge-domain 100`	(Optional) To show whether the bridge-domain is created or not. Also to show any bridge-domain property configured under it.
Step 8	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Example

The following example shows how to create a bridge-domain:

```
switch# configure terminal
switch(config)# system bridge-domain 100-200
switch(config)# bridge-domain 100
switch(config-bdomain)# name Cisco:tenant1
switch(config-bdomain)# no shutdown
switch(config-bdomain)# exit
switch(config)#
switch(config)# bridge-domain 101
switch(config-bdomain)# fabric-control
switch(config-bdomain)# name fabric-control_BD
switch(config-bdomain)# no shutdown
switch(config-bdomain)# exit
```

VN-Segment

VN-Segment network can support up to 16 million virtual network segments (also called Virtual Network Identifiers) and VN-Segment has global significance in Layer-2 network. In multi-tenant applications, tenant traffic can still be received as “Dot1Q” tagged that need to be classified to the VN-Segment assigned to those tenants. VN-Segment is the extension of VLANs – both need to coexist. VLAN range is from 1-4095 and VN-Segment (VNI) range is from 4096-16 Million.

Note For release 7.2(0)N1(1), to modify the VN-Segment of a VLAN, you must delete any existing VN-Segment mapping to add the new VN-Segment mapping.

Configuring VN-Segment

SUMMARY STEPS

1. **configure terminal**
2. **feature vni**
3. **vni <vni range>**
4. **shutdown/no shutdown vni**
5. **member vni <vni-range>**
6. **encapsulation profile vni <profile-name>**
7. **service instance vni**
8. **shutdown/no shutdown vsi**
9. **encapsulation profile vsi**
10. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters configuration mode.
Step 2	**feature vni** **Example:** `switch(config)# feature vni`	Enables feature VNI or Segmentation.
Step 3	**vni <vni range>** **Example:** `switch(config)# vni 5000-5002, 5005`	Creates a range of VNIs.

Step 4	**shutdown/no shutdown vni**
	Example:
	`Switch(config)# vni 5000-5001`
	`switch(config-vni)# [no] shutdown`
		Shuts down a range of VNIs.

Step 5	**member vni <vni-range>**
	Example:
	`switch(config)# bridge-domain 10-12`
	`switch(config-bdomain)# [no] member vni 5000-5002`
		Configures VNIs as members under a range of bridge-domains.

	Command or Action	Purpose

Step 6	**encapsulation profile vni <profile-name>**
	Example:
	`switch(config)# [no] encapsulation profile vni cisco`
	`switch(config-vni-encap-prof)# [no] dot1q 20 vni 5000`
		Creates an encapsulation profile named *cisco* with dot1q 20 mapped to vni 5000.

Step 7	**service instance vni**
	Example:
	`switch(config)# interface ethernet 3/1`
	`switch(config-if)# service instance 1 vni`
	
	`switch(config)# interface ethernet 3/2`
	`switch(config-if)# service instance vni default`
		Creates a numbered VSI under parent port interface Ethernet 3/1 and 3/2.

Step 8	**shutdown/no shutdown vsi**
	Example:
	`switch(config-if)# service instance 1 vni`
	`switch(config-if-srv)# [no] shut`
		Shuts a numbered VSI.

Step 9	**encapsulation profile vsi**
	Example:
	`switch(config-if)# service instance 1 vni`
	`switch(config-if-srv-def)# encapsulation profile cisco default`
		Applies the encapsulation profile to a VSI.

Step 10	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Detailed Steps

The 'Feature vni' or segmentation can be enabled only when the virtual device context has been limited to F3.

```
switch(config)# feature vni
Feature vni requires F3 or newer linecards
switch(config)# vdc switch
switch(config-vdc)# limit-resource module-type f3
This will cause all ports of unallowed types to be removed from this vdc. Continue (y/n)?
[yes] yes
switch(config-vdc)# feature vni
```

There has to be 1:1 mapping between VNI and bridge-domain. VNI has global significance in the Layer-2 network while bridge-domains remain local to the virtual data center (switch). Bridge-domains would have VNIs as members.

Commands to create a VNI and adding the VNI under a bridge-domain.

```
switch(config)# [no] vni 5000-5002
switch(config-vni)# [no] shutdown
       switch(config)# bridge-domain 50-52
       switch(config-bdomain)# [no] member vni 5000-5002
```

Existing legacy IEEE 802.1Q switches and End-host/Servers, capable of sending dot1q tagged traffic, should be able to connect to VN-Segment supported network. This capability is provided by **VN-Segment Service Instance (VSI)**. VN-Segment Service Instance Ports on the VN-Segment capable switch allows to map the dot1q tagged frames received on that port uniquely to a VN-Segment (VNI).

An encapsulation profile like a template needs to be created to define the dot1q to VNI mappings.

Command to create an encapsulation profile template named ***cisco*** and add/delete a dot1q to VNI mapping under it.

```
switch(config)# [no] encapsulation profile vni cisco
switch(config-vni-encap-prof)# [no] dot1q 20 vni 5000
```

Command to create an encapsulation profile template named ***cisco*** and add/delete the untagged frame VNI mapping under it.

```
switch(config)# [no] encapsulation profile vni cisco
switch(config-vni-encap-prof)# [no] untagged vni 6000
```

There are two types of VSIs - Numbered VSI and Default VSI. VSIs can be created under a physical port or a port channel. Numbered VSI range is from 1-4094 while 4095 VSI ID is reserved for default VSI. The default VSIs are by default set to admin up always. Note that a default VSI and a numbered VSI cannot exist together under the same parent port. Multiple numbered VSIs can be created under same parent port.

Command to create a numbered VSI and apply encapsulation profile under it.

```
switch(config)# interface ethernet3/1
switch(config-if)# service instance 1 vni
switch(config-if-srv)# no shut
switch(config-if-srv)# encapsulation profile cisco default
```

Command to create a default VSI with ***cisco*** as the encapsulation profile.

```
switch(config)# interface ethernet3/2
switch(config-if)# service instance vni default
switch(config-if-srv-def)# encapsulation profile cisco default
```

Sample VNI & VSI configuration:

```
switch(config)# vni 5000-5002
switch(config-vni)# no shutdown
switch(config-vni)# exit
switch(config)# bridge-domain 50-52
switch(config-bdomain)# member vni 5000-5002
switch(config-bdomain)# exit
switch(config)# encapsulation profile vni cisco
switch(config-vni-encap-prof)# dot1q 20-22  vni  5000-5002
switch(config-vni-encap-prof)# exit
switch(config)# interface ethernet9/1
switch(config-if)# no shutdown
switch(config-if)# service instance vni default
switch(config-if-srv-def)# encapsulation profile cisco default
```

Bridge-Domain Interface

A bridge-domain interface (BDI), is a virtual routed interface that connects a bridge-domain on the device to the Layer-3 router engine on the same device. Only one BDI can be associated with a bridge-domain. You must configure a BDI for a bridge-domain only when you want to route between bridge-domains or to provide IP host connectivity to the device through a virtual routing and forwarding (VRF) instance that is not the management VRF.

- You must enable the VLAN network interface feature before you can configure it.
- You must configure the BDI in the same virtual device context as the bridge-domain.
- You must create the bridge-domain range in the virtual device context, and BDI can only be created for that range. The configurations under a BDI are same as that under VLAN interface.
- You can route across BDI to provide Layer-3 inter-bridge-domain routing by configuring a BDI for each bridge-domain that you want to route traffic to and assigning an IP address on the BDI.

Configuring Bridge-Domain Interface

Before you begin

- Ensure that you are in the correct virtual data center (or use the **switchto vdc** command)

SUMMARY STEPS

1. **configure terminal**
2. **feature interface-vlan**
3. **interface bdi**
4. **ip address**
5. **ipv6 address**
6. **show interface bdi**
7. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters configuration mode.
Step 2	**feature interface-vlan** **Example:** `switch(config)# feature interface-vlan`	Enables BDI mode.
Step 3	**interface bdi** **Example:**	Creates a BDI. The *number* range specified in system bridge-domain command.

	Command or Action	Purpose
	`switch(config)# interface bdi 10`	
Step 4	**ip address** **Example:** `switch(config-if)# ip address 192.0.2.1/8`	Configures an IP address for this BDI.
Step 5	**ipv6 address** **Example:** `switch(config-if)# ipv6 address 2001:0DB8::1/8`	Configures an IPv6 address for this BDI.
Step 6	**show interface bdi** **Example:** `switch(config-if)# show interface vlan 10`	(Optional) Displays the Layer-3 interface statistics.
Step 7	**copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	(Optional) Saves this configuration change.

Example

The following example shows how to create a BDI:

```
switch# configure terminal
switch(config)# feature interface-vlan
switch(config)# interface bdi 10
switch(config-if)# ip address 192.0.2.1/8
switch(config-if)# copy running-config startup-config
```

Configuring Multiple Leaf

The following example shows the multi-tenancy support at leaf using VRFs:

```
system bridge-domain 2-3967
system fabric bridge-domain 3001-3967

configure profile vrf-tenant-profile
 vni $vrfSegmentId
 bridge-domain $bridgeDomainId
  member vni $vrfSegmentId
 interface bdi $bridgeDomainId
  vrf member $vrfName
  ip forward
  ipv6 forward
  no shutdown
configure terminal

bridge-domain 2,10-11
```

```
bridge-domain 2
 fabric-control
bridge-domain 2,10-11
 member vni 5000,10010-10011

vrf context Cisco:vrf1
 vni 20000
 ipv6 pim ssm range ff30::/12
 rd auto
 address-family ipv4 unicast
  route-target both auto
 address-family ipv6 unicast
  route-target both auto

interface Bdi10
 no shutdown
 vrf member Cisco:vrf1
 ip address 100.1.1.1/24
 fabric forwarding mode anycast-gateway

interface Bdi11
 no shutdown
 vrf member Cisco:vrf1
 ip address 100.1.2.1/24
 fabric forwarding mode proxy-gateway
```

CHAPTER 8

Segment ID Support for DHCP Relay

Feature Information for Segment ID

Table 5: Feature Information for Segment ID

Feature	Releases	Feature Information
DHCP	7.2(0)D1(1)	Included a new chapter on *Segment ID Support for DHCP Relay* . DHCP relay configuration.
DHCP Server	7.2(0)N1(1)	Included a new section on *Configuring Windows 2012 as DHCP Server* . Support common DHCP-Servers for IP address assignments within DFA.

Segment ID Support for DHCP Relay

This feature explains how a Cisco Nexus 7000 Series Switches perform the role of a DHCP relay in the DFA environment.

Note A detailed explanation of the DHCP feature documentation for Cisco Nexus 7000 Series Switches is available in the Configuring DHCP chapter of the *Cisco Nexus 7000 Series NX-OS Security Configuration Guide*.

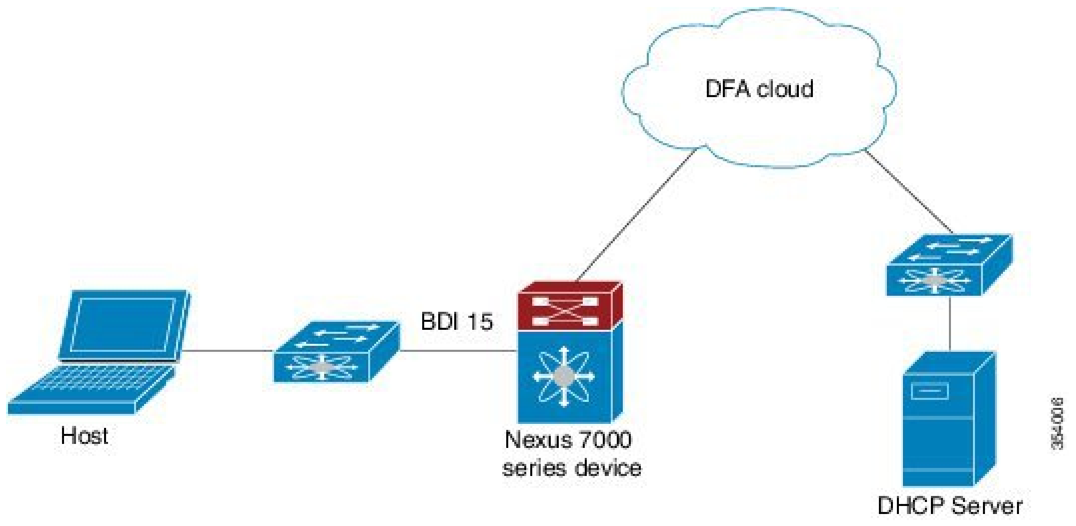
Guidelines and Limitations

- You should know about the DHCPv4 relay, DHCPv6 relay, BDI, and segment ID functions.

Information About Segment ID Support for DHCP Relay

DHCP Relay Configuration Overview

Figure 6: DHCP relay configuration through a BDI



The illustration depicts a Cisco Nexus 7000 Series Switches in a DFA environment (*Nexus 7000)*. The DHCP client (*Host*) seeking an IP address via DHCP is on the left side and the DHCP server that provides the IP address is on the right side (*DHCP Server*). Here, we configure the DHCP server address on a BDI of the Cisco Nexus 7000 Series Switch.

The following sections explain DHCP relay configuration:

1. Enabling a Cisco Nexus 7000 Series Switch as a DHCP relay agent.
2. Configuring a DHCP server address on the relay agent.
3. Configuring the VPN option for the DHCP relay agent.

Enabling a Cisco Nexus 7000 Series Switch as a DHCP Relay Agent

Enabling a Device as a DHCPv4 Relay Agent

```
switch# configure terminal
switch(config)# feature dhcp
switch(config)# ip dhcp relay
```

Enabling a Device as a DHCPv6 Relay Agent

```
switch# configure terminal
switch(config)# feature dhcp
switch(config)# ipv6 dhcp relay
```

Configuring a DHCP server address on the Relay Agent

Configuring a DHCPv4 Server Address on the Relay Agent

```
switch(config)# interface bdi 15
switch(config-if)# ip dhcp relay address 192.0.2.120 use-vrf management
```

- The interface level configuration command (**ip dhcp relay address**) is used to configure or disable a server address on a BDI.
- The **use-vrf** option is used to specify the VRF name of the server if the client and server are in different VRFs.

Configuring a DHCPv6 Server Address on the Relay Agent

```
switch(config)# interface bdi 15
switch(config-if)# ipv6 dhcp relay address 2001:DB8:1::1 use-vrf management2
```

- The interface level configuration command (**ipv6 dhcp relay address**) is used to configure or disable a server address on a BDI.
- The **use-vrf** is used to specify the VRF name of the server if the client and server are in different VRFs.
- The server address can either be a link scoped unicast or multicast address, or it can be a global or site local unicast or multicast address.
- An interface is required when the DHCP server address is a link local address or multicast address. It is not allowed for a unicast address.

Important You should be able to ping the server (for the specified server address) from the specified VRF.

Configuring the VPN option for the Relay Agent

Attention In a DFA environment with DHCP Relay, configuring the **vpn** option is mandatory. After configuring the **vpn** option, the DHCP server may be placed within the same or different VRF (default or management).

Configuring the VPN option for the DHCPv4 Relay Agent

```
switch# configure terminal
switch(config)# ip dhcp relay information option
switch(config)# ip dhcp relay information option vpn
```

- The global level **vpn** configuration command is used to enable or disable the DHCPv4 relay function within or across VRFs.

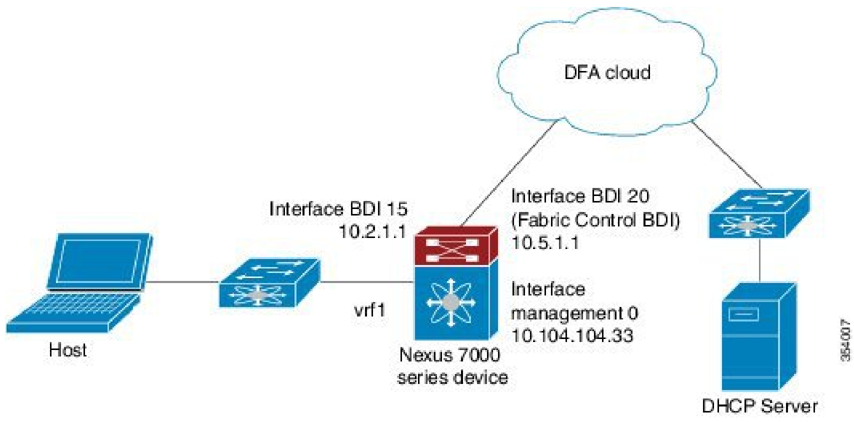
Configuring the VPN option for the DHCPv6 Relay Agent

```
switch# configure terminal
switch(config)# ipv6 dhcp relay option vpn
```

- The global level configuration command is used to enable or disable the DHCPv6 relay function across VRFs. When this is enabled, and the DHCPv6 server is in a different VRF, the relay agent inserts a virtual selection sub option in the *relay-forward* message. By default, this is disabled.

After DHCP relay configuration on a device and assignment of a DHCP server address through a BDI, the network topology looks like this:

Figure 7: DHCP relay configuration and DHCP server address assignment through a BDI



Configuring Windows 2012 as DHCP Server

You can have common DHCP-Servers (for example, Microsoft Windows) for IP address assignments within DFA. The DHCP-Servers can assign IP addresses to a simple DHCP request. The common DHCP-Server support does not rely on specific DHCP scope option (for example, simple-mode) by accepting some limitations or additional configuration.

We support Windows 2012 DHCP server by utilizing the 'Super Scope' as well as the policy on option 82 for address range selection. The DHCP policy on scope reserves the address space exclusively for the request matching the policy.

Note We support both Windows DHCPv4 and DHCPv6 servers and the configurations are similar to regular networks.

Let us assume the switch is using the address from subnet B (it can be the backbone subnet, management subnet, or any customer designated subnet for this purpose) to communicate with the Windows DHCP server. In DFA we have subnets S1, S2, S3, …, Sn for segment s1, s2, s3, …, sn.

To configure DHCP on Windows server.

1. Create a super scope. Within the super scope, create scope B, S1, S2, S3, …, Sn for the subnet B and the subnets for each segment.
2. In scope B, specify the 'Exclusion Range' to be the entire address range (so that the offered address range must not be from this scope).
3. For every segment scope Si, specify a policy that matches on Agent Circuit ID with value of '0108000600XXXXXX', where '0108000600' is a fixed value for all segments, the 6 numbers "XXXXXX" is the segment ID value in hexadecimal. Also ensure to check the **Append wildcard(*)** check box.
4. Set the policy address range to the entire range of the scope.

Configuring Infoblox as DHCP Server

Uses the Link Selection sub-option for scope selection, as this is by default set as the client facing SVI address. For other DHCP servers such as DHCPd and CPNR, GIAddr based scope selection is used. If you are already using Infoblox, then you must upgrade the Cisco NX-OS Switch to version 7.1(1)N1(1) or later.

Note We support only DHCPv4 for Infoblox and the configurations are similar to regular networks. You can refer to Infoblox user manual for configuration.

Let us consider a case where, the DHCP clients are VM hosts connecting to Cisco switches in DFA. The switches are configured with SVI as gateway for the VM hosts. The IP address of the SVI may not be unique in the DFA system, as when VM host moves to server connecting to another switch, then another SVI will be brought up on that switch and configured with the same gateway IP so that the VM does not need to change its gateway IP.

Configuring DHCPd as DHCP Server

The system has a centralized DHCP server that serves all VM hosts. Every switch has a DHCP relay agent running to forward the DHCP requests from VM hosts to the DHCP server. Because the SVI IP address is not unique, hence not reachable from the DHCP server, the relay agent on switch cannot use it as the GIAddr in the request. Instead, it uses another routing interface which has unique IP address as GIAddr. In order for the DHCP server to select the correct subnet for each host, the relay agent also put an identifier in the Circuit ID field in the Relay Agent Information option. The identifier uniquely identifies the subnet that a host connects to. However the identifier is only a portion of the Circuit ID.

Now on the DHCP server, you must configure it to fetch the identifier out of the Circuit ID and use the identifier to choose the right subnet. We are able to do this with DHCPd in the following way: we define classes matching on substring of the Circuit ID. All the host subnets are in a shared-network. The shared-network also contains the subnet for the routing interfaces on the switch, so that the shared-network will be picked when the request comes. The subnet for the routing interfaces does not have address pool, so it will not assign addresses. The address allocation is from the host subnets in the shared-network. Each host subnet only allows its own class members. Hence the server can correctly choose a subnet for address allocation based on the identifier carried in the request.

An example of the DHCPd configuration is given below. Here '59.2.8.0/24' and '99.1.3.0/24' are the host subnets, with identifier '01:5f:91' and '01:5f:92' respectively. Subnet '43.2.0.0/24' is the subnet of the routing interfaces. It is used to select the shared-network, but not used for address allocation.

```
# Start Segment 90001
class "15f91" {
match if substring (option agent.circuit-id, 5, 3) =01:5f:91;
}
# End Segment 90001

# Start Segment 90002
class "15f92" {
match if substring (option agent.circuit-id, 5, 3) =01:5f:92;
}
# End Segment 90002

shared-network "dfa-network" {

# Start Segment primarySubnet
subnet 43.2.0.0 netmask 255.255.255.0 {
}
# End Segment primarySubnet

# Start Segment 90001
subnet 59.2.8.0 netmask 255.255.255.0 {
option routers 59.2.8.1;
option vlan-id 90001;
 }
 pool {
allow members of "15f91";
range 59.2.8.2 59.2.8.254;
}
# End Segment 90001

# Start Segment 90002
subnet 99.1.3.0 netmask 255.255.255.0 {
option routers 99.1.3.1;
option vlan-id 90002;
 }
 pool {
allow members of "15f92";
range 99.1.3.2 99.1.3.254;
}
# End Segment 90002

}
```

CHAPTER 9

Tenant Configuration

Tenant Configuration

Dynamic Fabric Automation provides touchless tenant provisioning on the desired leaf nodes using auto-configuration. This chapter describes a set of configuration commands that are required for provisioning a tenant. A tenant configuration involves setting up both sides of the network, the server side and the fabric side of the network. All the commands described here are part of the configuration profiles defined in the Profiles Database of the Cisco Prime DCNM. Refer to configuration profiles on the Cisco Prime DCNM for a predefined end-host auto-configuration profile, defaultUniversalNetworkEfProfile, vrf-common-universal-profile, and the vrf-tenant-profile which is pre-configured on the leaf switch.

