Cisco IOS XRv 9000 Router Installation and Configuration Guide

First Published: 2015-08-26

Last Modified: 2026-06-09

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

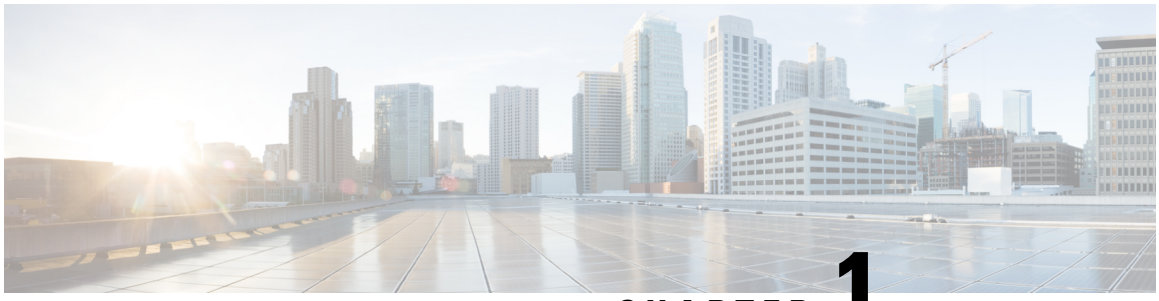
Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2017–2026 Cisco Systems, Inc. All rights reserved.



CHAPTER 1

Preparing for Installation

This chapter provides information about the prerequisites for installing Cisco IOS XRv 9000 Router.

- Introduction, on page 1
- Software Configuration and Management, on page 2
- Provisioning the VM, on page 3
- Router Interfaces, on page 3
- Mapping the Router Network Interfaces to VM Network Interface Cards, on page 4
- Cisco IOS XRv 9000 Installation Files, on page 6
- Installing the Cisco IOS XRv 9000 Router, on page 7

Introduction

The Cisco IOS XRv 9000 Router is a cloud-based router that is deployed on a virtual machine (VM) instance on x86 server hardware running 64-bit IOS XR software. Cisco IOS XRv 9000 Router provides traditional Provider Edge services in a virtualized form factor, as well as virtual Route Reflector capabilities. Cisco IOS XRv 9000 Router is based on Cisco IOS XR software, so it inherits and shares the wide breadth of routing functionality available on other IOS XR platforms.

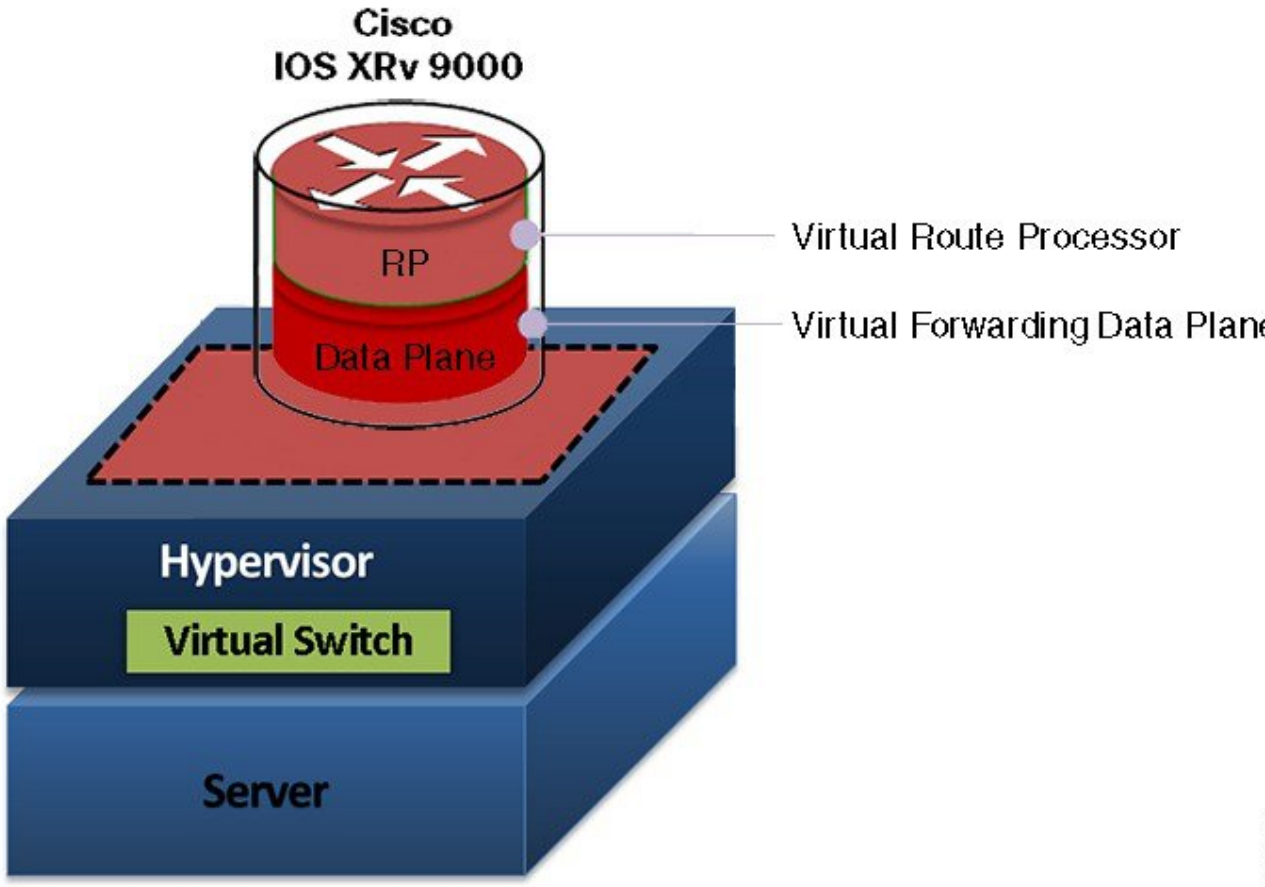
Note **RHEL KVM, ESXi, and AWS VMs are not supported for the Releases 24.4.1 and 24.4.2.**

The XRV-M7-APLN-25G and XRV-M7-APLN-100G M7 appliances are supported starting from Release 24.4.2.

For information on IOS XR features available on Cisco IOS XRv 9000 Router, refer section *Supported Cisco IOS XR Technologies* in the latest *Release Notes for Cisco IOS XRv 9000 Router* .

When the Cisco IOS XRv 9000 Router virtual IOS XR software is deployed on a VM, the Cisco IOS XR software functions just as if it were deployed on a traditional Cisco IOS XR hardware platform. The Cisco IOS XRv 9000 Router combines Route Processor, Line Card, and virtualized forwarding capabilities into a single, centralized forwarding instance. The Cisco IOS XRv 9000 Router has a fully featured, high speed virtual x86 data plane.

Figure 1: Cisco IOS XRv 9000 Router Virtual Form Factor



The Cisco IOS XRv 9000 router can be deployed as virtual form factor or in appliance mode.

This figure shows the basic virtual form factor for the Cisco IOS XRv 9000 Router. The Cisco IOS XRv 9000 Router is deployed as a VM on a hypervisor. Cisco IOS XRv 9000 Router offers various connectivity models, including virtualized interfaces connected to a virtual switch (vSwitch), or physical pass-through of 10G interfaces directly into the VM for maximum performance.

For information on VM requirement, and limitation of the Cisco IOS XRv 9000 Router, see section Virtual Machine Requirements, on page 163.

Cisco IOS XRv 9000 Router is deployed on VMware, KVM hypervisors, or on AWS servers. For details on supported hypervisors, see section Hypervisor Support, on page 164.

Software Configuration and Management

You can perform software configuration and manage the Cisco IOS XRv 9000 Router using VM or in appliance mode.

Using VM

- Provision a serial port in the VM and connect to the serial port to access Cisco IOS XR CLI commands.

- Use the VM console or the console on the virtual serial port to access Cisco IOS XR CLI commands.

Note A serial port can be used to manage a Cisco IOS XRv 9000 Router VM only if the underlying hypervisor supports the associating of a serial port with a VM. See your hypervisor documentation for details.

- Use remote SSH/Telnet to access Cisco IOS XR CLI commands.

  For more information on how to access the router console, see the Access the Cisco IOS XRv 9000 Router Through the Virtual Serial Port , on page 46

Using Appliance

- Connect to the XR console (or a controller) through the serial port which connects directly to CIMC.
- Login to the XR console using admin password.
- Configure the router manually using CLIs.