Tenant Configuration (Server-side)

The server-side configuration at a high level involves:

Multi-tenancy lite version

1. Creating a VLAN for the servers that are attached to the tenant. The server can be either a virtual or physical server. This VLAN is local to the leaf nodes to which the server is connected and is in mode FabricPath.
2. Specifying a globally identifiable segment ID, which will be used to send traffic to these servers over the fabric.
3. Creating a server facing SVI to provide the pervasive first hop anycast gateway.
4. Choosing the right fabric mode for the IP traffic; that is, if an enhanced forwarding using the proxy-gateway is required or a traditional forwarding using the anycast-gateway option.

A sample configuration is as follows:

1. Configure terminal.
2. Configure the VLAN in mode FabricPath for attaching the workloads.
3. Specify that the workload is a part of a global segment ID, for example 10000.
4. Specify that the global segment ID is available over the FabricPath network.

5. Specify the distributed anycast gateway and the IP subnet that this tenant's workload belongs to.
6. Specify that this IP network that is created is a part of tenants VRF.
7. Specify the anycast gateway IP address.
8. Specify the anycast gateway IPv6 address.
9. Specify the mode for anycast distributed gateway.
10. Use this command if the distributed anycast gateway mode is a proxy gateway as there will be virtual machines under the same leaf, which are within the same subnet.
11. Administratively enable the interfaces.

```
configure terminal
 vlan 100
 vn-segment 10000
 mode fabricpath
 interface vlan 100
  vrf member xyz
  ip address 10.1.1.1/24
  ipv6 address 10:1:1::1/64
  fabric forwarding mode [anycast-gateway | proxy-gateway]
  no ip redirects
  no ipv6 redirects
  no shutdown
```

Multi-tenancy full version

1. Creating a bridge-domain for the servers that are attached to the tenant. The server can be either a virtual or physical server. This bridge-domain is local to the leaf nodes to which the server is connected and is in mode FabricPath.
2. Specifying a globally identifiable segment ID, which will be used to send traffic to these servers over the fabric.
3. Creating a server facing SVI to provide the pervasive first hop anycast gateway.
4. Choosing the right fabric mode for the IP traffic; that is, if an enhanced forwarding using the proxy-gateway is required or a traditional forwarding using the anycast-gateway option.

A sample configuration is as follows:

1. Configure terminal.
2. Configure the bridge-domain for attaching the workloads.
3. Specify that the workload is a part of a global VNI, for example 10000.
4. Specify that the global VNI is available over the FabricPath network.
5. Specify the distributed anycast gateway and the IP subnet that this tenant's workload belongs to.
6. Specify that this IP network that is created is a part of tenants VRF.
7. Specify the anycast gateway IP address.
8. Specify the anycast gateway IPv6 address.
9. Specify the mode for anycast distributed gateway.

10. Use this command if the distributed anycast gateway mode is a proxy gateway as there will be virtual machines under the same leaf, which are within the same subnet.

11. Administratively enable the interfaces.

```
configure terminal
 bridge-domain 100
  member vni 10000
 vni 10000
 interface bdi 100
  vrf member xyz
  ip address 10.1.1.1/24
  ipv6 address 10:1:1::1/64
  fabric forwarding mode [anycast-gateway | proxy-gateway]
  no ip redirects
  no ipv6 redirects
  no shutdown
```

Tenant Configuration (Fabric-side)

Multi-tenancy lite version

The fabric-side configuration at a high level involves:

1. Creating the required VRF for the tenant.
2. Setting the Layer-3 Segment or the VNI (as shown below) that is used for identifying the routed traffic of the tenant globally.
3. BGP route redistribution to the route reflector.

The following steps detail the configuration steps:

1. Creating the VRF for the tenant— When a Layer-3 segment is configured, configuration corresponding to the Layer-3 segment is performed using a specific configuration profile. This configuration profiles must be available as part of the POAP of the leaf node. IP forward and IPv6 forward in this configuration profile indicates Layer-3 IPv4/IPv6 forwarding is required on this SVI.

Note The keyword 'VNI' in the VRF context helps to trigger this specific config-profile.

A sample configuration is as follows:

```
Configure profile vrf-tenant-profile
   vlan $vrfVlanId
       vn-segment $vrfSegmentId
       mode fabricpath
  interface vlan $vrfVlanId
   vrf member $vrfName
   ip forward
   no ip redirects
   ipv6 forward
   no ipv6 redirects
   no shutdown
```

2. VRF context tenant— Specify a partition for this tenant. Each tenant will be placed in its own VRF.

3. VNI 100000— Specify the Layer-3 segment ID of the tenant, which is used for routing traffic over the fabric to the different leaf nodes.
4. rd auto— Specify the route distinguisher (RD) value. With fabric, the manual specification of RD becomes unnecessary. With this command a RD value is automatically generated by the BGP process for each VRF, using the fabric control segment IP address and the VRF ID of the VRF.
5. address family ipv4 unicast— Enable the IPv4 address family for BGP.
6. route-target both auto— Specify that the route-targets automatically generated are both exported and imported on this leaf.
7. address family ipv6 unicast— Enable the IPv6 address family for BGP.
8. route-target both auto— Specify that the route-targets automatically generated are both exported and imported on this leaf.
9. router BGP 100— Enable route distribution to the route reflector so that it is available in the other leaf nodes. Whatever is configured under BGP is relevant only for the end-host SVI and not applicable for the core-facing SVI.
10. vrf tenant— Specify the VRF name for which this distribution is configured.
11. address family IPv4 unicast— Enable the IPv4 address family for BGP.
12. redistribute hmm route-map redist-host— Any address resolution protocols that are learnt under an SVI in fabric forwarding mode can be distributed using BGP to the other leaf nodes.

Multi-tenancy full version

The fabric-side configuration at a high level involves:

1. Creating the required VRF for the tenant.
2. Setting the Layer-3 Segment or the VNI (as shown below) that is used for identifying the routed traffic of the tenant globally.
3. BGP route redistribution to the route reflector.

The following steps detail the configuration steps:

1. Creating the VRF for the tenant— When a Layer-3 segment is configured, configuration corresponding to the Layer-3 segment is performed using a specific configuration profile. This configuration profiles must be available as part of the POAP of the leaf node. IP forward and IPv6 forward in this configuration profile indicates Layer-3 IPv4/IPv6 forwarding is required on this BDI.

Note The keyword 'VNI' in the VRF context helps to trigger this specific config-profile.

A sample configuration of vrf-tenant-profile is as follows:

```
Configure profile vrf-tenant-profile
   bridge-domain $vrfbridge-domainId
     member vni $vrfSegmentId
  vni $vrfSegmentId
  interface bdi $vrfbridge-domainId
   vrf member $vrfName
   ip forward
   no ip redirects
   ipv6 forward
   no ipv6 redirects
   no shutdown
```

A sample configuration of vrf-common-profile is as follows:

```
vrf context $vrfName
vni $vrfSegmentId
rd auto
ip route 0.0.0.0/0 $include_serviceNodeIpAddress
  address-family ipv4 unicast
     route-target import $include_borderLeafRt
  route-target both auto
address-family ipv6 unicast
     route-target import $include_borderLeafRt
  route-target both auto
router bgp $asn
     vrf $vrfName
   address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
   address-family ipv6 unicast
       redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
```

2. VRF context tenant— Specify a partition for this tenant. Each tenant will be placed in its own VRF.
3. VNI 100000— Specify the Layer-3 VNI of the tenant, which is used for routing traffic over the fabric to the different leaf nodes.
4. rd auto— Specify the Route Distinguisher (RD) value. With fabric, the manual specification of RD becomes unnecessary. With this command a RD value is automatically generated by the BGP process for each VRF, using the fabric control segment IP address and the VRF ID of the VRF.
5. address family ipv4 unicast— Enable the IPv4 address family for BGP.
6. route-target both auto— Specify that the route-targets automatically generated are both exported and imported on this leaf.
7. address family ipv6 unicast— Enable IPv6 address family for BGP.
8. route-target both auto— Specify that the route-targets automatically generated are both exported and imported on this leaf.
9. router BGP 100— Enable route distribution to the route reflector so that it is available in the other leaf nodes. Whatever is configured under BGP is relevant only for the end-host BDI and not applicable for the core-facing BDI.
10. vrf tenant— Specify the VRF name for which this distribution is configured.
11. address family IPv4 unicast— Enable the IPv4 address family for BGP.
12. redistribute hmm route-map redist-host— Any address resolution protocols that are learnt under an BDI in fabric forwarding mode can be distributed using BGP to the other leaf nodes.

CHAPTER 10

Auto-Configuration

Feature Information for Auto-Configuration

Table 6: Feature Information for Auto-Configuration

Feature	Releases	Feature Information
Auto-configuration	7.2(0)D1(1)	Included a new section on *Dynamic provisioning*. This feature simplifies the management of the VRF and VLAN/BD configurations.
Routable Loopback Address	7.2(0)N1(1)	Updated the example of a profile that auto-configures a routable loopback with **configure profile vrf-common-loopback-universal**command. VRF profile is updated on the leaf resulting in the loopback routable IP address being auto-configured under that VRF.
Logging of Profile instantiation	7.2(0)N1(1)	Updated the *Profile Refresh* section. Enhanced syslogs are generated when profile apply, profile un-apply, and profile refresh is performed.

Auto-Configuration

This chapter briefly describes about the following:

- Configuration Profile
- Universal Profile
- Profile Refresh
- Profile Migration

Note Auto-configuration with multicast is not supported, also the border leaf needs to be configured manually.

Information About Auto-Configuration in DFA

Configuration Profile

A configuration profile in Cisco Programmable Fabric is a collection of commands used to instantiate a specific configuration. Based on appropriate end-host triggers (VDP or data plane trigger (any data frame)), the configuration profiles are grouped to allow flexible and extensible options to instantiate day-1 tenant-related configurations on a Cisco Programmable Fabric leaf on need basis.

The commands are entered using variables for certain parameters instead of entering the actual value. The switch then populates the actual values to derive the completed command. When the required parameters for a particular configuration profile are available, the profile can be instantiated to create a configuration set. The switch then applies this configuration set to complete the command execution belonging to the configuration set.

The commands which are supported under a configuration profile are called config-profile-aware commands. Most of the commands in the switch can be entered into the configuration profile.

Note Various sets of configuration profiles can be created and stored in the network database, and each network can use a different configuration profile. The configuration profiles can be used from the network to set up on the leaf whenever required.

Universal Profile

Note The Universal Profiles are available since Cisco NX-OS 7.1(0)N1(1) for Cisco Nexus 5000 and 6000 Series Switches and from Cisco NX-OS 7.2(0)D1(1) for Cisco Nexus 7000 Series Switches and Cisco NX-OS 7.1(1) on Cisco DCNM.

Universal profiles are enhanced configuration profiles with the ability to support optional parameters and non-disruptive profile refresh. The support is also extended to VRF profiles by using the **include profile any** command in the network individual VRF profiles. VRF profiles (vrf-common) need not be statically defined using POAP; instead select any universal VRF profile in Cisco DCNM while creating a partition (vrf-common-universal, vrf-common, and so on). During the instantiation of the network, the system downloads the selected VRF profile for the partition and performs the necessary VRF, VLAN/SVI or bridge-domain/BDI and BGP configurations.

Profile Refresh

Profile refresh involves updating and/or removing profile parameters (arguments or variables) without disrupting the traffic while using universal profiles. After the changes are done, Cisco DCNM executes the **fabric database refresh vni/vrf** command.

Profile Migration

Migration involves moving from an existing applied individual profile to a universal profile, and it is necessary if features such as optional parameter and refresh are required. Migration needs to be done for all the networks under a partition/VRF at the same time. Within the same VRF/partition, a combination of universal and individual network profiles are not supported. To migrate the networks under partition/VRF, change the network profile from defaultNetworkProfile to defaultNetworkUniversalProfile, which is a disruptive process.

LDAP Configuration

There are three different tables, which you can query from:

- Network Table
- Partition Table
- Profile Table

Network Table

All the parameters for a network host are stored in this table in the LDAP. ToR will query this LDAP table for network parameters based on the following configuration:

```
Switch# configure terminal
Switch(config)# fabric database type network
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=networks, dc=cisco, dc=com key-type 1
```

Partition Table

All the parameters that are required to provision a VRF on the ToR are stored in this table. When a network is using a Universal Profile, querying this table can identify corresponding VRF profile. ToR queries this table for every new network that is provisioned using Universal Profile. The following configuration helps ToR to query the partition table:

```
Switch# configure terminal
Switch(config)# fabric database type partition
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=partitions, dc=cisco, dc=com
```

Profile Table

Multi-tenancy lite version

This table is used to store configuration profiles, which are pre-packaged with DCNM and custom config profiles that are created. The Cisco Nexus 5000 Series Switches or Cisco Nexus 6000 Series Switches employ this tenancy model where a maximum of 4K VLANs are supported on the ToR. So, if a profile is not pre-configured on the system then ToR will query profile table to download profile contents and cache it locally.

The following configuration helps ToR to query the profile table:

```
Switch# configure terminal
Switch(config)# fabric database type profile
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=profiles, dc=cisco, dc=com
```

Multi-tenancy full version

The Cisco Nexus 7000 Series Switches employ this tenancy model where, upto 4K VLANs or dot1q tags can be supported on a per port basis.

The following configuration helps ToR to query the profile table:

```
Switch# configure terminal
Switch(config)# fabric database type profile
Switch(config)# server protocol ldap ip 10.1.1.2 vrf <default/management>
Switch(config)# db-table ou=profilesBridgeDomain, dc=cisco, dc=com
```

Specifying profile Mapping for network instances

It is necessary to specify how a particular network instance will be provisioned. That is either using network database lookup or by using a static Profile on the switch. The following configuration describes how to configure ToR to fetch network parameters from Remote database (LDAP).

```
Device(config)# fabric database profile-map global
Device(config-profile-map-global)# ethernet-tag encapsulation vni default dynamic
Device(config-profile-map-global)# vdp vni default dynamic
```

Configuring a Profile

SUMMARY STEPS

1. **configure profile** *profile-name*
2. **interface vlan** *vlan-id* or **interface bdi** *bd-id*
3. **no shutdown**
4. **no ip redirects**
5. **include profile** *profile-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure profile** *profile-name* **Example:** `Switch(config)# configure profile profile1`	Creates a configuration profile.
Step 2	**interface vlan** *vlan-id* or **interface bdi** *bd-id* **Example:** `Switch(config)# interface vlan 50`	Defines a VLAN or BDI and enters interface configuration mode.
Step 3	**no shutdown** **Example:** `Switch(config-if)# no shutdown`	Administratively enables the interface.
Step 4	**no ip redirects** **Example:** `Switch(config-if)# no ip redirects`	Disables sending ICMP redirect messages.

	Command or Action	Purpose
Step 5	**include profile** *profile-name* **Example:** `Switch(config-if)# include profile profile2`	Includes a particular profile in another configuration profile, which will be uniquely instantiated for a set of parent profiles belonging to the same configuration set.

Example for Configuring an Individual Profile

Example for Configuring a Universal Profile

Multi-tenancy lite version

The following is an example of an individual network configuration profile:

```
configure profile defaultNetworkIpv4EfProfile
 vlan $vlanId
 vn-segment $segmentId
 mode fabricpath
interface vlan $vlanId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ip dhcp relay address $dhcpServerAddr use-vrf default
 fabric forwarding mode proxy-gateway
 no ip redirects
 no shutdown
include profile vrf-common
```

Multi-tenancy full version

```
config profile defaultNetworkIpv4EfProfile
 vni $segmentId
 bridge-domain $bridgeDomainId
  member vni $segmentId
interface bdi $bridgeDomainId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ip dhcp relay address $dhcpServerAddr use-vrf management
 fabric forwarding mode proxy-gateway
 no ip redirects
 no shutdown
include profile vrf-common
```

Multi-tenancy lite version

The following is an example for a universal configuration profile:

```
configure profile defaultNetworkUniversalEfProfile
 vlan $vlanId
 vn-segment $segmentId
 mode fabricpath
interface vlan $vlanId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ip dhcp relay address $dhcpServerAddr use-vrf $vrfDhcp
 ipv6 address $gatewayIpv6Address/$prefixLength tag 12345
 fabric forwarding mode proxy-gateway
 no ip redirects
 no ipv6 redirects
 no shutdown
include profile any
```

Multi-tenancy full version

The following is an example for a universal configuration profile:

```
configure profile defaultNetworkUniversalEfProfile
 vni $segmentId
 bridge-domain $bridgeDomainId
 member vni $segmentId
interface bdi $bridgeDomainId
 vrf member $vrfName
 ip address $gatewayIpAddress/$netMaskLength tag 12345
 ipv6 address $gatewayIpv6Address/$prefixLength tag 12345
 fabric forwarding mode proxy-gateway
 no ip redirects
 no ipv6 redirects
 no shutdown
include profile any
```

The following is an example for vrf-common-universal configuration profile:

```
configure profile vrf-common-universal
 vrf context $vrfName
 vni $include_vrfSegmentId
 rd auto
 ip route 0.0.0.0/0 $include_serviceNodeIpAddress
  address-family ipv4 unicast
    route-target import $include_borderLeafRt
    route-target both auto
  address-family ipv6 unicast
    route-target import $include_borderLeafRt
    route-target both auto
router bgp $asn
 vrf $vrfName
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
  address-family ipv6 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
```

The VRF profile is updated on the leaf resulting in the loopback routable IP address being auto-configured under that VRF as well as advertised via MP-BGP to all leaf nodes.

The following is an example of a profile that auto-configures a routable loopback interface per ToR per VRF. This profile is pre-packaged in DCNM and looks as follows. Any parameter prefixed with the keynote as 'system_auto_' indicates that the corresponding value will be auto-generated by the ToR.

```
configure profile vrf-common-loopback-universal
interface loopback $system_auto_loopbackId
  vrf member $vrfName
 ip address $system_auto_backboneIpAddress/32 tag 12345
```

```
  vrf context $vrfName
  vni $include_vrfSegmentId
 rd auto
 ip route 0.0.0.0/0 $include_serviceNodeIpAddress
  address-family ipv4 unicast
    route-target both auto
  address-family ipv6 unicast
    route-target both auto
router bgp $asn
 vrf $vrfName
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
    address-family ipv6 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    redistribute direct route-map FABRIC-RMAP-REDIST-SUBNET
    maximum-paths ibgp 2
```

Dynamic Provisioning

Note This section is applicable only for multi-tenancy full version.

Dynamic provisioning simplifies the management of the VRF and VLAN/BD configurations. Dynamic provisioning can be triggered by:

- Any data frame Frame snooping
- VDP signaling from the server

Per-port configuration can be used to enable or disable dynamic provisioning. When the VM comes up in a leaf switch for the first time, the port receives an ARP or ND packet and for cases where VDP is enabled, the hypervisor may issue a VDP packet. The platform is expected to punt these packets to the HMM component, which will then install a tenant profile based on the information in this packet. In the simplest case, a (Port, VLAN) information from the incoming packet will be used to derive the tenant profile that needs to be installed. The following needs to be implemented when the Vinci functionality is enabled on the switch to support this feature:

- The server-facing port needs to be configured with 'default VSI' and 'auto-config' must be enabled. The Ingress CBL state must be set to allow packets in all the VLANs. Note that for DHCP based snooping, 'feature dhcp' should be enabled.
- Global ACLs must be set up to punt the packets of interest such as ARP and ND to the HMM component. All the remaining packets must be dropped to retain the CBL behavior. The global ACLs must be used only for the (Port, VLAN) that has not been configured on the system.

Once the tenant profile is installed, the VLAN/BD configuration along with the associated port membership will be configured.

The following is an example for VN-Segment configuration:

```
system bridge-domain 2-3967
```

```
feature vni
 vni 5000, 10010-10011, 20000

 system fabric bridge-domain 3001-3967
 bridge-domain 2,10-11

vrf context management

 encapsulation profile vni all_tenants
 dot1q 6-204 vni 7002-7200
 encapsulation profile vni cisco_all
 dot1q 10-1010 vni 10010-11010
 bridge-domain 2,10-11
 member vni 5000,10010-10011

interface Ethernet3/6
 no shutdown
 service instance 3 vni
 no shutdown
 encapsulation profile cisco_all default
```

The following is an example for VN-Segment dynamic auto-configuration:

```
interface ethernet 2/2
  service instance vni default
      encapsulation dynamic vdp


interface ethernet 2/2
  service instance vni default
      encapsulation dynamic frame-snoop profile cisco
```

CHAPTER 11

Auto-Configuration of Layer-3 DCI

Border Leaf and DC Edge Router Auto-Configuration

This feature works in partnership with Cisco Prime DCNM (starting with version 7.1.1 release) to enable auto-configuration of Layer-3 fabric External connectivity on a per tenant basis. Enhancements and modifications have been made to the following list of products and components in order to simplify and automate the extension of the tenant networks:

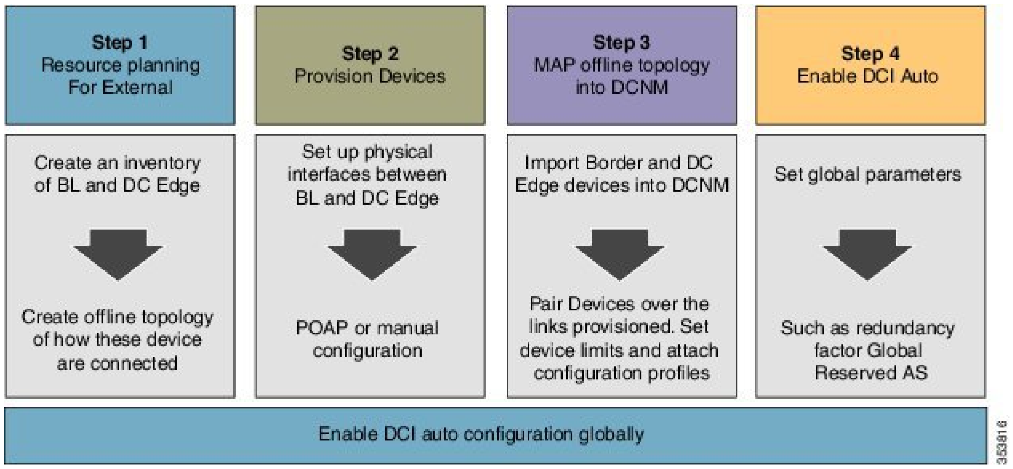
- Nexus 5600/6000/7000 as Border Leaf (Cisco NX-OS)
  - New POAP template for border leaf
  - Auto-configuration for VRF extension
- Cisco Prime DCNM 7.1
  - GUI enhancement for extending connectivity
  - Modification of LDAP Schema
- Cisco UCS Director 5.2
  - Workflow extension
- OpenStack (Juno)
  - Addition of border leaf extension capability

This functionality can be used towards the Data Center Edge router for WAN connectivity beyond the fabric or to introduce Layer-3 Data Center Interconnect (DCI) capabilities.

In this release we introduce automation of border leaf by using auto-configuration for the most common topologies that Customers use to connect to the DC edge router and connectivity beyond the fabric. In subsequent releases we will enhance this functionality with deep integration of MPLS Layer-3 VPN. This guide will use a sample topology to illustrate and guide on how this functionality works.

Summary of Steps

Figure 8: Provisioning Steps


Step 1
Resource planning For External
Step 2
Provision Devices
Step 3
MAP offline topology into DCNM
Step 4
Enable DCI Auto
Create an inventory of BL and DC Edge
Create offline topology of how these device are connected
Set up physical interfaces between BL and DC Edge
POAP or manual configuration
Import Border and DC Edge devices into DCNM
Pair Devices over the links provisioned. Set device limits and attach configuration profiles
Set global parameters
Such as redundancy factor Global Reserved AS
Enable DCI auto configuration globally
353816

Detailed Steps

Resource Planning

The two most common topologies for the border leaf to DC Edge connectivity are described as follows. Each pair of border leaf can be repeated multiple times in order to scale-out and achieve the desired scale and redundancy for you specific tenant extension needs:

- Direct Connect
  - A border leaf is only connected to the neighboring DC edge router. This topology can also be achieved with a pair of border leaf to achieve a certain level of redundancy.
- Full mesh
  - A pair of border leaf is connected to a pair of DC edge routers. This is the recommended topology as it offers the highest level of redundancy for failure of every node.

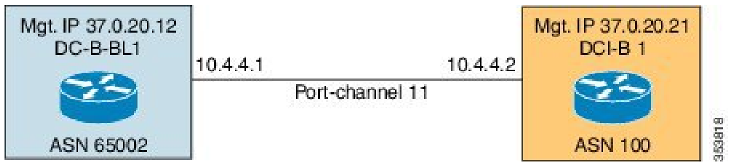
Assign the desired IP addresses to the sub interfaces that will be automatically created. The same address pairs can be reused on a per-VRF base as long as these are not redistributed into a routing protocol and stay unique within the VRF.

Note The maximum number of DC edge neighbors to a border leaf can have is two, similarly the maximum number of border leaf neighbors a DC edge router can have is also two. In future releases, will increase the amount of pairing neighbors.

Direct Connect Topology

A border leaf is connected to a DC edge device. This pairing can be repeated any number of times as per requirements. This does not offer redundancy on DC edge node failure. Multiple such directly connected pairs can be present in the topology.

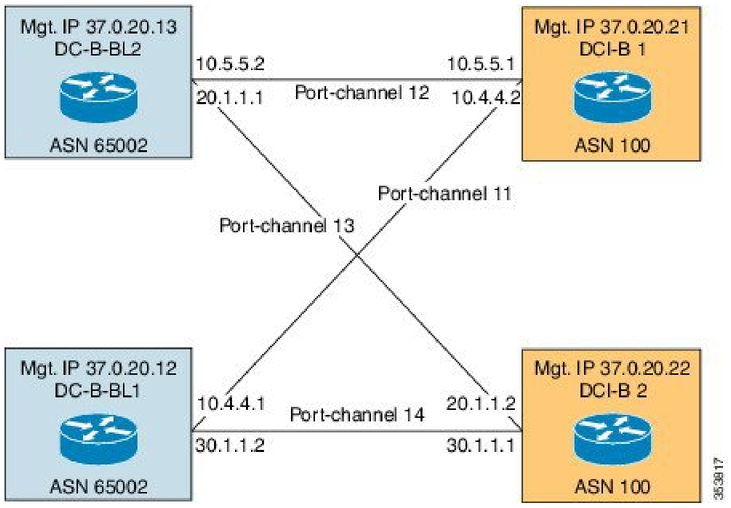
Figure 9: Direct Connect Topology



Full Mesh Topology

This topology provides full redundancy and it is the most recommended way to set up the border leaf to DC edge connectivity. The below figure is referred to a Topology with a redundancy factor of 2 but can be replicated any number of times as per requirements to satisfy the scale and redundancy.

Figure 10: Full Mesh Topology



Note The above topology examples show port-channel as the connected interface, as this is the best practice for redundancy. Ethernet interfaces are also supported, for this release the user will have to configure those on the box and not through POAP.

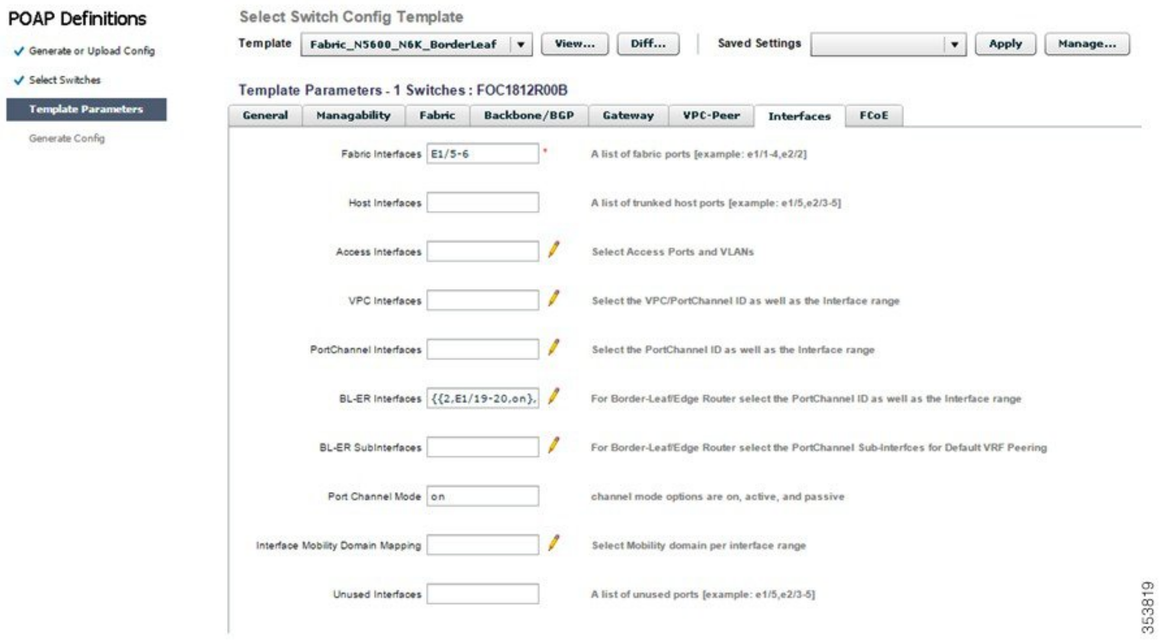
Provision Devices

POAP can be used to bring up the devices and the respective port-channels for the external connectivity between the devices. The port-channels between the border leaf and DC edge, 'can be a single member port-channel' and will be brought up with 'no switchport' mode so that a Layer-3 sub interfaces can be formed. The interfaces, port-channels and members, should be brought up with 'no shutdown'.

Note In the first release, we only support Layer-3 port-channel for neighbor connectivity and the POAP Template is only supported for the Cisco Nexus 5600 Platform Switches and Cisco Nexus 6000 Series Switches.

The below figure shows the POAP configuration for the border leaf to DC edge interface connection, which is port-channel only; single member port-channel is supported.

Figure 11: POAP Interface Settings for Border Leaf

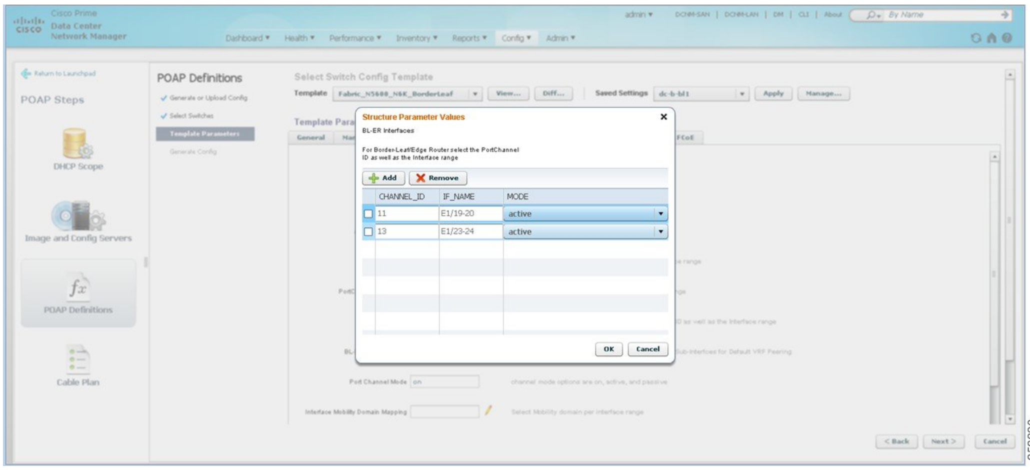


Template: Fabric_N5600_N6K_BorderLeaf
Saved Settings:

Template Parameters - 1 Switches : FOC1812R00B

Fabric Interfaces	E1/5-6	A list of fabric ports [example: e1/1-4,e2/2]
Host Interfaces		A list of trunked host ports [example: e1/5,e2/3-5]
Access Interfaces		Select Access Ports and VLANs
VPC Interfaces		Select the VPC/PortChannel ID as well as the Interface range
PortChannel Interfaces		Select the PortChannel ID as well as the Interface range
BL-ER Interfaces	{{2,E1/19-20,on},	For Border-Leaf/Edge Router select the PortChannel ID as well as the Interface range
BL-ER SubInterfaces		For Border-Leaf/Edge Router select the PortChannel Sub-Interfces for Default VRF Peering
Port Channel Mode	on	channel mode options are on, active, and passive
Interface Mobility Domain Mapping		Select Mobility domain per interface range
Unused Interfaces		A list of unused ports [example: e1/5,e2/3-5]

Figure 12: Interface Settings for Border Leaf (detail)



Template: Fabric_N5600_N6K_BorderLeaf
Saved Settings: dc-b-bl1

Structure Parameter Values

BL-ER Interfaces

For Border-Leaf/Edge Router select the PortChannel ID as well as the Interface range

CHANNEL_ID	IF_NAME	MODE
11	E1/19-20	active
13	E1/23-24	active

Port Channel Mode: on — channel mode options are on, active, and passive

Interface Mobility Domain Mapping — Select Mobility domain per interface range

Manually Configure Interfaces between Border Leaf and DC Edge router

The links between border leaf and DC edge will use sub interfaces. The belonging parent interfaces have to be "no switchport" mode in order to provide Layer-3 sub interface functionality. Logical port-channel interfaces provide link redundancy and module redundancy if members from different modules. They also provide more bandwidth by bundling several interfaces together.

Following are the interface configuration examples:

Example using logical port-channel interfaces:

```
interface Ethernet1/3
  no switchport
  channel-group 11 mode active

interface Ethernet1/4
  no switchport
  channel-group 11 mode active

interface port-channel11
  no switchport
```

Example using physical ethernet interface:

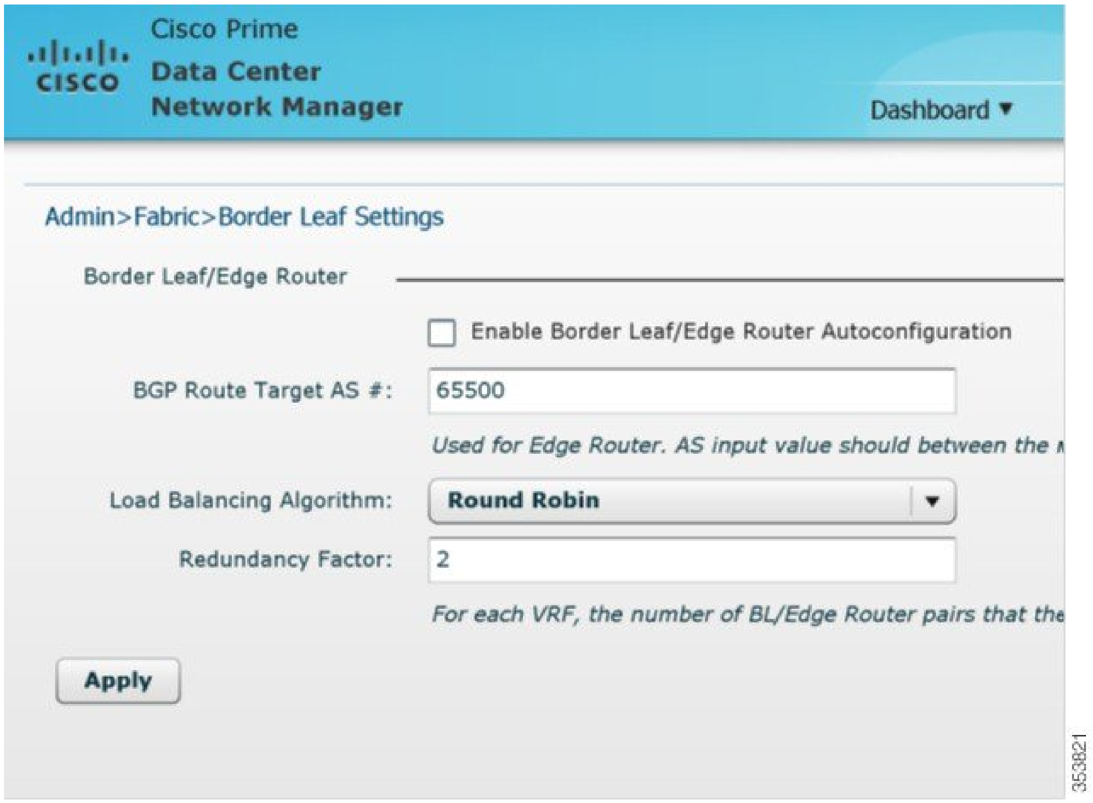
```
interface Ethernet1/4
  no switchport
```

Mapping offline topology into DCNM

Note Disable border leaf and DC edge router auto-configuration globally till topology mapping is done. This process will avoid partial configuration during setup phase. The default state is disabled.

1. In DCNM, choose **Admin** and select **Border Leaf Settings** option from the **Fabric** area.
2. Uncheck **Enable Border Leaf/Edge Autoconfiguration** check box to globally disable. By default it is unchecked.
3. Click **Apply**.

Figure 13: Border Leaf Settings - Disable auto-configuration population



Enable Border Leaf/Edge Router Autoconfiguration

BGP Route Target AS #: 65500

Used for Edge Router. AS input value should between the

Load Balancing Algorithm: Round Robin

Redundancy Factor: 2

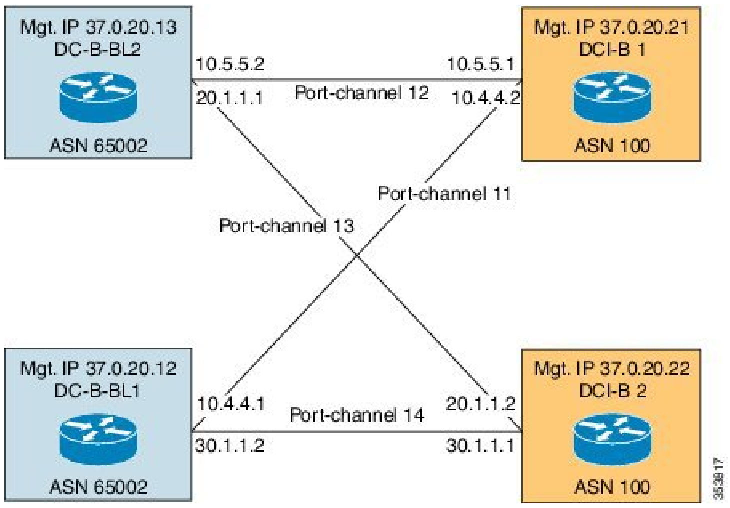
For each VRF, the number of BL/Edge Router pairs that the

For more information on parameter description, see Global Enable section.

Border Leaf to DC Edge router pairing

The following steps shows an example of using Cisco Prime DCNM 7.1, with Cisco Nexus 7000/7700 Series Switches, running Cisco NX-OS 6.2, and Cisco Nexus 5600/6000 Border Leaf Series Switches running Cisco NX-OS 7.1. With this combination, end-to-end auto-configuration for border leaf is supported. Cisco Nexus 7000/7700 configuration is generated within Cisco Prime DCNM and can be transported via copy/pasted into the Cisco Nexus 7000 command prompt. Future releases of Cisco Nexus 7000/7700, the admin can choose to deliver the Cisco Nexus 7000 DC edge configuration automatically. The option to send configuration to the DC edge device or not is controlled at Cisco Prime DCNM GUI when importing the DC edge device into Cisco Prime DCNM for device pairing.

Figure 14: Example Topology



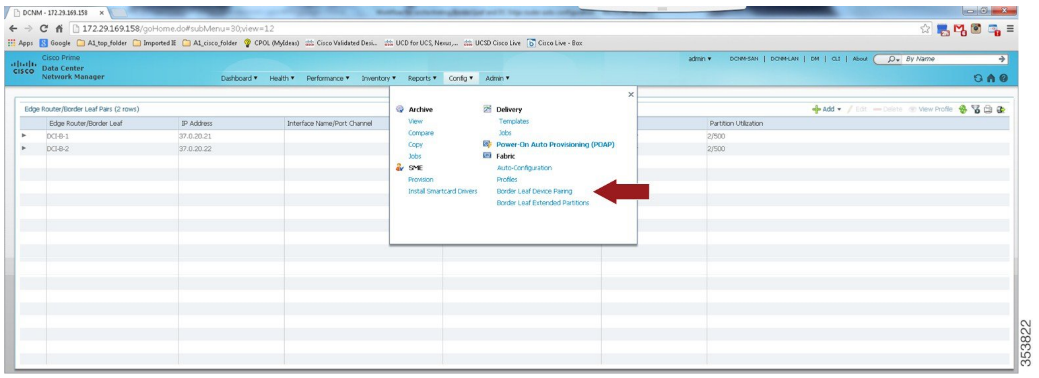
Note The neighboring DC edge router does not need to be imported or managed by Cisco Prime DCNM. For the following examples and figures the value is assumed to be 500 VRF as maximum.