Provisioning the VM

Cisco hardware routers are normally shipped with the Cisco IOS XR software pre-installed. Because the Cisco IOS XRv 9000 Router is not hardware-based, you must download the Cisco IOS XR software from Cisco.com and install it directly onto the virtual machine (VM). However, as part of the initial installation process, you must first provision the attributes (memory, hard disk, etc) of the VM so that the Cisco IOS XRv 9000 Router image can install and boot.

Also, Cisco IOS XRv 9000 Appliance is the pre-installed Cisco IOS XRv 9000 Router software that is sent from the factory on a bare metal UCS server hardware. Cisco IOS XRv 9000 Appliance is inclusive of all applicable licenses. The Appliance package enables you to virtualize your network routing function without having operational concerns about ownership of hardware and software.

Router Interfaces

Cisco IOS XRv 9000 Router interfaces perform the same function as those on hardware-based Cisco routers. Cisco IOS XRv 9000 interface naming convention is:

- Interfaces are logically named as GigabitEthernet interfaces. These interfaces can either be virtualized interfaces such as VMXNET3 or E1000, or physical 1 Gigabit interfaces which are passed into the VM through PCI passthrough.
- Interfaces are logically named based on their speed and are passed into the VM via PCI passthrough:
  - HundredGigE for 100G interface
  - FortyGigE for 40G interface
  - TenGigE for 10G interface
  - GigabitEthernet for 1G interface

- The interface numbering starts at 0 and increases monotonically; for example:

```
interface GigabitEthernet 0/0/0/0
interface GigabitEthernet 0/0/0/1
interface GigabitEthernet 0/0/0/2
```

or in the case of Ten Gigabit interfaces:

```
interface TenGigE 0/0/0/0
interface TenGigabitE 0/0/0/1
```

or in the case of forty Gigabit interfaces:

```
interface FortyGigE 0/0/0/0
interface FortyGigE 0/0/0/1
```

or in the case of hundred Gigabit interfaces:

```
interface HundredGigE 0/0/0/0
interface HundredGigE 0/0/0/1
```

Mapping the Router Network Interfaces to VM Network Interface Cards

Cisco IOS XRv 9000 Router maps the logical virtual network interface cards (vNICs) or physical NICs assigned to the VM to management Ethernet interface, GigabitEthernet, TenGigE, TwentyFiveGigE, or HundredGigE interface.

Each time Cisco IOS XRv 9000 Router is booted, the first NIC is used as the management Ethernet interface of the virtual router, the second and third NICs are used by the virtual router internally. These three NICS can be E100 or any other. The rest of the NICs are mapped to the data plane as line interfaces.

The naming convention of line interfaces is <GigabitEthernet | TenGigE | TwentyFiveGigE | HundredGigE | FortyGigE> 0/0/0/<port number>.

The port numbers are allocated across the NIC types. Here are the rules for allocating port number :