Pair Border Leaf # 1 and DC Edge router # 1

To add DC edge device:

1. On Cisco Prime DCNM, click **Config** tab and select **Border Leaf Device Pairing** option.

Figure 15: Menu: Border Leaf Device Pairing

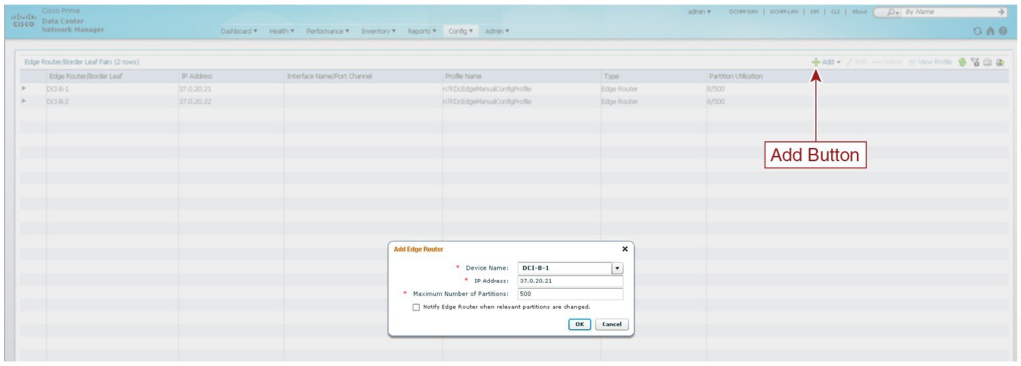


Edge Router/Border Leaf Pairs (2 rows)

Edge Router/Border Leaf	IP Address	Interface Name/Port Channel	Partition Utilization
DCI-B-1	37.0.20.21		2/500
DCI-B-2	37.0.20.22		2/500

2. Click **Add**.

Figure 16: Add Edge Router



Edge Router/Border Leaf Pairs (2 rows)

Edge Router/Border Leaf	IP Address	Interface Name/Port Channel	Profile Name	Type	Partition Utilization
DCI-B-1	37.0.20.21		n7KDcEdgeManualConfigProfile	Edge Router	8/500
DCI-B-2	37.0.20.22		n7KDcEdgeManualConfigProfile	Edge Router	8/500

Add Edge Router

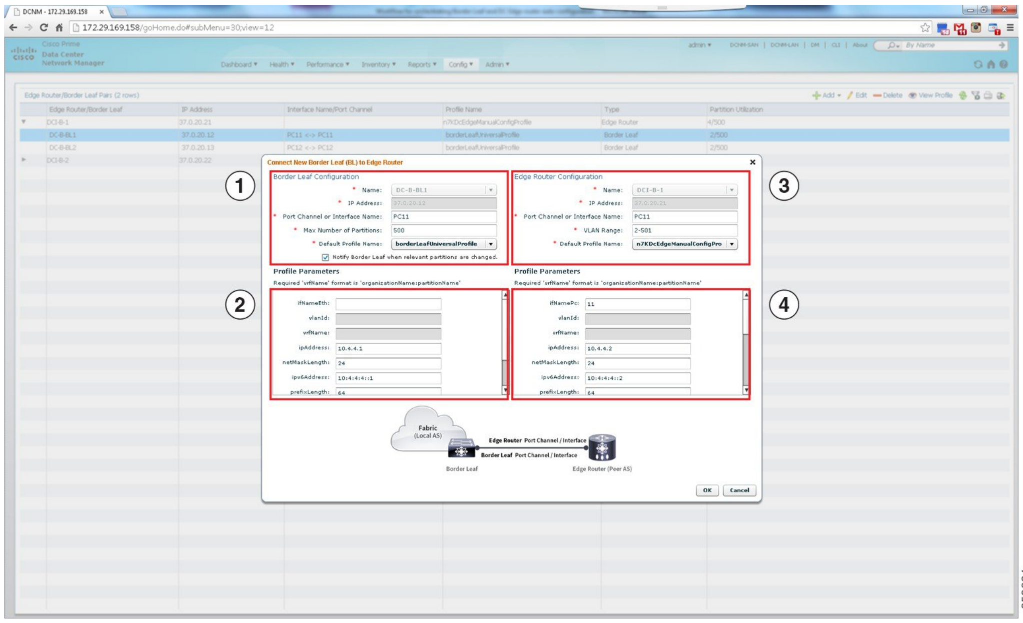
* Device Name: DCI-B-1
* IP Address: 37.0.20.21
* Maximum Number of Partitions: 500
☐ Notify Edge Router when relevant partitions are changed.

3. In **Add Edge Router** dialog box, enter the following details:
   - Device Name: (Optional) Enter the device name as DCI-B-1. It is not mandatory to enter the device name to import into Cisco Prime DCNM.
   - IP Address: Enter the management IP address.
   - Maximum Number of Partitions: Enter the maximum number of partitions on this device as 500.
   - Notify Edge Router when relevant partitions are changed: Check this check box only when you have Cisco Nexus 7000 Series Switches that supports this feature. We recommend you to uncheck this check box.
4. Click **OK**.
5. Select the **DCI-B-1** device.
6. Click **Add** and select **Border Leaf**.
7. Select **BL-B-1** from the drop-down list of border leafs imported into Cisco Prime DCNM.

   If BL-B-1 does not show up in the drop-down list, then it has not been imported into Cisco Prime DCNM properly with the role of border leaf.

   The **Connect New Border Leaf (BL) to Edge Router** shows the pairing values and meaning:

Figure 17: Add Border Leaf


1
2
3
4
Fabric (Local AS)
Edge Router Port Channel / Interface
Border Leaf Port Channel / Interface
Border Leaf
Edge Router (Peer AS)
OK
Cancel
353824

Edge Router/Border Leaf Pairs (2 rows)

Edge Router/Border Leaf	IP Address	Interface Name/Port Channel	Profile Name	Type	Partition Utilization
DCI-B-1	37.0.20.21		n7KDcEdgeManualConfigProfile	Edge Router	4/500
DC-B-BL1	37.0.20.12	PC11 <-> PC11	borderLeafUniversalProfile	Border Leaf	2/500
DC-B-BL2	37.0.20.13	PC12 <-> PC12	borderLeafUniversalProfile	Border Leaf	2/500
DCI-B-2	37.0.20.22				

Connect New Border Leaf (BL) to Edge Router

Border Leaf Configuration

* Name:	DC-B-BL1
* IP Address:	37.0.20.12
* Port Channel or Interface Name:	PC11
* Max Number of Partitions:	500
* Default Profile Name:	borderLeafUniversalProfile

[✓] Notify Border Leaf when relevant partitions are changed.

Edge Router Configuration

* Name:	DCI-B-1
* IP Address:	37.0.20.21
* Port Channel or Interface Name:	PC11
* VLAN Range:	2-501
* Default Profile Name:	n7KDcEdgeManualConfigPro

Profile Parameters

Required 'vrfName' format is 'organizationName:partitionName'

ifNameEth:	
vlanId:	
vrfName:	
ipAddress:	10.4.4.1
netMaskLength:	24
ipv6Address:	10:4:4:4::1
prefixLength:	64

Profile Parameters

Required 'vrfName' format is 'organizationName:partitionName'

ifNamePc:	11
vlanId:	
vrfName:	
ipAddress:	10.4.4.2
netMaskLength:	24
ipv6Address:	10:4:4:4::2
prefixLength:	64

Table 7: Figure Legend

1 - Border Leaf Configuration	3 - Edge Router Configuration
2 - Profile Parameters Border Leaf	4 - Profile Parameters Edge Router

1 - Border Leaf Configuration area:

- Port-channel or Interface Name: The interface name on border leaf. This can be a different name than what is on the DC edge box.
- Max Number of Partitions: The maximum number of partitions on the border leaf.
- Default Profile Name: The border leaf profile for extension from the drop-down list.
- Check the Notify Border Leaf when relevant partitions are changed check box so that Cisco Prime DCNM will trigger auto-configuration on border leaf. Uncheck, if auto-configuration is not required and manual copy and paste is preferred.

2 - Profile Parameters area:

- ipAddress and ipv6Address: The interface address and the mask to be used on sub interfaces created on border leaf. This address will be repeated on all sub interfaces for all tenants. Add IPv6 address if IPv6 address family is needed.
- ifNamePc: Leave this field empty, it will be filled by Cisco Prime DCNM.
- ifNameEth: Leave this field empty, it will be filled by Cisco Prime DCNM.
- asn: ASN for border leaf will be obtained by FABRIC ASN setting in POAP.

3 - Edge Router Configuration area:

- Port-channel or Interface Name: Interface name on DC edge router. Leave this field empty, it will be filled by Cisco Prime DCNM.
- VLAN Range: DOT1q range for the interface. A contiguous range is supported.
- Default Profile Name: Choose this configuration profile n7KDcEdgeManualProfile from drop-down list.

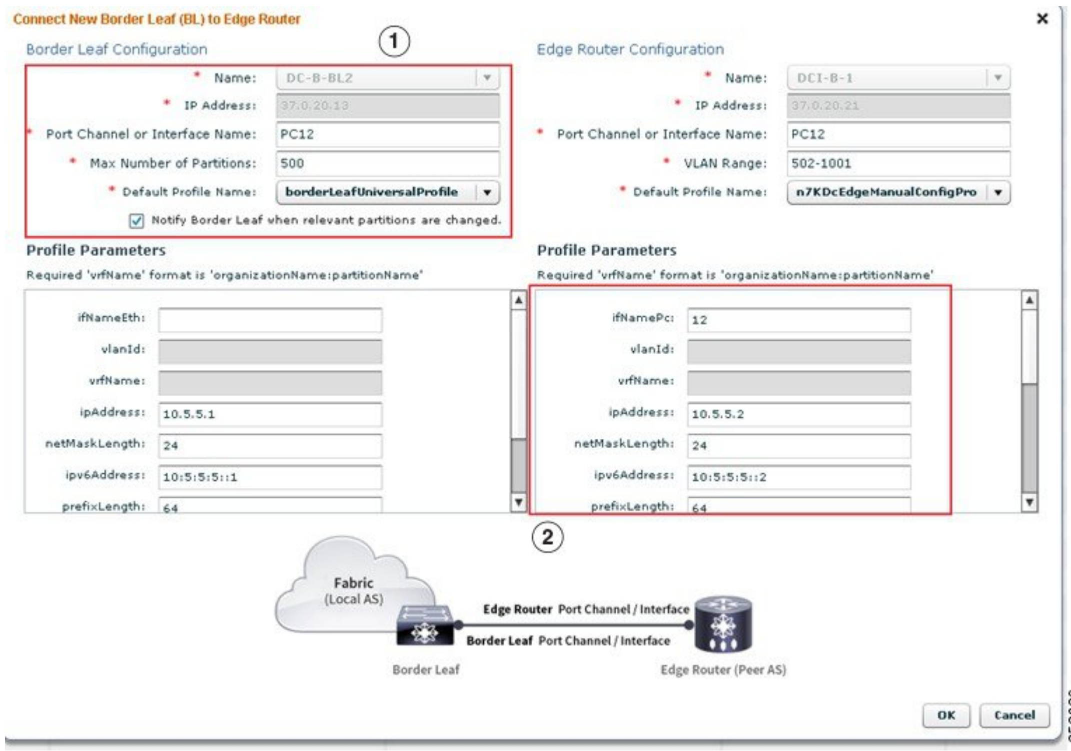
4 - Profile Parameters area:

- ifNamePc: Leave this field empty, it will be filled by Cisco Prime DCNM.
- ipAddress and ipv6Address: The interface address and the mask to be used on sub interfaces created on edge router. This address will be repeated on all sub interfaces for all tenants. Add IPv6 address if IPv6 address family is needed. DC edge end of IP address. Border leaf will automatically use this as peering address for BGP session.
- asn: Set the ASN number for DC edge router here. It will be used to derive the peer ASN on border leaf also.
- peerIpAddress1: BGP neighbor address, same as interface address on border leaf side.
- peerAsn: The Border leaf ASN.
- peerIpAddress2: If building full mesh, then put the interface address of DC-B-BL2. If building a direct topology, leave it blank.

8. If direct pairing is required, then pairing of these two nodes is complete. To Pair more direct paring nodes, nodes if any in your topology, repeat the above steps. To accomplish full mesh topology, go to next step.

9. Click **Add** and select **Border Leaf** to pair DCI-B-1 with DC-BL-2.

Figure 18: Add 2nd Border Leaf



Connect New Border Leaf (BL) to Edge Router

Border Leaf Configuration

- * Name: DC-B-BL2
- * IP Address: 37.0.20.13
- * Port Channel or Interface Name: PC12
- * Max Number of Partitions: 500
- * Default Profile Name: borderLeafUniversalProfile
- ☑ Notify Border Leaf when relevant partitions are changed.

Edge Router Configuration

- * Name: DCI-B-1
- * IP Address: 37.0.20.21
- * Port Channel or Interface Name: PC12
- * VLAN Range: 502-1001
- * Default Profile Name: n7KDcEdgeManualConfigPro

Profile Parameters

Required 'vrfName' format is 'organizationName:partitionName'

- ifNameEth:
- vlanId:
- vrfName:
- ipAddress: 10.5.5.1
- netMaskLength: 24
- ipv6Address: 10:5:5:5::1
- prefixLength: 64

Profile Parameters

Required 'vrfName' format is 'organizationName:partitionName'

- ifNamePc: 12
- vlanId:
- vrfName:
- ipAddress: 10.5.5.2
- netMaskLength: 24
- ipv6Address: 10:5:5:5::2
- prefixLength: 64

1
2
Fabric
(Local AS)
Edge Router Port Channel / Interface
Border Leaf Port Channel / Interface
Border Leaf
Edge Router (Peer AS)
OK
Cancel
353826

Table 8: Figure Legend

1 - Border Leaf Configuration	2 - Profile Parameters Edge Router

10. Repeat the above steps to add the peering of DCI-B-2 with DC-B-BL1 and DCI-B-2 with DC-B-BL2. The full mesh topology is now complete for one pod. If there are more such pods, repeat the above steps to pair them and iterate till topology is complete.

Enable Border Leaf Auto-Configuration

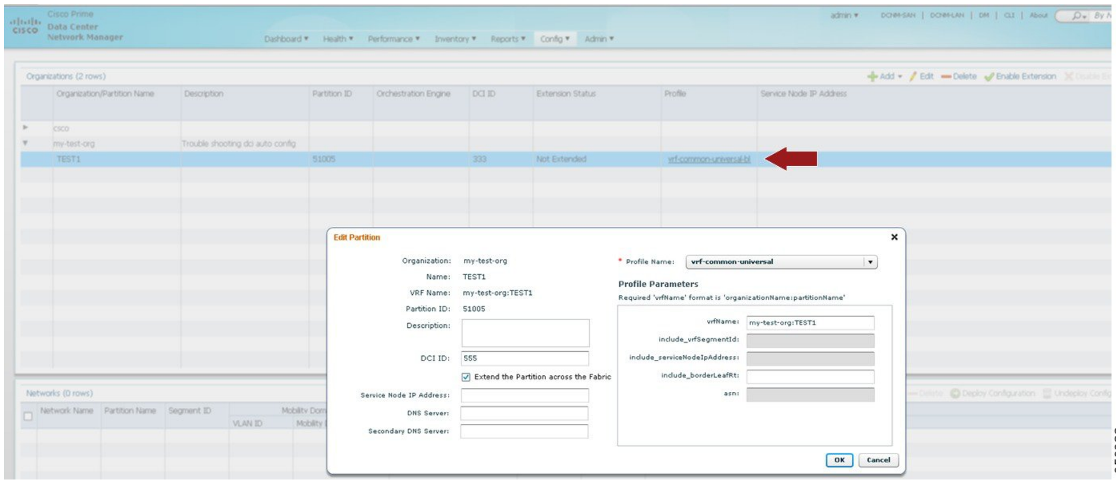
Testing the topology set up by extending from DCNM

Once topology has been set up and feature is globally enabled, auto-configuration is ready. Partition can be extended from Cisco Prime DCNM, UCSD5.2+ and OpenStack 2.0 for Cisco Nexus Fabric. Given below is an example using Cisco Prime DCNM.

Create a test partition and enable it for extension using the following figure at Cisco Prime DCNM. If there are no errors on Cisco Prime DCNM, ssh to the devices where partition is extended and verify that the configuration is fine.

Extend partition at DCNM

Figure 19: Extended DCNM partition


Cisco Prime Data Center Network Manager
Dashboard ▾ Health ▾ Performance ▾ Inventory ▾ Reports ▾ Config ▾ Admin ▾
admin ▾ DCNM-SAN | DCNM-LAN | DM | CLI | About
Add ▾ Edit Delete Enable Extension
Deploy Configuration Undeploy Config
Networks (0 rows)
Network Name Partition Name Segment ID VLAN ID Mobility Dom
353829

Organizations (2 rows)

	Organization/Partition Name	Description	Partition ID	Orchestration Engine	DCI ID	Extension Status	Profile	Service Node IP Address
▸	cisco							
▾	my-test-org	Trouble shooting dci auto config						
	TEST1		51005		333	Not Extended	vrf-common-universal-bl	

Edit Partition

Organization:	my-test-org
Name:	TEST1
VRF Name:	my-test-org:TEST1
Partition ID:	51005
Description:	
DCI ID:	555
☑ Extend the Partition across the Fabric	
Service Node IP Address:	
DNS Server:	
Secondary DNS Server:	

* Profile Name: vrf-common-universal

Profile Parameters

Required 'vrfName' format is 'organizationName:partitionName'

vrfName:	my-test-org:TEST1
include_vrfSegmentId:	
include_serviceNodeIpAddress:	
include_borderLeafRt:	
asn:	

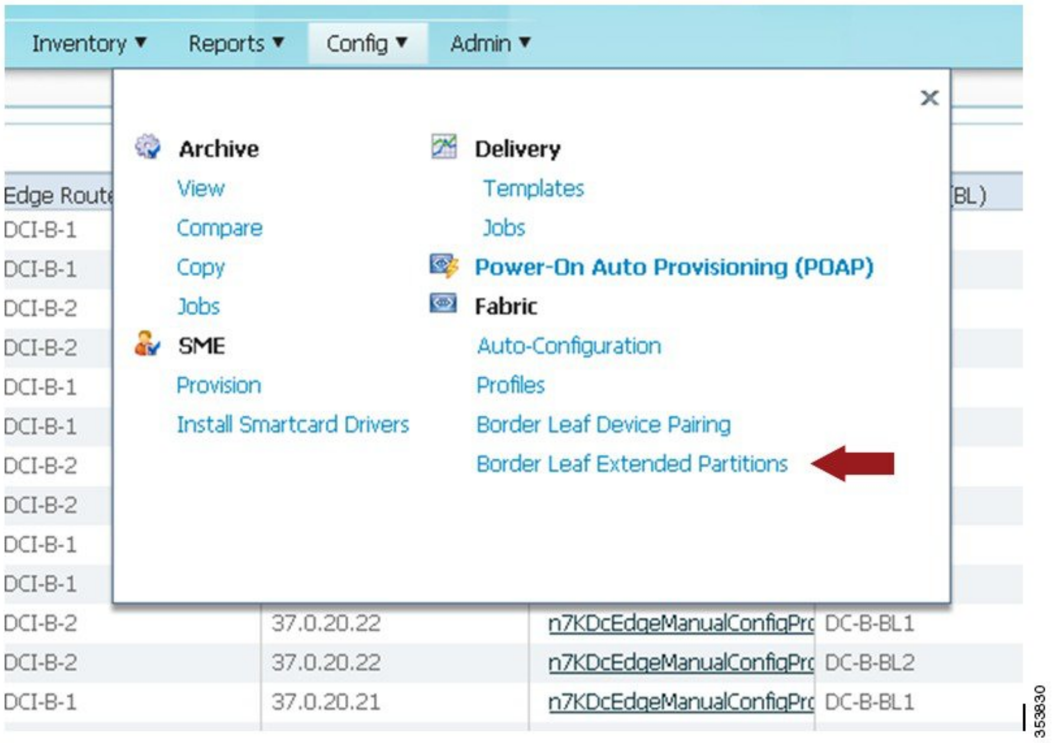
OK Cancel

- Select a profile, value that should be same as what it is in interior leaf nodes. Border leaf uses override command to use a border leaf specific profile.
- Set DCI ID. This has to be same in every fabric for the same tenant. User has responsibility to ensure that its unique and matches on other fabric. Check the **Extend the Partition across the Fabric** check box and click **OK**.

Check which nodes the tenant is extended on

Use the following button to get the screen, which nodes the partition just extended 'my-test-org:TEST1' is deployed.

Figure 20: Border Leaf Extend Partitions

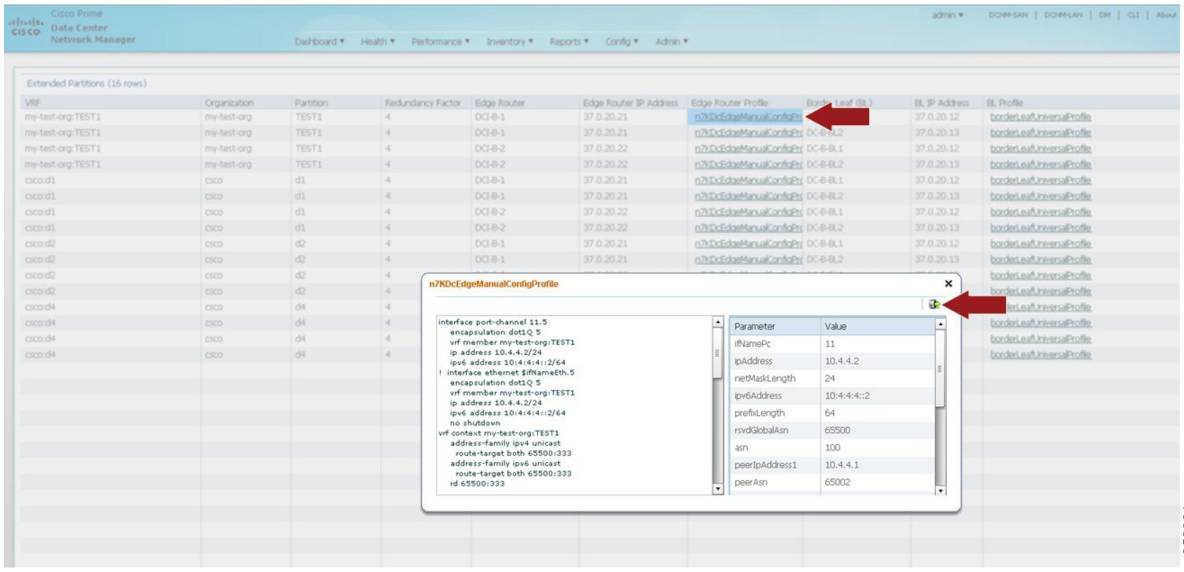


DCI-B-1			
DCI-B-1			
DCI-B-2			
DCI-B-2			
DCI-B-1			
DCI-B-1			
DCI-B-2			
DCI-B-2			
DCI-B-1			
DCI-B-1			
DCI-B-2	37.0.20.22	n7KDcEdgeManualConfigPrc	DC-B-BL1
DCI-B-2	37.0.20.22	n7KDcEdgeManualConfigPrc	DC-B-BL2
DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPrc	DC-B-BL1

Manually configurations for DC Edge devices

Border leaf nodes are fully auto-configured. Verify the configuration on the border leaf by connecting to device and displaying the VRF configuration. DCNM generates the configuration at DCNM for all the extended partitions and all the nodes. For DCI-B-1 and DCI-B-2 by clicking the profile name next to the device pairing. For example, for the DCI-B-1 and DC-B-BL1 pairing, click the n7KDcEdgeManualProfile for DCI-B-1 configuration and then save the generated configuration to a file or directly copy from the pop-up window and paste to configuration shell of the device. Repeat this for every Cisco Nexus 7000 Series Switches.

Figure 21: Configuration Detail Extended Partitions



Extended Partitions (16 rows)

VRF	Organization	Partition	Redundancy Factor	Edge Router	Edge Router IP Address	Edge Router Profile	Border Leaf (BL)	BL IP Address	BL Profile
my-test-org:TEST1	my-test-org	TEST1	4	DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPr	[illegible]	37.0.20.12	borderLeafUniversalProfile
my-test-org:TEST1	my-test-org	TEST1	4	DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPr	DC-B-BL2	37.0.20.13	borderLeafUniversalProfile
my-test-org:TEST1	my-test-org	TEST1	4	DCI-B-2	37.0.20.22	n7KDcEdgeManualConfigPr	DC-B-BL1	37.0.20.12	borderLeafUniversalProfile
my-test-org:TEST1	my-test-org	TEST1	4	DCI-B-2	37.0.20.22	n7KDcEdgeManualConfigPr	DC-B-BL2	37.0.20.13	borderLeafUniversalProfile
csco:d1	csco	d1	4	DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPr	DC-B-BL1	37.0.20.12	borderLeafUniversalProfile
csco:d1	csco	d1	4	DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPr	DC-B-BL2	37.0.20.13	borderLeafUniversalProfile
csco:d1	csco	d1	4	DCI-B-2	37.0.20.22	n7KDcEdgeManualConfigPr	DC-B-BL1	37.0.20.12	borderLeafUniversalProfile
csco:d1	csco	d1	4	DCI-B-2	37.0.20.22	n7KDcEdgeManualConfigPr	DC-B-BL2	37.0.20.13	borderLeafUniversalProfile
csco:d2	csco	d2	4	DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPr	DC-B-BL1	37.0.20.12	borderLeafUniversalProfile
csco:d2	csco	d2	4	DCI-B-1	37.0.20.21	n7KDcEdgeManualConfigPr	DC-B-BL2	37.0.20.13	borderLeafUniversalProfile
csco:d2	csco	d2	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	borderLeafUniversalProfile
csco:d2	csco	d2	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	borderLeafUniversalProfile
csco:d4	csco	d4	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	borderLeafUniversalProfile
csco:d4	csco	d4	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	borderLeafUniversalProfile
csco:d4	csco	d4	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	borderLeafUniversalProfile
csco:d4	csco	d4	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	borderLeafUniversalProfile

n7KDcEdgeManualConfigProfile

```
interface port-channel 11.5
   encapsulation dot1Q 5
   vrf member my-test-org:TEST1
   ip address 10.4.4.2/24
   ipv6 address 10:4:4:4::2/64
!  interface ethernet $ifNameEth.5
   encapsulation dot1Q 5
   vrf member my-test-org:TEST1
   ip address 10.4.4.2/24
   ipv6 address 10:4:4:4::2/64
   no shutdown
vrf context my-test-org:TEST1
   address-family ipv4 unicast
    route-target both 65500:333
   address-family ipv6 unicast
    route-target both 65500:333
   rd 65500:333
```

Parameter	Value
ifNamePc	11
ipAddress	10.4.4.2
netMaskLength	24
ipv6Address	10:4:4:4::2
prefixLength	64
rsvdGlobalAsn	65500
asn	100
peerIpAddress1	10.4.4.1
peerAsn	65002

Example configuration generated by DCNM for N7K DC edge router

For DCI-B1 router, for the DC-B-BL1 pairing for partition my-test-org: TEST1

```
interface port-channel 11.5
    encapsulation dot1Q 5
    vrf member my-test-org:TEST1
    ip address 10.4.4.2/24
    ipv6 address 10:4:4:4::2/64

!  interface ethernet $ifNameEth.5
    encapsulation dot1Q 5
    vrf member my-test-org:TEST1
    ip address 10.4.4.2/24
    ipv6 address 10:4:4:4::2/64
    no shutdown

vrf context my-test-org:TEST1
    address-family ipv4 unicast
      route-target both 65500:555
    address-family ipv6 unicast
      route-target both 65500:555
    rd 65500:555

router bgp 100

    vrf my-test-org:TEST1
address-family ipv4 unicast
      maximum-paths 2
      maximum-paths ibgp 2
      additional-paths send
      additional-paths receive
      additional-paths selection route-map ALL-PATHS
address-family ipv6 unicast
      maximum-paths 2
      maximum-paths ibgp 2
      additional-paths send
```

```
additional-paths receive
additional-paths selection route-map ALL-PATHS
!keeping both neighbors as it is not predictable which neighbor will be populated
!per each link
neighbor 10.4.4.1 remote-as 65002
  address-family ipv4 unicast
    default-originate
    send-community both
neighbor 10.5.5.1 remote-as 65002
  address-family ipv4 unicast
    default-originate
    send-community both
neighbor 10:4:4:4::1 remote-as 65002
  address-family ipv6 unicast
    default-originate
    send-community both
neighbor 10:5:5:5::1 remote-as 65002
  address-family ipv6 unicast
    default-originate
    send-community both
```

Border Leaf required Configurations specific to this feature

If the device is loaded by POAP, these commands are already present.

- VRF override profile command: This command over rides the partition profile with the one configured below.

```
fabric database override-vrf-profile vrf-common-universal-bl
```

- LDAP configuration for bl-dci table, rest of LDAP is same as interior leaf nodes. Configuration below shows optional redundant LDAP.

```
fabric database type bl-dci
    server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
    db-security user cn=reader,dc=cisco,dc=com password1

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
    db-table ou=bl-dcis,dc=cisco,dc=com
   db-security user cn=reader,dc=cisco,dc=com password1
```

Limitations

The following restrictions are specific to Cisco Prime DCNM 7.1(1) and Cisco NX-OS 7.1.

- Maximum number of neighbors is two
- Profiles limited to Layer-3 unicast scenarios in this release

Global Enable

Set the following values in the Global Parameters.

BGP Route-Target AS #:

If it is intended to interconnect multiple fabrics together, this value has to be same on all DCNMs in all fabrics. It is a 2-byte unsigned integer value (1 – 65535) used to construct the RT used between DC edge devices.

The route-target is constructed by appending the DCI to this value in the following way:

RT = BGP Route-Target AS # : DCI ID

The DCI ID (4-byte unsigned integer) is a user entered value when creating/modifying a partition. It is your responsibility to ensure that same DCI ID is same in different fabrics when extending a partition across fabrics.

Note This value is only used on DC edge device. If you are not using auto-configuration feature including the copy and paste on DC edge device then this value is not relevant but must be provided for this feature to work.

For example, if BGP Route-Target AS #: 65500 and DCI ID is 555 then the following RT is generated for DC edge router:

```
vrf context my-test-org:TEST1
    address-family ipv4 unicast
      route-target both 65500:555
    address-family ipv6 unicast
      route-target both 65500:555
    rd 65500:555
```

Redundancy factor

The number of border leafs one tenant is provisioned on. If a border leaf is chosen then,

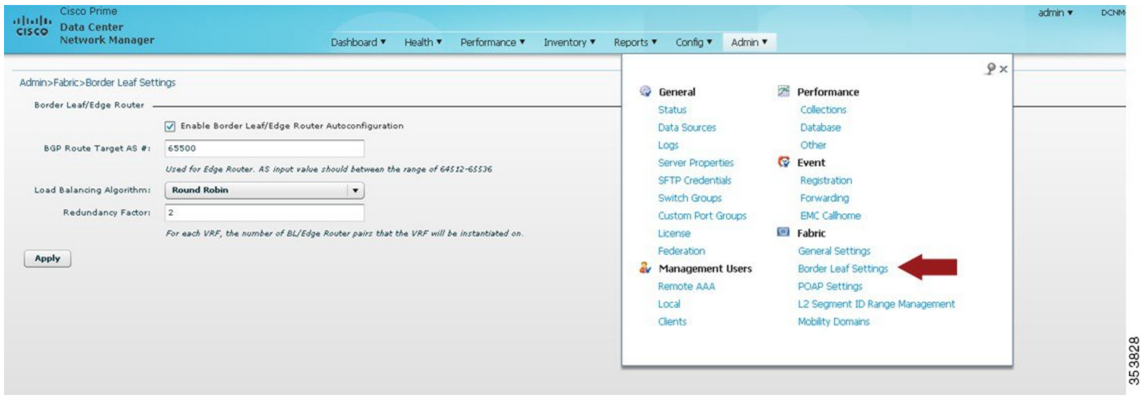
1. If it is in a vPC pair with another border leaf then its vPC neighbor is also chosen, even if redundancy factor is 1. In presence of vPC pairs, number of border leafs a partition is deployed is always an even number. Without vPC pairs the number of border leaf a tenant is deployed is exactly equal to redundancy factor as long as capacity permits.
2. Once a border leaf is chosen then all its DC edge neighbors are also chosen. Maximum of two neighbors per node are allowed.

Note For vPC pair in vinci leaf, We recommend to configure vpc with **dual-active exclude interface-vlan-bridge-domain** *allow-vlans* and make sure SVI/BDI do not go down when vPC peer link goes down. This will prevent duplicate packet forwarding in TF mode.

Load Sharing Algorithm

Currently the only algorithm supported is round robin. In future, users may be able to provide their own algorithm. As the name suggests, this algorithm tries to round robin partition deployment over the available border leaf resources, maintaining the redundancy factor.

Figure 22: Border Leaf Settings



Admin>Fabric>Border Leaf Settings

Border Leaf/Edge Router

[✓] Enable Border Leaf/Edge Router Autoconfiguration

BGP Route Target AS #: 65500

Used for Edge Router. AS input value should between the range of 64512-65536

Load Balancing Algorithm: Round Robin

Redundancy Factor: 2

For each VRF, the number of BL/Edge Router pairs that the VRF will be instantiated on.

Apply

- Click **Admin** and select **Border Leaf Settings** from Fabric.
- After setting the BGP route-target AS and redundancy factor, check the **Enable Border Leaf/Edge Router Autoconfiguration** check box to enable the feature globally.
- Click **Apply**. The system is ready for auto-configuration.

CHAPTER 12

Unicast Forwarding

Feature Information for Unicast Forwarding

Table 9: Feature Information for Unicast Forwarding

Feature	Releases	Feature Information
Forwarding	7.2(0)D1(1)	Included a new chapter on *Unicast Forwarding*. Enhanced Forwarding functionality.

Unicast Forwarding

The Enhanced Forwarding functionality consists of the following:

- **End Host Registration and Tracking**: Leaf switch will intercept the ARP and DHCP packets and learns the IP to MAC binding for the host. In the case of virtualized environments, VDP may be used to track the VM moves.
- **End Host Route Distribution using BGP**: MP-BGP is used within the Vinci fabric to distribute the IP reachability information across the leaf switches in the fabric. Route reflectors will be used for scalability purposes.
- **ARP and IPv6 ND Proxy**: One of the goals of Vinci Enhanced forwarding is to eliminate flooding across the fabric. For this purpose, every leaf switch will act as a ARP/IPv6 ND proxy for all the hosts connected to the leaf switch.
- **Anycast Default Gateway**: Leaf switches act as the default gateway for routing across subnets. All the leafs that handle a given subnet are configured with the same anycast IP address and MAC address.
- **Forwarding Behavior at Leaf switches**: A fabric VLAN is allocated per tenant VRF in the Vinci fabric. BGP advertises the reachability of hosts attached to remote leafs to all the leaf nodes. Ingress leaf switch routes the traffic from the tenant VLAN to the egress leaf switch where the destination host is attached via the fabric VLAN. Egress leaf switch routes the traffic back to the tenant VLAN. This allow the platforms to optimize by scaling the FIB host route table without scaling the MAC address table.

- **Routing within the subnet**: Leaf switch responds with the Anycast Default Gateway MAC to the ARP requests for hosts within the same subnet.
- **Selective FIB download**: Routes are installed by the FIB in hardware based on conversations to optimize the usage of the FIB entries in the platform. The FIB entries will also be aged out if the conversation between the hosts ceases to exist.

In the traditional forwarding with dynamic subnets mode, the functionality for end host registration and tracking, end host distribution via BGP and Selective FIB download remains the same as enhanced forwarding. The difference in behavior is as follows:

- **Bridging within the subnet**: ARP requests are flooded in the subnet across the fabric and traffic within the subnet is bridged. This is predominantly to support "unknown unicast" cases where the remote host never speaks. In the case of Enhanced forwarding, IP traffic to unknown destination is dropped.
- **Anycast Default Gateway**: This solution is still applicable to traffic across subnets. If the host route for the destination host does not exist, traffic is punted to SUP for glean processing. In the case of enhanced forwarding, it is expected that the remote host route is announced via BGP. In the traditional forwarding mode, the glean process will trigger a ARP request across the fabric for the remote host to deal with the host that never speaks. While the ARP request will be sent with a source MAC = Ingress Leaf Router MAC, the ARP payload itself will contain the IP address = Anycast Default Gateway IP and MAC = Anycast Default Gateway MAC. Most hosts will respond to the ARP request with a ARP response destined to the MAC in the ARP payload and in this case, it will be destined to the Anycast Default Gateway MAC. The egress leaf will receive this ARP response and announce the remote host via BGP. It is expected that the ARP request in the ingress leaf switch will eventually timeout.
- **Flooding**: As mentioned in the previous two cases, flooding is allowed in traditional forwarding.

CHAPTER 13

Multicast Forwarding

Feature Information for Multicast Forwarding

Table 10: Feature Information for Multicast Forwarding

Feature	Releases	Feature Information
Border leaf conversation Learning	7.2(0)N1(1)	Included a new section on *Conversational learning on Border Leaf*. Enable conversational learning on all leaf nodes.

Multicast Forwarding

Multicast forwarding allows the hardware engine to forward IP packets, instead of having multicast traffic sent to the line card CPU for slower path processing.

- PIM SM, SSM and Bidir are supported in DFA.
- Any source connected to a leaf will route traffic to local receivers and also route to fabric SVI if there are any interested receivers only.
- All receivers expecting traffic from remote source will receive routed traffic through fabric SVI.

- RP’s can be configured on border leafs or any router outside of the fabric but should be reachable via border leafs only.

Enabling Fabric Multicast

Before you begin

- Feature Fabric Forwarding should be enabled
- Ensure fabric unicast is configured and unicast reachability and BGP peering is successfully established
- For multi-tenancy lite version system fabric dynamic VLANs must be configured. For multi-tenancy full version system fabric bridge-domain must be configured.
- Ensure RP is configured on border leafs or any external router connected or reachable via border leaf only
- Ensure RP reachability is from all leaf nodes and border leafs

SUMMARY STEPS

1. **configure terminal**
2. **install feature-set fabric**
3. **feature-set fabric**
4. **feature fabric forwarding**
5. **[no] feature pim**
6. **[no] feature fabric multicast**
7. **[no] ip multicast fabric-forwarding**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**install feature-set fabric** **Example:** `switch(config)# install feature-set fabric`	Install feature fabric.
Step 3	**feature-set fabric** **Example:** `switch(config)# feature-set fabric`	Install feature fabric.
Step 4	**feature fabric forwarding** **Example:** `switch(config)# feature fabric forwarding`	Enable/Disable Fabric Forwarding Protocol: Host Mobility Manager (HMM).

	Command or Action	Purpose
Step 5	**[no] feature pim** **Example:** `switch(config)# feature pim`	Enables PIM. **Note** No need to enable ip pim sparse-mode under SVI/BDI configuration.
Step 6	**[no] feature fabric multicast** **Example:** `switch(config)# feature fabric multicast`	Enable/Disable NGMVPN features.
Step 7	**[no] ip multicast fabric-forwarding** **Example:** `switch(config)# ip multicast fabric-forwarding`	Enables multicast enhanced fabric forwarding.

Configuring Host VLANs

PIM is enabled automatically and operates in passive mode on a server facing SVI/BDI.

In order to enable IGMPv3, the following configuration must be done on the server facing SVI/BDI.

SUMMARY STEPS

1. **ip igmp version 3**
2. **show ip pim**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**ip igmp version 3**
	Example:
	Multi-tenancy lite version
	`Vlan 100`
	` Mode fabricpath`
	
	`Int vlan 100`
	`   Ip address 100.1.1.1/24`
	`   Ip igmp version 3   // use this config for PIM`
	` SSM`
	Example:
	Multi-tenancy full version
	`Bridge-domain 100`
	
	`Int bdi 100`
	`   Ip address 100.1.1.1/24`
	`   Ip igmp version 3   // use this config for PIM`
		Configures ip igmp for SSM.

	Command or Action	Purpose
	`SSM`	

Step 2	**show ip pim**
	
	Example:
	Multi-tenancy lite version
	
	`Device# show ip pim interface vlan 77`
	`PIM Interface Status for VRF "ORG:RED"`
	`Vlan77, Interface status:`
	`protocol-up/link-up/admin-up`
	`IP address: 10.1.1.2 IP subnet: 10.1.1.0/24`
	`PIM DR: 10.1.1.2, DR's priority: 1`
	`PIM neighbor count: 0`
	`PIM hello interval: 30 secs, next hello sent in:`
	`0.000000`
	`PIM neighbor holdtime: 105 secs`
	`PIM configured DR priority: 1`
	`PIM configured DR delay: 3 secs`
	`PIM border interface: no`
	`PIM GenID sent in Hellos: 0x147bad9d`
	`PIM Hello MD5-AH Authentication: disabled`
	`PIM Neighbor policy: none configured`
	`PIM Join-Prune inbound policy: none configured`
	`PIM Join-Prune outbound policy: none configured`
	`PIM Join-Prune interval: 1 minutes`
	`PIM Join-Prune next sending: 1 minutes`
	`PIM BFD enabled: no`
	`PIM passive interface: yes`
	`PIM VPC SVI: no`
	`PIM Auto Enabled: yes`
	`PIM Interface Statistics, last reset: never`
	`General (sent/received):`
	`Hellos: 0/0 (early: 0), JPs: 0/0, Asserts: 0/0`
	
	Example:
	Multi-tenancy full version
	
	`Device# show ip pim interface bdi 77`
	`PIM Interface Status for VRF "ORG:RED"`
	`bd77, Interface status:`
	`protocol-up/link-up/admin-up`
	`IP address: 10.1.1.2 IP subnet: 10.1.1.0/24`
	`PIM DR: 10.1.1.2, DR's priority: 1`
	`PIM neighbor count: 0`
	`PIM hello interval: 30 secs, next hello sent in:`
	`0.000000`
	`PIM neighbor holdtime: 105 secs`
	`PIM configured DR priority: 1`
	`PIM configured DR delay: 3 secs`
	`PIM border interface: no`
	`PIM GenID sent in Hellos: 0x147bad9d`
	`PIM Hello MD5-AH Authentication: disabled`
	`PIM Neighbor policy: none configured`
	`PIM Join-Prune inbound policy: none configured`
	`PIM Join-Prune outbound policy: none configured`
	`PIM Join-Prune interval: 1 minutes`
	`PIM Join-Prune next sending: 1 minutes`
	`PIM BFD enabled: no`
		Verifies PIM in passive state.

	Command or Action	Purpose

	`PIM passive interface: yes`
	`PIM VPC BDI: no`
	`PIM Auto Enabled: yes`
	`PIM Interface Statistics, last reset: never`
	`General (sent/received):`
	`Hellos: 0/0 (early: 0), JPs: 0/0, Asserts: 0/0`	

Configuring BGP MVPN

For Spine configured as route reflector ensure the following configurations have been added under MVPN address family.

SUMMARY STEPS

1. **configure terminal**
2. **router bgp asn**
3. **[no] address-family**
4. **[no] neighbor** *<ip address>***remote-as***<AS number>*
5. **[no] address-family**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**router bgp asn** **Example:** `switch(config)# router bgp asn`	Creates BGP instance.

Step 3	**[no] address-family**
	Example:
	`switch(config-router)# address-family ipv4 mvpn`
	`switch(config-router)# additional-paths send`
	`switch(config-router)# additional-paths selection route-map RMAP`
	`switch(config-router)#`
	`switch(config-router)# address-family ipv6 mvpn`
	`switch(config-router)# additional-paths send`
	`switch(config-router)# additional-paths selection route-map RMAP`
	
	`//Note: RMAP is a route map that can be configured as below:`
	`switch(config-router)# route-map RMAP permit 10`
	`switch(config-router)# set path-selection all advertise`
		Configures multicast VPN.