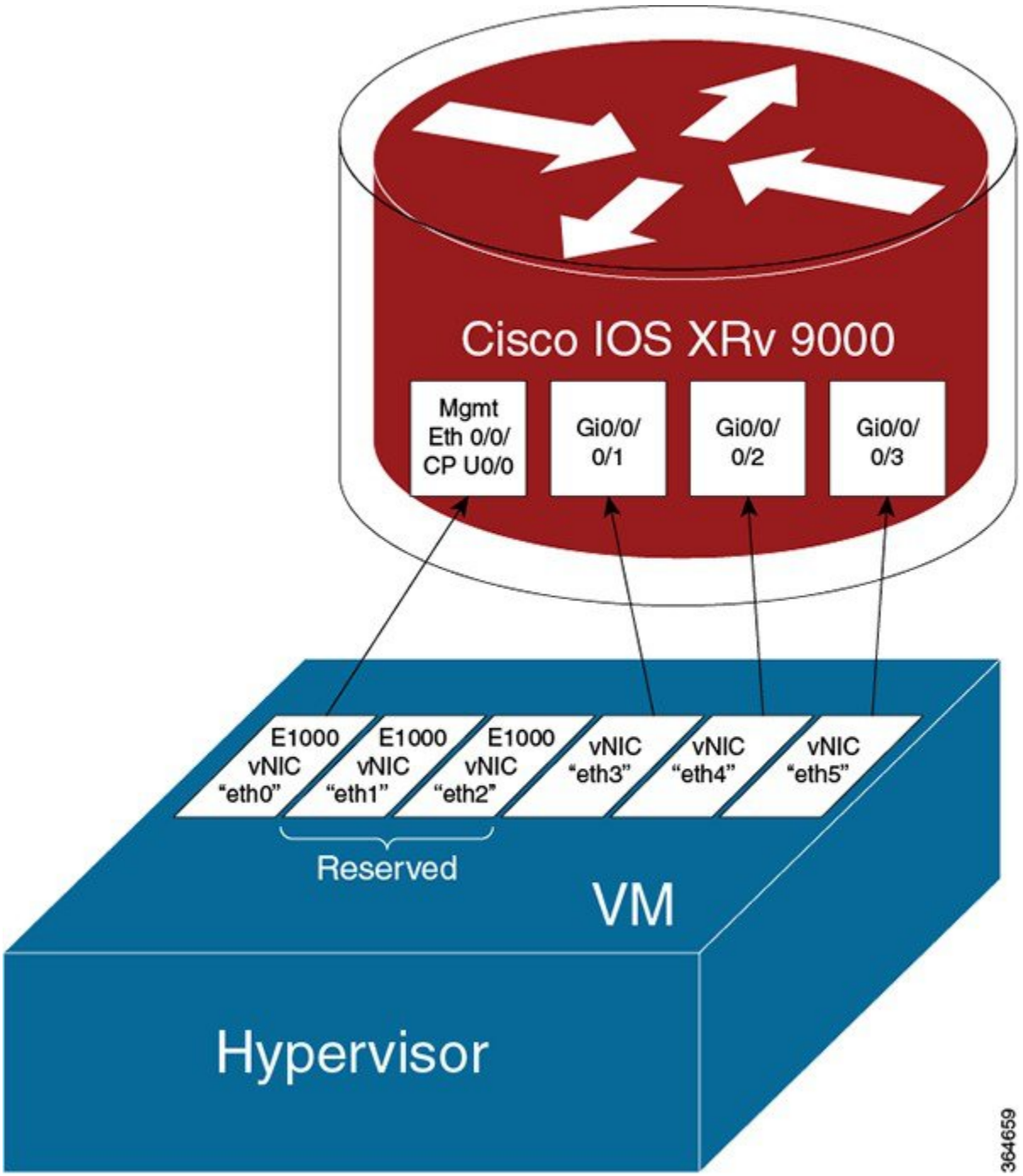
- Allocate faster interfaces before slower interfaces. For example, 10GE interface comes before 1GE then,
- within the same speed allocate lower PCI addresses before higher addresses. For example, 04:00.0 comes before 04.00.1.

Assume that there is one TegGigE NIC, one physical GigabitEthernet NIC and one virtual NIC mapped to the data plane. in such a case then naming of line interface is:

- TenGigE NIC is names as TegGigE 0/0/0/0.
- Physical GigabitEthernet NIC is name as GigabitEthernet 0/0/0/1.
- The virtual NIC is named as GigabitEthernet 0/0/0/2.

This figure shows an example of 6 NICs (virtual and physical) mapped to Cisco IOS XRv 9000 Router. The figure is applicable for VMware and KVM hypervisors:

Figure 2: Mapping NICs with Cisco IOS XRv 9000 Router



For interface mapping on XRv 9000 Appliance, see Appliance Physical Connections Overview, on page 119

Cisco IOS XRv 9000 Installation Files

The following file types are included in the Cisco IOS XRv 9000 Router's software image package and are used to install the Cisco IOS XRv 9000 Router on the supported hypervisors.

- .iso—Used for installing the software image on the VM. This can be used to create a VM in any supported hypervisor environment.
- .ova—Used for deploying the Open Virtualization Appliance (OVA) template on the VM (in TAR format). The OVA image is recommended for deploying Cisco IOS XRv 9000 Router on the VMware ESXi hypervisor. The OVA file contains a Virtual machine disk image (VMDK) with the Cisco IOS XRv 9000 Router software installed.

Note The VMDK disk inside the OVA file is in a stream optimized format and cannot be booted directly. In order to use the the stream optimized VMDK disk it must be converted to a standard Read/Write disk format. During OVA deployment, ESXi will transparently convert the disk to a standard Read/Write VMDK format. A stream optimized VMDK can also be converted to a standard Read/Write VMDK or QCOW2 disk using a standard disk tool like **qemu-img**.

- .qcow2 —Used for booting the software image in KVM OpenStack environments. The qcow2 disk image contains the Cisco IOS XRv 9000 Router software installed.
- virsh.xml—Sample XML that can be used for launching the Cisco IOS XRv 9000 Router in KVM environments using Virsh.
- CML2—You can create a lab with CML2.1(Cisco Modelling Lab) and start XRv 9000 Router on CML.

Installation Image Variants

The above image types come in a default or vrr (virtual Route Reflector) variant. Any image without vrr in the filename is a default variant.

- default—tuned for working as a virtual Provider Edge (vPE) router. vPE has more CPU for data plane versus vRR for control plane. VPE is the default profile for other hypervisors like ESXI. Use VPE mode for high speed L2 functions at PE
- vrr—tuned for working as a virtual Route Reflector (vRR). vRR is the default profile for baremetal appliances.

These are the default type installation images:

- xrv9k-fullk9-x.iso
- xrv9k-fullk9-x.ova
- xrv9k-fullk9-x.qcow2
- xrv9k-fullk9-x.virsh.xml

These are the vrr type installation images:

- xrv9k-fullk9-x.vrr.iso
- xrv9k-fullk9-x.vrr.ova
- xrv9k-fullk9-x.vrr.qcow2
- xrv9k-fullk9-x.vrr.virsh.xml

xrv9k-li-x.pkg—Optional package for Lawful Intercept.

From Cisco IOS XR Software Release 7.2.1, xrv9k-full(k9).iso build has been discontinued. From Cisco IOS XR Software Release 7.2.1, we are building GISO images as a part of post build and these images are renamed to xrv9k-fullk9-iso images as well. With Cisco IOS XR Software Release 7.8.1 XRv 9000 platform has moved to LJAM build and with LJAM migration, automatic GISO post build scripts have been discontinued. So, now we must build the GISO with GISO tool with xrv9k-mini.iso and the required RPMs. You shouldn't use the xrv9k-full and xrv9k-fullk9.iso from Cisco.com.

Build the GISO image with the GISO tool with xrv9k-mini.iso and the required RPMs and try the install upgrade/downgrade.

GISO Tool Link: Download `gisobuild.py` from GitHub

See limitations Golden ISO Workflow, on page 30.

Console Image Variants

If you don't want to login through serial port, then you can go for VGA image which provides VGA console. The installation images with vga in the filename are console image variants. If VGA image is used for installation of the router then the XR console is mapped to the VGA console, else the XR console will be mapped to the first serial port. The installation procedure is same with or without VGA images.

These are the console image variants:

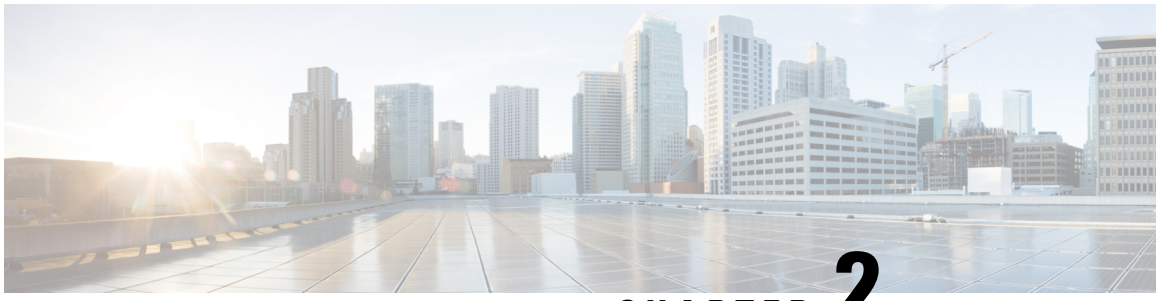
- xrv9k-fullk9_vga-x.iso
- xrv9k-fullk9_vga-x.ova
- xrv9k-fullk9_vga-x.qcow2
- xrv9k-fullk9_vga-x.vrr.iso
- xrv9k-fullk9_vga-x.vrr.ova
- xrv9k-fullk9_vga-x.vrr.qcow2
- xrv9k-fullk9-x_vga.virsh.xml
- xrv9k-fullk9-x.vrr_vga.virish.xml