	Command or Action	Purpose
Step 4	**[no] neighbor** *<ip address>***remote-as***<AS number>* **Example:** `switch(config)# neighbor 10.1.1.2 remote-as 65000`	Configures a BGP peer.

Step 5	**[no] address-family**
	
	Example:
	`switch(config-router)# address-family ipv4 mvpn`
	`switch(config-router)# send-community extended`
	`switch(config-router)# route-reflector-client`
	`switch(config-router)# address-family ipv6 mvpn`
	`switch(config-router)# send-community extended`
	`switch(config-router)# route-reflector-client`
		Send extended community attribute to BGP peer. Enable route reflector on spine.

Verifying BGP MVPN configuration

This example shows the BGP neighbors currently peered. Verify the BGP configs if you do not get an output:

```
switch(config)# show bgp ipv4 mvpn summary

BGP summary information for VRF default, address family IPv4 MVPN
BGP router identifier 10.1.1.1, local AS number 10
BGP table version is 62, IPv4 MVPN config peers 1, capable peers 1
6 network entries and 10 paths using 936 bytes of memory
BGP attribute entries [8/1120], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [2/8]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.1.1.2        4    10   32823   32787       62    0    0     3w1d 4
```

Ensure additional-paths capability is exchanged and active on the Spine-Leaf or Spine-Border Leaf peering

```
switch(config)# show bgp ipv4 mvpn neighbors | grep -A 4 Additional

Additional Paths capability: advertised received
  Additional Paths Capability Parameters:
  Receive capability advertised to Peer for AF:
    IPv4 Unicast  IPv6 Unicast  IPv4 MVPN  IPv6 MVPN
  Send capability received from Peer for AF:
    IPv4 Unicast  IPv6 Unicast  IPv4 MVPN  IPv6 MVPN
```

Border Leaf Specific Configurations

PIM RP on Border leaf

In DFA, it is required that an RP is configured on the border leaf for redundancy and load balancing. An RP on a border leaf can be configured for the desired mode (bidir / ASM) as described in the Cisco Nexus 5000 Series Switches, see Multicast Routing Configuration Guide.

The following modes of RP configuration are supported:

- Static RP
- Anycast RP
- BSR mechanism
- Auto RP

SUMMARY STEPS

1. **Show fabric multicast ipv4 ssm-range vrf all**
2. **Show fabric multicast ipv4 mroute vrf all**
3. **show ip pim group-range vrf all**
4. **show fabric multicast { ipv4 | ipv6 } rp-grange vrf all**
5. **show bgp ipv4 mvpn**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Show fabric multicast ipv4 ssm-range vrf all** **Example:** `switch(config)# Show fabric multicast ipv4 ssm-range vrf all`	Verify that ssm range is configured correctly in fabric multicast.
Step 2	**Show fabric multicast ipv4 mroute vrf all** **Example:** `switch(config)# Show fabric multicast ipv4 mroute vrf all`	Verify multicast routes are communicated correctly to fabric_mcast.
Step 3	**show ip pim group-range vrf all** **Example:** `switch(config)# Show ip pim group-range vrf all`	Verify RP is configured correctly on the border leaf. Ensure RP is configured correctly in PIM.

Step 4	**show fabric multicast { ipv4 | ipv6 } rp-grange vrf all**
	Example:
	`switch(config)# show fabric multicast ipv4 rp-grange vrf all`
	`switch(config)# show fabric multicast ipv6 rp-grange vrf all`
		Verify RP is configured correctly on the border leaf. Ensure RP information is communicated correctly to fabric_mcast.

Step 5	**show bgp ipv4 mvpn** **Example:** `switch(config)# Show bgp ipv4 mvpn`	Ensure sure RP information is communicated correctly to BGP. Check for Group-to-RP mapping.

Various PIM RP Placements

RP Configured on the Border Leaf

- RP can be configured on the border leaf.
- There can be multiple border leafs for redundancy. Anycast RP can be configured between multiple RP across multiple border leafs.

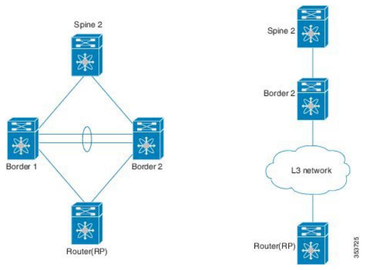
RP Configured Outside of Border Leaf

- RP can be configured outside the fabric on a router connected with Layer-3 links (physical port, subinterfaces on Layer-3 port channel) to the border leaf or border leafs in a vPC+.

Note Use of legacy SVI on border leaf nodes in a vPC+ setup is not supported, as this will result in multicast traffic duplication within the DFA fabric.

- Any IGP like OSPF can be run to ensure reachability to the external RP.

Figure 23: RP configured outside of Border Leaf



Various PIM configurations supported

- Static, BSR, Anycast RP and Auto RP can be configured on the router.
- Ensure unicast reachability to RP from the border leafs with any routing protocols like OSPF.

Sample Border Leaf

Sample border leaf BGP configuration.

```
Device# show running-config bgp

!Time: Wed Feb 28 20:45:54 2001

version 7.1(0)N1(1)
feature bgp

router bgp 65101
  router-id 10.1.1.2
  fabric-soo 65101:1
```

```
address-family ipv4 unicast
  redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
  maximum-paths ibgp 2
  nexthop trigger-delay critical 250 non-critical 10000
  nexthop route-map bgp_next_hop_filter
  additional-paths receive
  additional-paths selection route-map ALL-PATHS
address-family ipv6 unicast
  redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
  maximum-paths ibgp 2
  nexthop trigger-delay critical 250 non-critical 10000
  additional-paths receive
  additional-paths selection route-map ALL-PATHS
  address-family vpnv4 unicast
  nexthop trigger-delay critical 250 non-critical 10000
  additional-paths receive
address-family vpnv6 unicast
  nexthop trigger-delay critical 250 non-critical 10000
  additional-paths receive
address-family ipv4 mvpn
  nexthop trigger-delay critical 250 non-critical 10000
  additional-paths receive
  additional-paths selection route-map ALL-PATHS
address-family ipv6 mvpn
  nexthop trigger-delay critical 250 non-critical 10000
  additional-paths receive
  additional-paths selection route-map ALL-PATHS
  neighbor 10.1.1.2 remote-as 65101
address-family ipv4 unicast
  send-community both
  route-map deny-default-route in
  next-hop-self
address-family ipv6 unicast
  send-community both
address-family vpnv4 unicast
  send-community extended
  route-map deny-default-route in
address-family vpnv6 unicast
  send-community extended
address-family ipv4 mvpn
  send-community both
address-family ipv6 mvpn
  send-community both
```

Configuring Legacy SVI

On Border Leafs:

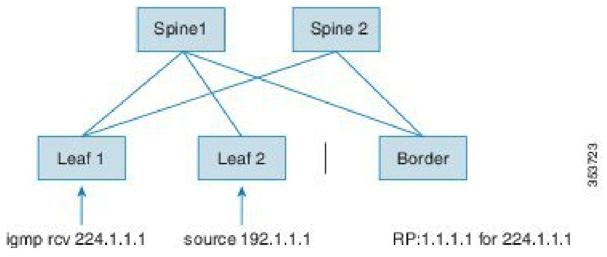
- If an SVI over FabricPath VLAN is explicitly configured with PIM, that is configured with “ip pim sparse-mode” then such SVIs become legacy SVIs.
- Multicast forwarding using legacy SVI is same as in non-DFA environment with PIM enabled. For example, PIM in active mode, PIM hellos exchanged between multicast routers.
- Legacy SVI should be used to communicate with services outside fabric or with non-DFA routers like Cisco Nexus 5000 Series Switches or Services (for example, Firewall and Loadbalancer) .
- Use of legacy SVI on border leaf nodes in a vPC+ setup is not supported, as this will result in multicast traffic duplication within the DFA fabric.

On Leaf Nodes:

- Any VLAN without SVI is considered as a legacy VLAN.

Traffic Flow in a Spine Leaf Topology

Figure 24: Traffic flow in a simple spine leaf topology



Note The default drop route value for Layer-3 unregistered packets is 224/4. Ensure not to configure bidir with prefix of 224/4 as this may result in an unexpected behavior.

Multi-tenancy lite version

Consider the receiver VLAN 20 is on Leaf 1, the source VLAN 30 on Leaf 2 and the RP 1.1.1.1 for group 224.1.1.1 on the border leaf:

1. Consider Leaf 1 receives joins on VLAN 20, this triggers a join through BGP/NGMVPN (fabric_mcast process) towards the fabric.

   The mroute state on Leaf1 is shown in the following example:

   ```
   (*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
     Incoming interface: Vlan3001, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1)
      Vlan20, uptime: 00:35:19, igmp
   ```

2. Border leaf listens to this join from NGMVPN imported through BGP and simulate a PIM registration behavior. Note that joins are exported through BGP/NGMVPN and not through PIM.

   The mroute state on border leaf is shown in the following example:

   ```
   (*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
     Incoming interface: Loopback0, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1)
     Vlan3001, uptime: 00:35:19, igmp
   ```

3. In Leaf 2, when source traffic is received for a particular group on VLAN 30, NGMPVN would identify that there is an interested receiver in the fabric and adds fabric SVI as the OIF.

   ```
   (192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, fabric_mcast mrib pim ip
     Incoming interface: Vlan30, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1) (Fabric OIF)
       Vlan3001, uptime: 00:50:19, mrib
   ```

   Once the traffic is sent to the fabric, interested receivers in remote leafs would accept it and route to all the local receivers. In this example Leaf 1 receives the traffic from Fabric SVI and route it to the local receiver VLAN 20.

The states of Leaf1 is shown in the following example:

```
(*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
  Incoming interface: Vlan3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
    Vlan20, uptime: 00:35:19, igmp

(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, igmp ip pim
  Incoming interface: Vlan3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
    Vlan20, uptime: 00:50:19, mrib
```

Traffic coming from fabric is tagged with the VNI and hence picks the right VRF and route is based on the VRF to VNI and fabric VLAN mapping.

The state of Leaf2 is shown in the following example:

```
(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, fabric_mcast mrib pim ip
  Incoming interface: Vlan30, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1) (Fabric OIF)
    Vlan3001, uptime: 00:50:19, mrib
```

The states of Border Leaf is shown in the following example:

```
(*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
  Incoming interface: Loopback0, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
    Vlan20, uptime: 00:35:19, igmp

(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, igmp ip pim
  Incoming interface: Vlan3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1) (Fabric OIF)
    Vlan3001, uptime: 00:50:19, mrib
```

Multi-tenancy full version

Consider the receiver bridge-domain 20 is on Leaf 1, the source bridge-domain 30 on Leaf 2 and the RP 1.1.1.1 for group 224.1.1.1 on the border leaf:

1. Consider Leaf 1 receives joins on bridge-domain 20, this triggers a join through BGP/NGMVPN (fabric_mcast process) towards the fabric.

   The mroute state on Leaf1 is shown in the following example:

```
(*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
  Incoming interface: bd3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
   bridge-domain20, uptime: 00:35:19, igmp
```

2. Border leaf listens to this join from NGMVPN imported through BGP and simulate a PIM registration behavior. Note that joins are exported through BGP/NGMVPN and not through PIM.

   The mroute state on border leaf is shown in the following example:

```
(*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
  Incoming interface: Loopback0, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
  bd3001, uptime: 00:35:19, igmp
```

3. In Leaf 2, when source traffic is received for a particular group on bridge-domain 30, NGMPVN would identify that there is an interested receiver in the fabric and adds fabric SVI as the OIF.

```
(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, fabric_mcast mrib pim ip
  Incoming interface: bridge-domain30, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1) (Fabric OIF)
    bd3001, uptime: 00:50:19, mrib
```

Once the traffic is sent to the fabric, interested receivers in remote leafs would accept it and route to all the local receivers. In this example Leaf 1 receives the traffic from Fabric SVI and route it to the local receiver VLAN 20.

The states of Leaf1 is shown in the following example:

```
(*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
  Incoming interface: bd3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
    bridge-domain20, uptime: 00:35:19, igmp

(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, igmp ip pim
  Incoming interface: bd3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
    bridge-domain20, uptime: 00:50:19, mrib
```

Traffic coming from fabric is tagged with the VNI and hence picks the right VRF and route is based on the VRF to VNI and fabric VLAN mapping.

The state of Leaf2 is shown in the following example:

```
(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, fabric_mcast mrib pim ip
  Incoming interface: bridge-domain30, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1) (Fabric OIF)
    bd3001, uptime: 00:50:19, mrib
```

The states of Border Leaf is shown in the following example:

```
(*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
  Incoming interface: Loopback0, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1)
    bridge-domain20, uptime: 00:35:19, igmp

(192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, igmp ip pim
  Incoming interface: bd3001, RPF nbr: 10.1.1.4
  Outgoing interface list: (count: 1) (Fabric OIF)
    bd3001, uptime: 00:50:19, mrib
```

Configuring VPC+ Border Leaf

Limitation

- Multiple FabricPath MDTs are currently not supported on VPC+ border leafs. It will be supported only in the future releases.
- Use of legacy SVI on border leaf nodes in a vPC+ setup is not supported, as this will result in multicast traffic duplication within the DFA fabric.

Configure VPC Border Leaf

1. Configure VPC+ between Border 1 and Border 2. Configure similar VPC+ configuration on Border 2 and verify if VPC+ is up and running.
2. Enable enhanced multicast.
3. Ensure BGP peering with the fabric spine from both the border leafs.
4. Configure border leafs.

5. Configure RP in the border Leafs (PIM RP on Border leaf , on page 104).
6. One of the border leaf will be considered if same RP address is configured on both the border leafs in VPC+ based on unicast reachability.

Before you begin

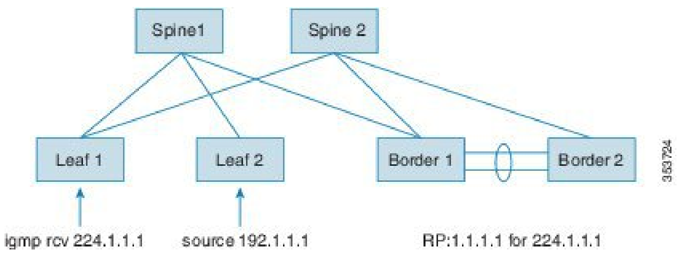
- VPC+ is configured between border routers
- Feature Fabric Forwarding must be enabled
- Ensure fabric unicast is configured and unicast reachability and BGP peering is successfully established
- System fabric dynamic VLANs should be configured
- Enhanced fabric multicast must be enabled

Traffic Flow with Border Leafs in VPC

Overview

Border leafs can be configured in VPC+. VPC+ configuration remains same as configuring routers in VPC+ FabricPath topologies.

Figure 25: Traffic flow with Border Leafs in VPC



Note The default drop route value for Layer-3 unregistered packets is 224/4. Ensure not to configure bidir with prefix of 224/4 as this may result in an unexpected behavior.

Multi-tenancy lite version

1. Consider Leaf 1 receives joins on VLAN 20, this triggers a join through BGP/NGMVPN (fabric_mcast process) towards the fabric.

   The mroute state on Leaf1 is shown in the following example:

   ```
   (*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
     Incoming interface: Vlan3001, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1)
      Vlan20, uptime: 00:35:19, igmp
   ```

2. One of the border leaf listens to this join from NGMVPN imported through BGP and simulate a PIM registration behavior. Note that joins are exported through BGP/NGMVPN and not through PIM.

3. In Leaf 2, when source traffic is received for a particular group on VLAN 30, NGMPVN would identify that there is an interested receiver in the fabric and adds fabric SVI as the OIF.

   ```
   (192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, fabric_mcast mrib pim ip
     Incoming interface: Vlan30, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1) (Fabric OIF)
       Vlan3001, uptime: 00:50:19, mrib
   ```

   Once the traffic is sent to the fabric, interested receivers in remote leaf nodes would accept it and route to all the local receivers. In this example Leaf 1 receives the traffic from fabric SVI and route it to the local receiver VLAN 20.

Multi-tenancy full version

1. Consider Leaf 1 receives joins on bridge-domain 20, this triggers a join through BGP/NGMVPN (fabric_mcast process) towards the fabric.

   The mroute state on Leaf1 is shown in the following example:

   ```
   (*, 224.1.1.1/32), uptime: 00:35:19, igmp ip pim
     Incoming interface: bd3001, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1)
      bridge-domain20, uptime: 00:35:19, igmp
   ```

2. One of the border leaf listens to this join from NGMVPN imported through BGP and simulate a PIM registration behavior. Note that joins are exported through BGP/NGMVPN and not through PIM.

3. In Leaf 2, when source traffic is received for a particular group on VLAN 30, NGMPVN would identify that there is an interested receiver in the fabric and adds BDI as the OIF.

   ```
   (192.1.1.1, 224.1.1.1/32), uptime: 00:50:19, fabric_mcast mrib pim ip
     Incoming interface: bridge-domain30, RPF nbr: 10.1.1.4
     Outgoing interface list: (count: 1) (Fabric OIF)
       bd3001, uptime: 00:50:19, mrib
   ```

   Once the traffic is sent to the fabric, interested receivers in remote leaf nodes would accept it and route to all the local receivers. In this example Leaf 1 receives the traffic from BDI and route it to the local receiver bridge-domain 20.

Conversational learning on Border Leaf

You can enable conversational learning on all leaf nodes by using the **fabric forwarding conversational-learning all** command. For this to work, the subnet needs to be instantiated on the leaf. But in case of a border leaf, this is not true as the border leaf might not have any hosts connected to it. So, the routes will always get installed in forwarding information base (FIB). But border leaf is the point of heavy load in the network and needs to conserve precious forwarding space. In this regard, we can add configuration at the border leaf for each subnet using the **fabric forwarding aggregate-subnet-prefix <>** command.

Another way to prevent having to add this configuration for each subnet is to have border leaf learn about aggregate subnet prefixes from the leaf nodes with special cost community value. The CC value tells the software to not install an ECMP route for this prefix, but to install a glean entry in the FIB, to achieve this, the leaf nodes will need to export the anycast gateway IP into iBGP with a special cost community value as shown below.

To enable Layer-3 conversational learning-based route download into the forwarding information base (FIB), use the **fabric forwarding conversational-learning all** command. And to configure the conversational aging timeout value, use the **fabric forwarding conversational-aging** *timeout* command. For more information on these conversational commands, see Cisco Dynamic Fabric Automation Command Reference.

Note To enable this feature on border leaf, all the internal leaf nodes in the fabric must have a route-map. And ensure that conversational learning is enabled on border leaf.

The following example shows the leaf configuration:

Multi-tenancy lite version

```
route-map direct permit 10
  match tag 12345
  set extcommunity cost pre-bestpath 112 4294967295 transitive

router bgp 100
  router-id 100.1.1.5
  neighbor 100.1.1.1 remote-as 100
    update-source Vlan2
    address-family vpnv4 unicast
      send-community extended
  vrf vpn4
    address-family ipv4 unicast
      redistribute direct route-map direct
```

Multi-tenancy full version

```
route-map direct permit 10
  match tag 12345
  set extcommunity cost pre-bestpath 112 4294967295 transitive

router bgp 100
  router-id 100.1.1.5
  neighbor 100.1.1.1 remote-as 100
    update-source bridge-domain2
    address-family vpnv4 unicast
      send-community extended
  vrf vpn4
    address-family ipv4 unicast
      redistribute direct route-map direct
```

To prevent all direct routes from being distributed, you must tag the routes and export only the ones with the tag:

```
interface Vlan40
  no shutdown
  vrf member vpn4
  ip address 40.1.1.1/8 tag 12345
  fabric forwarding mode proxy-gateway
```

The following example shows how to configure per-vrf label allocation on border leaf:

```
router bgp 100
  vrf csco:vrf-11
    address-family ipv4 unicast
      maximum-paths 32
      maximum-paths ibgp 32
       label-allocation-mode per-vrf
    address-family ipv6 unicast
      maximum-paths 32
      maximum-paths ibgp 32
      label-allocation-mode per-vrf
```

The following show command example displays the type:

```
Switch# show fabric forwarding ip aggregate-subnet-prefix vrf vpn3

HMM aggregate-subnet-prefix IPv4 routing table information for VRF vpn3

Status: *-valid, x-deleted, c-cleaned in 00:03:36

       Host                   Type                 Flags
*   40.1.1.1/8          static | dynamic          0x40204
```

CHAPTER 14

Multi-Mobility Domain Auto-Configuration

Multi-Mobility Domain

Note This chapter is applicable only for multi-tenancy lite version as multi-tenancy full version already supports multi-mobility.

The multi-mobility domain chapter describes the characteristics of DFA multi-mobility domain, mobility domain and detectable range, auto-configuration flow on a switch, per-port VLAN auto-configuration, VPC+ and per-port VLAN translation, and per-port VLAN translation and FEX.

Information About Multi-Mobility Domain

A mobility domain defines a unique Layer-2 name space (for example, the VLAN domain and range), which can be represented by a virtual machine manager or data center definition.

A mobility domain is configured using the **fabric database mobility-domain** *name* command in a leaf switch.

This global mobility domain is used as a key to retrieve the DFA auto-configuration profile for a dot1q based host/tenant from the remote repository (LDAP). The lookup is dot1q VLAN ID and mobility domain. This is a switch global configuration, hence a leaf switch can only belong to a single mobility domain.

A given dot1q VLAN ID can be associated only to a single mobility domain that is the same dot1q VLAN ID cannot be used for different hosts/tenants belonging to different customer on a leaf switch. To overcome this situation, DFA multi-mobility domain feature introduces the flexibility to have the following:

- Reuse of a same dot1q VLAN ID on different hosts (VMs) on a leaf switch, which might belong to different customers while segregating the traffic for these VLAN IDs using the VLAN translation feature.

Note This feature is only applicable to dot1q packet-instantiated auto-configuration, but not applicable to VDP control plane-instantiated auto-configuration.

This feature is supported only on Cisco Nexus 5600 Platform Switches and Cisco Nexus 6000 Series Switches.

Understanding Multi-Mobility Domain

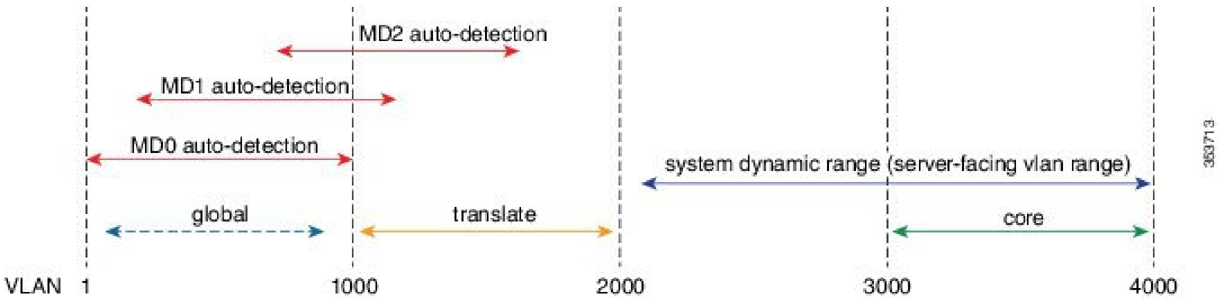
- The multi-mobility domain feature allows configurations of multiple mobility domains on a leaf switch and ports are made members of one of the mobility domain. A port can belong to one mobility domain (MD) at any time.
- A set of dot1q VLAN IDs is also defined in a leaf switch, which can be reused and are subject to the VLAN translation for the mobility domains. This set of VLAN ID is referred to as the translation VLAN range. The other VLAN IDs have global significance (no translation) across the mobility domains.
- The VLAN IDs, which are not part of translation VLAN range, have global significance (no translation) across the mobility domains.
- DFA auto-configuration with VLAN translation is triggered for VLAN ID in the translation range using the following lookup key: incoming dot1q VLAN ID + interface mobility domain. Hence, if required, there has to be a unique DFA auto-configuration remote DB (LDAP) entry for each VLAN ID in the translation range per mobility domain.
- A VLAN translation involves a pair of VLANs: the incoming dot1q tag on the wire, which we will refer to as the from VLAN, and the translated VLAN, which we will refer to as the to VLAN.
- The to VLAN is picked from the DFA Dynamic Server VLAN Pool during the auto-configuration.
- The VLAN translation is unique to a mobility domain and will be applied to all ports belonging to the mobility domain during the auto-configuration. The appropriate switchport VLAN mapping <from> <to> CLIs will be generated on the ports.
- A special mobility domain, called the global mobility domain, has to be provisioned on the switch before other mobility domain can be provisioned and is mandatory in case other mobility domain has to be provisioned. This mobility domain has these characteristics:
    - Equivalent of the global mobility domain from previous release
    - All switch Layer-2 CE access/trunk interfaces belong to this mobility domain by default
    - Does not support any VLAN translations, that is VLAN IDs defined under this range will not be translated
- VLAN IDs, which are not in the translation range, can be auto-configured from any interface/port regardless of the mobility domain the interface/port belongs to. There is a single DFA auto-configuration entry for the global VLAN ID in the remote database (LDAP). This entry is looked up using global VLAN ID + global mobility domain.
- The maximum number of mobility domains that can be supported is 96 per switch.

Mobility Domain Detectable Range

- You must specify the set of detectable VLANs during mobility domain configuration and it must be a subset of the (4K - DFA Dynamic System VLAN range).
    - The global mobility domain does not support VLAN translation, hence its detectable range cannot cover the translate range.
- The figure below shows the relationships of the new configurations and the mobility domain detectable ranges.

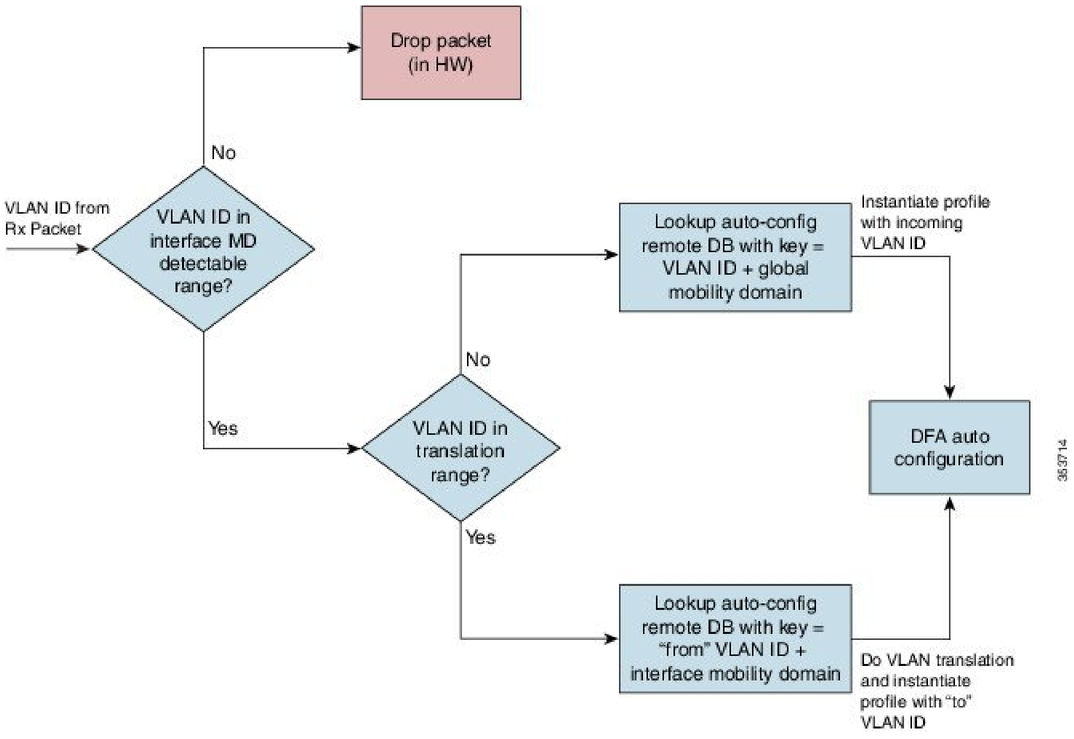
Note VLAN IDs used are only for illustration. MD0 is the global mobility domain.

Figure 26: Mobility Domain Detectable Range



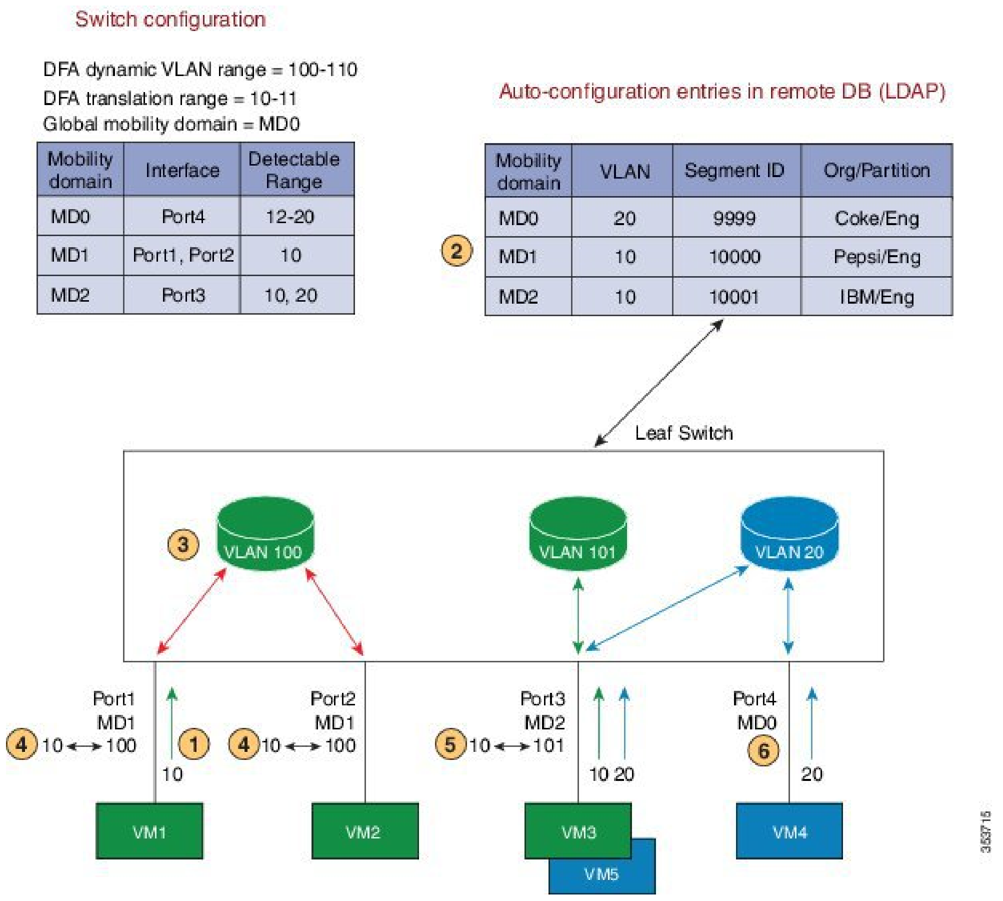
The DFA auto-configuration flow summary on switch is shown below.

Figure 27: Auto-Configuration Flow Summary


Drop packet (in HW)
No
VLAN ID from Rx Packet
VLAN ID in interface MD detectable range?
Yes
Lookup auto-config remote DB with key = VLAN ID + global mobility domain
Instantiate profile with incoming VLAN ID
No
VLAN ID in translation range?
DFA auto configuration
Yes
Lookup auto-config remote DB with key = "from" VLAN ID + interface mobility domain
Do VLAN translation and instantiate profile with "to" VLAN ID
353714

DFA Per-Port VLAN Auto-Configuration

Figure 28: DFA Per-Port VLAN Auto-Configuration



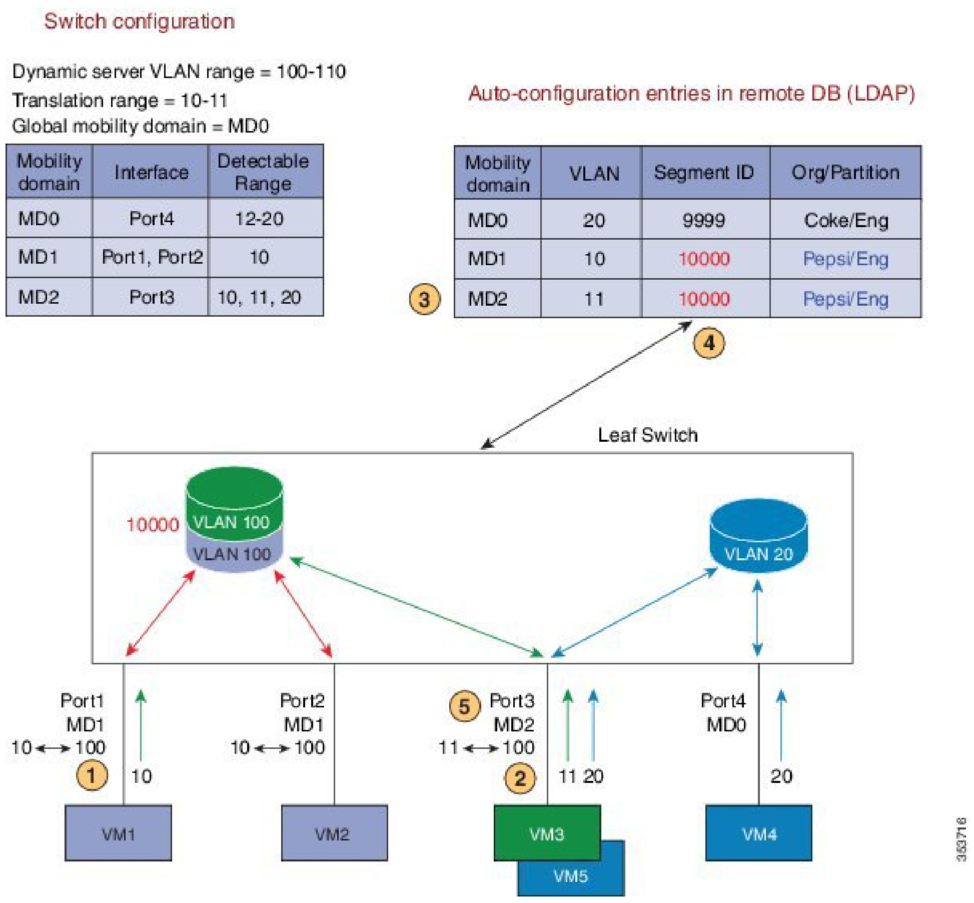
Mobility domain	Interface	Detectable Range
MD0	Port4	12-20
MD1	Port1, Port2	10
MD2	Port3	10, 20

Mobility domain	VLAN	Segment ID	Org/Partition
MD0	20	9999	Coke/Eng
MD1	10	10000	Pepsi/Eng
MD2	10	10001	IBM/Eng

1. Packet arrives on interface.
2. Lookup VLAN 10 + MD1 in remote DB.
3. Determine the "to" VLAN to use for translation. Pick free VLAN from system dynamic VLAN range.
4. Program VLAN translation (switchport VLAN mapping 10-100) on all interfaces of the mobility domain.
5. Similar process as above for VLAN 10 packet arriving on MD2 interface. Note that the "to" VLAN is different, to provide the traffic segregation.
6. Auto-configuration of a global VLAN 20. VLAN is available on all interfaces, except on MD1, where it is not a part of the detectable range.

Stitching of Multi-Mobility Domain (Special Case)

Figure 29: Stitching of Multi-Mobility Domain



Switch configuration

Dynamic server VLAN range = 100-110
Translation range = 10-11
Global mobility domain = MD0

Mobility domain	Interface	Detectable Range
MD0	Port4	12-20
MD1	Port1, Port2	10
MD2	Port3	10, 11, 20

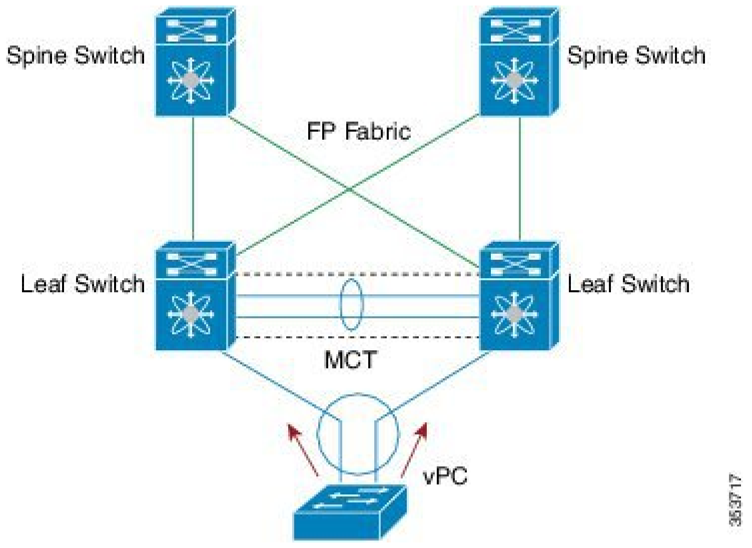
Auto-configuration entries in remote DB (LDAP)

Mobility domain	VLAN	Segment ID	Org/Partition
MD0	20	9999	Coke/Eng
MD1	10	10000	Pepsi/Eng
MD2	11	10000	Pepsi/Eng

1. For VLAN 10 packet arriving on MD1 interface similar process as in previous example and "to" VLAN is 100 for MD1.
2. Packet arrives on interface.
3. Lookup VLAN 11 + MD2 in remote database (DB).
4. The "to" VLAN already exist in the local database (DB) for the retrieved Cisco Virtual Network Identifier (VNI).
5. Program VLAN translation (switchport vlan mapping 11 100) on all interfaces of the mobility domain.

VPC and Per-Port VLAN Translation

Figure 30: VPC+ and Per-Port VLAN Translation



- The mobility domain configuration needs to be consistent on the VPC+ switches to avoid auto-configuration issues.
- New per-interface VPC+ inconsistency rules introduced that suspend the interfaces on an inconsistency.

Per-Port VLAN Translation and FEX

- Fabric Extender (FEX) HW/SW design requires the same VLAN translations for all Host Interfaces (HIF) belonging to the FEX module.
    - The obvious conclusion is that entire FEX module can only belong to one mobility domain.
    - The mobility domain can be configured on HIF (and HIF PO) interfaces only. The last successful HIF mobility domain configuration determines the mobility domain for the entire FEX module.
        - All HIFs that are not configured with the same mobility domain as the FEX module mobility domain are error-disabled with status/reason = **Mobility-DomainMismatch**
        - The HIFs are automatically recovered from the error-disabling when the mobility domain configuration is made consistent with the FEX module
- The VLAN translation CLIs are not generated for the HIF interfaces after the auto-configuration, but instead are programmed for the FEX NIF interfaces/PO.

Configuring Multi-Mobility Domain Auto-Configuration

Configuring Translate Eligible VLANs

Before you begin

- Feature Fabric Forwarding should be enabled.
- System fabric dynamic VLANs should be configured.

Guidelines

- Ensure translate eligible range does not overlap system fabric dynamic VLAN ranges.
- Ensure translate eligible range does not include any VLANs in the switch that have already been created.

SUMMARY STEPS

1. **configure terminal**
2. **[no] system fabric translate-vlans** *vlan-range*
3. **copy running-config startup-config**
4. **show running-config all**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**[no] system fabric translate-vlans** *vlan-range* **Example:** `switch(config)# system fabric translate-vlans 100-110,1001-1499,3501-3502`	Configures system eligible translate VLAN range for a leaf switch.
Step 3	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	(Optional) Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.
Step 4	**show running-config all** **Example:** `switch# show running-config all`	Displays the running configuration for the switch, which also includes translate eligible VLAN range configuration.

Configuring Mobility Domain

Before you begin

- Global mobility domain must be configured before other mobility domains

Guidelines

- All Layer-2 CE access/trunk ports become part of this mobility domain automatically
- Detectable VLAN rules:
  - Must include the native VLANs of the trunks (including VLAN 1) for proper Layer-2 protocol operations
  - Cannot overlap the system dynamic VLAN range
  - For the global mobility domain, it cannot overlap the translate range
- The "default" keyword will set the detectable range as follows:
  - Global mobility domain: 4K – system dynamic VLAN range – translate range
  - Mobility domain: 4K – system dynamic VLAN range

SUMMARY STEPS

1. **configure terminal**
2. **[no] system fabric global-mobility-domain detectable-vlans** {<*vlan-id-or-range*> **| default**}
3. **[no] system fabric mobility-domain** *md-name* **detectable-vlans**{<*vlan-id-or-range*> **| default**}
4. **show global-mobility-domain**
5. **show mobility-domain** <*md-name*>
6. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.

Step 2	**[no] system fabric global-mobility-domain detectable-vlans** {<*vlan-id-or-range*> **| default**}
	Example:
	`switch(config)# system fabric global-mobility-domain detectable-vlans 1, 200-998`
		Configures global mobility domain and detectable VLANs for this global mobility domain.
		Note Interface native VLAN should be included in the detectable range of mobility domain.

Step 3	**[no] system fabric mobility-domain** *md-name* **detectable-vlans**{<*vlan-id-or-range*> **| default**}
	Example:
	`switch(config)# system fabric mobility-domain md2 detectable-vlans 1, 100-110, 500, 1000-1008`
		Configures other mobility domains in the switch.

Step 4	**show global-mobility-domain** **Example:** `switch# show global-mobility-domain`	(Optional) Displays detectable VLANs configured under global mobility domain and all interfaces which are part of global mobility domain.

	Command or Action	Purpose
Step 5	**show mobility-domain** <*md-name*> **Example:** `switch# show mobility-domain md2`	(Optional) Displays detectable VLANs and translate eligible VLANs configured under the input mobility domain and interfaces which are part of this mobility domain.
Step 6	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	(Optional) Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Example

This example shows how to display the global mobility domain information:

```
switch# show global-mobility-domain

Detectable VLANs: 200-998,3503
Translate VLANs:
Interfaces: Eth1/9 Eth1/10 Eth1/11 Eth1/12 Eth1/14 Eth1/15 Eth1/16 Eth1/17 Eth1/18 Eth1/19
 Eth1/20 Eth1/21 Eth1/22 Eth1/23 Eth1/24 Eth1/25
Eth1/26 Eth1/27 Eth1/28 Eth1/29 Eth1/30 Eth1/31 Eth1/32 Eth1/33 Eth1/34 Eth1/35 Eth1/36
Eth1/37 Eth1/38 Eth1/41 Eth1/42 Eth1/43 Eth1/44
Eth1/45 Eth1/46 Eth1/48
```

This example shows how to display the other mobility domain information:

```
switch# show mobility-domain md2

Detectable VLANVLANs: 1,100-110,500,1000-1008
Translate VLANVLANs: 100-110,1001-1008
Interfaces: Po5 Po11 Po12 Po13 Po14 Po15 Po20 Po21 Eth1/1 Eth1/2 Eth1/3 Eth1/4 Eth1/5 Eth1/6
 Eth1/7 Eth1/8 Eth1/13 Eth101/1/2 Eth101/1/10
```

Configuring Per Port Mobility Domain

Before you begin

- Ensure translate VLAN ranges and mobility domains are configured

Guidelines

- CLI configuration is supported only for Layer-2 CE trunk interfaces
- Only Layer-2 CE trunk interfaces can be made members of non-global mobility domain
- Issuing the 'no' CLI moves the interface to the global mobility domain automatically

SUMMARY STEPS

1. **configure terminal**
2. **interface ethernet** *slot/chassis*
3. **switchport mode trunk**

4. **[no] switchport mobility-domain** <*md-name*>
5. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**interface ethernet** *slot/chassis* **Example:** `switch(config)# interface ethernet 1/1`	Enters interface configuration mode.
Step 3	**switchport mode trunk** **Example:** `switch(config-if)# switchport mode trunk`	Configures interface in switchport trunk mode.
Step 4	**[no] switchport mobility-domain** <*md-name*> **Example:** `switch(config)# switchport mobility-domain md2`	Configures mobility domain on an interface.
Step 5	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	(Optional) Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Verifying Per Port VLAN Mapping

SUMMARY STEPS

1. **show running-config interface ethernet** *slot/chassis*.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show running-config interface ethernet** *slot/chassis*. **Example:** `switch# show running-config interface ethernet 1/1`	Displays the running configuration of an input interface.

Show Commands - vpc + related

This example shows the port-channel configuration:

```
switch# show running-config interface po5

!Command: show running-config interface port-channel5
```

```
!Time: Mon Dec  8 02:34:01 2014

version 7.1(0)N1(1)

interface port-channel5
  switchport mode trunk
  switchport vlan mapping 101 3000
  switchport mobility-domain md1
  vpc 5
```

This example shows how to check the interface MD and its corresponding detectable range:

```
switch# show vpc consistency-parameters int port-channel 5

Legend:
        Type 1 : vPC will be suspended in case of mismatch
Name                        Type  Local Value             Peer Value
-------------               ----  ----------------------  -----------------------
Shut Lan                    1     No                      No
STP Port Type               1     Default                 Default
STP Port Guard              1     None                    None
STP MST Simulate PVST       1     Default                 Default
mode                        1     on                      on
Speed                       1     1000 Mb/s               1000 Mb/s
Duplex                      1     full                    full
Port Mode                   1     trunk                   trunk
Native VLAN                 1     1                       1
MTU                         1     1500                    1500
Admin port mode             1     trunk                   trunk
Detectable VLANs            1     20-100                  20-100
Mobility Domain             1     MD1                     MD1
vPC+ Switch-id              1     50                      50
vPC card type               1     Empty                   Empty
Allowed VLANs               -     1                       1
Local suspended VLANs       -     -                       -
```

This example shows how to check the interface mobility domain configuration related consistency for VPC interface:

```
switch# show vpc brief

[snip]
Per-VLAN consistency status    : failed
Type-2 consistency status      : success
vPC role                       : secondary
Number of vPCs configured      : 4
[snip]
vPC Peer-link status
---------------------------------------------------------------------
id   Port   Status Active VLANs
--   ----   ------ --------------------------------------------------
1    Po24   up     1,2130
vPC status
---------------------------------------------------------------------------
id     Port        Status Consistency Reason       Active VLANs vPC+ Attrib
--     ----------  ------ ----------- ------       ------------ -----------
[snip]
5    Po5           down*  failed      Mobility     -            DF: No, FP
                                      domain                    MAC: 50.0.0
                                      related
                                      inconsistency
```

CHAPTER 15

Dynamic Virtual Port

Dynamic Virtual Port

Note This chapter is applicable only for multi-tenancy lite version as multi-tenancy full version already supports this as the default configuration.

The Dynamic Virtual Port (DVP) feature describes the following:

- Virtual Port
- Problems with Virtual Ports
- Overview of Dynamic Virtual Ports

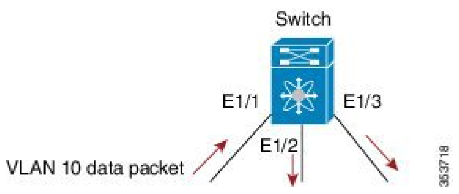
Virtual Port

- A VLAN is activated on all Layer-2 CE trunk interfaces when it is created on the switch
- A (port, VLAN) instance is referred as a 'Virtual Port (VP)' or 'Logical Interface (LI)'

Limitations of Virtual Ports

- Each VP is associated with system resources including memory, control protocol instance, CPU processing time to bring the VP up and so on. This places a limit on the number of instances (port, VLAN) that can be safely supported on the switch—a scaling limit.
- Also, a VP causes traffic to be flooded on interfaces that might not need them but just because the VLAN is activated on that interface.

Figure 31: VP Limitation



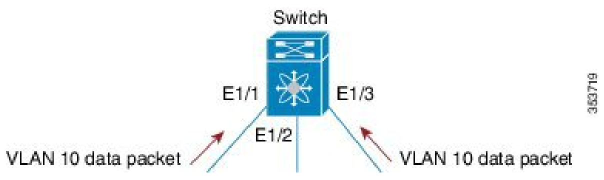
In the figure shown above:

- E1/1-3 are CE trunks
- When VLAN 10 is created on switch:
  - E1/1-3 will be members of VLAN 10
  - STP will run on E1/1-3
  - Traffic ingressing E1/1 will be flooded out on E1/2-3

Overview of Dynamic Virtual Port

- The DVP feature is designed to address the (port, VLAN) scalability on the switch
- When a VLAN is created on the switch: it is not brought up on the CE trunk interfaces by default
- Dynamically add VLAN to interfaces based on the requirement thus saving crucial resources
- Dynamically remove the VP when not required
- Works in conjunction with workload auto-configuration and the host mobility manager (HMM)
- Per Interfaces control available to follow traditional VP creation mechanism, that is CLI is also available to enable/disable the dynamic virtual port on an given interface

Figure 32: VP Scalability



When a host connected to E1/1 and sends data packet on VLAN 10:

- Workload auto-configuration feature will detect host for VLAN 10 on E1/1
- VLAN 10 is created on switch
- Create (E1/1, 10) instance only (not on E1/2 & E1/3)
- STP for VLAN 10 runs on E1/1 only

Another host connected to E1/3 and sends data packet on VLAN 10:

- Workload auto-configuration feature will detect host for VLAN 10 on E1/3

- VLAN 10 creation is skipped since it already exists on switch
- Create (E1/3, 10) instance
- STP for VLAN 10 runs on E1/1 & E1/3 only

Guidelines and Limitations

- This feature cannot coexist with FCOE VLANs so the feature must be disabled for an interface, which is bonded with FCOE VLAN
- If the allowed VLAN list is configured manually on an interface (using the **switchport trunk allowed vlan***vlan-id* command), the DVP must be disabled locally for that interface using the **no switchport trunk allocate vlan dynamic** command.
- You must ensure the consistent configuration of DVP in case of VPC peers

Configuring Dynamic Virtual Port

Configuring Dynamic Virtual Port on a Switch

Before you begin

- Feature Fabric Forwarding should be enabled
- System fabric dynamic VLANs should be configured

Guidelines

- Copy running to startup and reboot is required for this feature to become operational on the switch
- This feature works only with auto-configuration (feature fabric forwarding)
- No additional license needed (requirements inherited from fabric forwarding)
- Feature enabled in default POAP Leaf config template

SUMMARY STEPS

1. **configure terminal**
2. **[no] system default trunk allocate VLAN dynamic**
3. **show system trunk dynamic status**
4. **copy running-config startup-config**
5. **reboot**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal`	
Step 2	**[no] system default trunk allocate VLAN dynamic** **Example:** `switch(config)# system default trunk allocate vlan dynamic`	Configures Dynamic Virtual Port.
Step 3	**show system trunk dynamic status** **Example:** `switch# show system trunk dynamic status`	Displays Dynamic Virtual Port configuration and operational status.
Step 4	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.
Step 5	**reboot** **Example:** `switch(config)# reboot`	Reboot is required for feature to become operational on the switch.

Example

Note The DVP feature will be operational by default on all CE trunks after switch reload. This requires copy running-config to startup-config before switch reload.

```
switch# show system trunk dynamic status

Global Status
-------------
Configured  : Enabled (Will take effect on reboot)
Operational : Disabled
Number of operationally enabled dynamic logical interfaces   : 0
```

This example shows the system trunk dynamic status after copy run start and reboot:

```
switch# show system trunk dynamic status

Global Status
-------------
Configured  : Enabled
Operational : Enabled
Number of operationally enabled dynamic logical interfaces   : 0
```

Disabling auto-configuration feature will require a reboot with saved running config:

```
switch# show system trunk dynamic status

Global Status
-------------
Configured  : Enabled
Operational : Enabled
```

```
Number of operationally enabled dynamic logical interfaces   : 10
```

After you perform the **no feature fabric forwarding** command, the system default trunk allocate VLAN dynamic feature is disabled but still operational on all CE trunks, it will be non-operational after switch reload. This requires copy running-config to startup-config before switch reload:

```
switch# show system trunk dynamic status

Global Status
-------------
Configured  : Disabled  (Will take effect on reboot)
Operational : Enabled
Number of operationally enabled dynamic logical interfaces   : 10
```

Configuring Dynamic Virtual Port on an Interface

Guidelines

- Enables/Disables the feature for an interface
- This CLI is default on all Layer-2 trunk ports
- CLI is operational only on CE trunk interfaces
- CLI is disabled in default POAP Leaf configuration template
- CLI is operational when DVP feature is operationally enabled

SUMMARY STEPS

1. **configure terminal**
2. **[no] switchport trunk allocate vlan dynamic**
3. **show system trunk dynamic status** [**enabled-interfaces** | **interfaces** *int1*]
4. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**[no] switchport trunk allocate vlan dynamic** **Example:** `switch(config)# switchport trunk allocate vlan dynamic`	Configures Dynamic Virtual Port Feature on an input interface.
Step 3	**show system trunk dynamic status** [**enabled-interfaces** | **interfaces** *int1*] **Example:**	Displays list of DVP enabled interfaces or enable status on given interface.

	Command or Action	Purpose
	`switch# show system trunk dynamic status enabled-interfaces`	
Step 4	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	(Optional) Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Example

This example shows how to configure the dVP feature on an interface:

```
switch(config-if)# switchport trunk allocate VLAN dynamic
switch(config-if)# show system trunk dynamic status enabled-interfaces

Interface   Status
---------   ------
Eth1/2      Enabled

switch# show running-config interface ethernet 1/2

interface Ethernet1/2
  switchport mode trunk

switch(config-if)# no switchport trunk allocate VLAN dynamic
switch(config-if)# show running-config interface ethernet 1/2

!Command: show running-config interface Ethernet1/2
interface Ethernet1/2
  switchport mode trunk
  no switchport trunk allocate VLAN dynamic

switch(config-if)# show system trunk dynamic status enabled-interfaces

Interface   Status
---------   ------
Eth1/2      Disabled
```

This example shows the sample output of show system trunk dynamic status:

```
switch# show system trunk dynamic status interface ethernet 1/3-5

Interface   Status
---------   ------
Eth1/3      Enabled
Eth1/4      Disabled
Eth1/5      Not-Applicable
```

This example shows the instances on the CE trunk interfaces:

```
switch# show interface trunk

--------------------------------------------------------------------------------
 Port          Native  Status        Port
               VLAN                  Channel
 --------------------------------------------------------------------------------
 Eth1/1        1       trunking      --
 Eth1/2        1       trunking      --
```

```
Eth1/3        1       trunking      --

--------------------------------------------------------------------------------
Port          VLANs Allowed on Trunk
--------------------------------------------------------------------------------
Eth1/1        1-4094
Eth1/2        1-4094
Eth1/3        1-4094
[snip]

--------------------------------------------------------------------------------
Port          STP Forwarding
--------------------------------------------------------------------------------
Eth1/1        none
Eth1/2        10
Eth1/3        none
```

CHAPTER 16

Appendix

Appendix

The configuration of roles are given below.

Leaf Node Configuration

This is a configuration example for leaf brought up by POAP using FUJI2 leaf template. This example shows two route reflectors and LDAP configuration for auto-configuration.

Note This is not a complete configuration, but only an illustration of configuration for unicast forwarding shown in Fabric Control Segment and BGP Control Plane.

The following is an example for BGP and FabricPath IS-IS:

```
install feature-set fabricpath
install feature-set fabric
feature-set fabricpath
feature-set fabric

feature fabric forwarding
feature bgp

ip access-list HOSTS
  10 permit ip any any
ipv6 access-list V6HOSTS
  10 permit ipv6 any any

ip prefix-list control-subnet seq 100 permit 44.2.0.0/22
route-map ALL-PATHS permit 10
  set path-selection all advertise
route-map FABRIC-RMAP-REDIST-HOST deny 10
  match interface Vlan1
route-map FABRIC-RMAP-REDIST-HOST permit 20
  match ip address HOSTS
route-map FABRIC-RMAP-REDIST-SUBNET permit 10
  match tag 12345
route-map FABRIC-RMAP-REDIST-V6HOST deny 10
  match interface Vlan1
```

```
route-map FABRIC-RMAP-REDIST-V6HOST permit 20
  match ip address V6HOSTS
route-map bgpMap permit 10
  match route-type internal
route-map bgp_next_hop_filter deny 100
  match ip address prefix-list control-subnet
route-map bgp_next_hop_filter permit 200
  match ip address HOSTS


!control SVI
vlan 2
  mode fabricpath
interface Vlan1
  no shutdown
  mtu 9192
  ip address 44.2.3.33/22
  ipv6 forward
  fabric forwarding control-segment

router bgp 100
  !Generally the SVI address of the local control subnet
  router-id 44.2.3.33
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map bgp_next_hop_filter
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family vpnv4 unicast
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
  address-family vpnv6 unicast
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
  address-family ipv4 mvpn
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 mvpn
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
 !First RR
  neighbor 44.2.0.101 remote-as 100
    address-family ipv4 unicast
      send-community both
    address-family ipv6 unicast
      send-community both
    address-family vpnv4 unicast
      send-community both
    address-family vpnv6 unicast
      send-community both
    address-family ipv4 mvpn
      send-community both
    address-family ipv6 mvpn
      send-community both
```

```
!Second RR
  neighbor 44.2.0.144 remote-as 100
    address-family ipv4 unicast
      send-community both
    address-family ipv6 unicast
      send-community both
    address-family vpnv4 unicast
      send-community both
    address-family vpnv6 unicast
      send-community both
    address-family ipv4 mvpn
      send-community both
    address-family ipv6 mvpn
      send-community both
```

LDAP configuration with a backup LDAP. It is not mandatory to have second LDAP but we recommend.

```
fabric database type network
  server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
     db-table ou=networks,dc=cisco,dc=com key-type 1
     db-security user admin password cisco123

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
     db-table ou=networks,dc=cisco,dc=com key-type 1
     db-security user admin password cisco123

fabric database type profile
  server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
     db-table ou=profiles,dc=cisco,dc=com
     db-security user admin password cisco123

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
     db-table ou=profiles,dc=cisco,dc=com
     db-security user admin password cisco123

fabric database type partition
  server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
     db-table ou=partitions,dc=cisco,dc=com
     db-security user admin password cisco123

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
     db-table ou=partitions,dc=cisco,dc=com
     db-security user admin password cisco123
```

Route Reflector Configuration

The following is an example for BGP:

```
router bgp 100
!Generally the SVI address of the local control subnet
  router-id 44.2.0.101
  address-family ipv4 unicast
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 unicast
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths selection route-map ALL-PATHS
  address-family vpnv4 unicast
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family vpnv6 unicast
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv4 mvpn
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 mvpn
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths send
    additional-paths receive
    additional-paths selection route-map ALL-PATHS

  neighbor 44.2.0.0/22 remote-as 100
    address-family ipv4 unicast
      send-community both
      route-reflector-client
    address-family ipv6 unicast
      send-community both
      route-reflector-client
    address-family vpnv4 unicast
      send-community both
      route-reflector-client
    address-family vpnv6 unicast
      send-community both
      route-reflector-client
    address-family ipv4 mvpn
      send-community both
      route-reflector-client
    address-family ipv6 mvpn
      send-community both
      route-reflector-client
```

Border Leaf Node Configuration

The example shows the border leaf configuration. The border leaf specific configuration is in bold and the rest is common to interior leaf. A tenant configuration and its neighbor ship to DCI Edge router is also shown in this example.

The following is an example for BGP:

```
!One tenant which is extended to DC Edge box
vrf context CiscoLive:Part4
  vni 65004
  rd auto
  address-family ipv4 unicast
    route-target both auto

!Sub interface towards DC Edge box
interface port-channel400.1004
  encapsulation dot1q 1004
  vrf member CiscoLive:Part4
  ip address 4.1.1.2/24

!!Border leaf specific policy:
!Deny the default route received from within fabric by other border leaf(s) Permit any other
 route
ip prefix-list default-route seq 5 permit 0.0.0.0/0 le 1
route-map deny-default-route deny 100
```

```
  match ip address prefix-list default-route
route-map deny-default-route permit 200
  match ip address HOSTS

router bgp 100

 !Generally the SVI address of the local control subnet
  router-id 44.2.3.63
  fabric-soo 100:1
  address-family ipv4 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    nexthop route-map bgp_next_hop_filter
    default-information originate
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 unicast
    redistribute hmm route-map FABRIC-RMAP-REDIST-V6HOST
    maximum-paths ibgp 2
    nexthop trigger-delay critical 250 non-critical 10000
    default-information originate
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family vpnv4 unicast
    nexthop trigger-delay critical 250 non-critical 10000
!optional configuration: use when all tenants are on all Border Leafs
    default-information originate always rd 44.2.3.63:100 route-target 100:9999
    additional-paths receive
  address-family vpnv6 unicast
    nexthop trigger-delay critical 250 non-critical 10000
     !optional configuration: use when all tenants are on all Border Leafs
    default-information originate always rd 44.2.3.63:100 route-target 100:9999
    additional-paths receive
  address-family ipv4 mvpn
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
  address-family ipv6 mvpn
    nexthop trigger-delay critical 250 non-critical 10000
    additional-paths receive
    additional-paths selection route-map ALL-PATHS
!First RR
neighbor 44.2.0.101 remote-as 100
    address-family ipv4 unicast
      send-community both
      route-map deny-default-route in
      next-hop-self
    address-family ipv6 unicast
      send-community both
    address-family vpnv4 unicast
      send-community both
      route-map deny-default-route in
    address-family vpnv6 unicast
      send-community both
    address-family ipv4 mvpn
      send-community both
    address-family ipv6 mvpn
      send-community both
!Second RR
  neighbor 44.2.0.144 remote-as 100
    address-family ipv4 unicast
      send-community both
      route-map deny-default-route in
      next-hop-self
    address-family ipv6 unicast
      send-community both
    address-family vpnv4 unicast
      send-community both
      route-map deny-default-route in
    address-family vpnv6 unicast
      send-community both
    address-family ipv4 mvpn
      send-community both
    address-family ipv6 mvpn
      send-community both
!BGP session for tenant towards Border Leaf
vrf CiscoLive:Part4
    address-family ipv4 unicast
      maximum-paths 2
      maximum-paths ibgp 2
      additional-paths send
      additional-paths receive
      additional-paths selection route-map ALL-PATHS

!For Border Leaf to DC Edge tenant extension auto-configuration
fabric database override-vrf-profile vrf-common-universal-bl
server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
     db-table ou=profiles,dc=cisco,dc=com
     db-security user admin password cisco123

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
     db-table ou=profiles,dc=cisco,dc=com
     db-security user admin password cisco123

fabric database type partition
  server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
     db-table ou=partitions,dc=cisco,dc=com
     db-security user admin password cisco123

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
     db-table ou=partitions,dc=cisco,dc=com
     db-security user admin password cisco123

!For Border Leaf to DC Edge tenant extension auto-configuration feature
fabric database type bl-dci
    server protocol ldap host ldap-server1.cisco.com vrf management enable-ssl
    db-table ou=bl-dcis,dc=cisco,dc=com
    db-security user admin password cisco123

  server protocol ldap host ldap-server2.cisco.com vrf management enable-ssl
    db-table ou=bl-dcis,dc=cisco,dc=com
    db-security user admin password cisco123
```

DC Edge Router

Example of tenant session towards border leaf for Cisco Nexus 7000 Series Switches running on 6.2 image.

```
!Sub interface towards border leaf
interface port-channel400.1004
  encapsulation dot1q 1004
  vrf member CiscoLive:Part4
  ip address 4.1.1.1/24
  no shutdown

!External RD and RT
vrf context CiscoLive:Part4
  rd 65500:1604
  address-family ipv4 unicast
    route-target import 65500:1604
    route-target export 65500:1604
router bgp 400
  !BGP session towards border leaf
  vrf CiscoLive:Part4
    neighbor 4.1.1.2 remote-as 100
      address-family ipv4 unicast
        send-community both
        default-originate
```