Installing the Cisco IOS XRv 9000 Router

The Cisco IOS XRv 9000 Router is installed either on VMware or KVM hypervisors. See these sections for detailed information on installing the Cisco IOS XRv 9000 Router:

- *Installing the Cisco IOS XRv 9000 Router in VMware ESXi Environments*
- *Installing the Cisco IOS XRv 9000 Router in KVM Environments*

For XRv 9000 Appliance software installation, see Configuring the Appliance, on page 122



CHAPTER 2

Installing the Cisco IOS XRv 9000 Router in VMware ESXi Environments

These file types are needed to install Cisco IOS XRv 9000 Router on the VMware ESXi hypervisor:

- .iso—Used for installing the image on the VM. This can also be used to create a VM in any supported hypervisor environment.
- .ova—Used for deploying the OVA template on the VM (in TAR format). The OVA image is recommended for deploying Cisco IOS XRv 9000 Router on the VMware ESXi hypervisor.

- Installation Requirements for VMware ESXi, on page 9
- Installing the Cisco IOS XRv 9000 Router to the VM, on page 10
- Installing the Cisco IOS XRv 9000 Router to the VM using OVA Template , on page 10
- Installing the Cisco IOS XRv 9000 Router to the VM using ISO Template , on page 12
- Increasing Performance on VMware ESXi Configurations, on page 15

Installation Requirements for VMware ESXi

Before installing Cisco IOS XRv 9000 Router, you must first set up your VMware environment, including the necessary host and client software. For example, if you are installing Cisco IOS XRv 9000 Router in a VMware ESXi environment, you must first install the vSphere Client.

For information on the installation requirements for VMware ESXi, refer to the latest *Release Notes for Cisco IOS XRv 9000 Router*.

Note When deploying Cisco IOS XRv 9000 Router with OVA file, six vNICs are automatically created. You can manually add these vNICs to the VM after Cisco IOS XRv 9000 Router has booted.

See these sections for information on VMware ESXi support, and supported VMware features and operations:

- *VMware ESXi Support Information*
- *Supported VMware Features and Operations*

Installing the Cisco IOS XRv 9000 Router to the VM

VMWare ESXi supports installing Cisco IOS XRv 9000 Router to the VM using the OVA and ISO file.

Note The KVM and Microsoft Hyper-V implementations do not support deploying the VM using the .ova file. You must install the VM using the .iso file.

Installation using OVA file

The OVA file package includes an OVF file that contains a default VM configuration based on Cisco IOS XR release. Use the default.ova package to deploy Cisco IOS XRv 9000 Router as virtual Provider Edge, a high speed virtual router and use the vrr.ova package to deploy Cisco IOS XRv 9000 Router as virtual Route Reflector, a high scale route reflector.

For information on how to install Cisco IOS XRv 9000 Router using OVA file, see the section Installing the Cisco IOS XRv 9000 Router to the VM using OVA Template , on page 10.

Installation using ISO file

For information on how to install Cisco IOS XRv 9000 Router using ISO file, see the section *Deploying the Cisco IOS XRv 9000 Router to the VM using ISO Template* .

Installing the Cisco IOS XRv 9000 Router to the VM using OVA Template

The following procedure provides a general guideline for how to deploy the Cisco IOS XRv 9000 Router. However, the exact steps that you need to perform may vary depending on the characteristics of your VMware environment and setup.

If VM's configurations such as memory, CPU, and NICs are modified, then the Cisco IOS XRv 9000 Router must be rebooted for the changes to take effect.

Before you begin

Make sure that:

- The vSphere Client is installed on your machine.
- You have set the correct Firewall Options to allow VM Serial port to be connect over network.

Procedure

Step 1 In the vSphere client, select **File** > **Deploy OVF Template**.

Step 2 Select the location where the Cisco IOS XRv 9000 Router .ova file is stored and click **Next**.

Step 3 Verify OVF template details, and click **Next**.

Step 4 Specify the name of the VM, select Inventory Location, and click **Next**.

Step 5 (Cisco IOS XR Release 5.4 and later) Select a hardware deployment configuration from the drop-down, and click **Next**.

Step 6 Select the datastore for the VM files, and click **Next**.

Step 7 Select the format in which virtual disks are stored, and click **Next**.

These are the disk formats you can choose:

- Thick Provision Lazy Zeroed
- Thick Provision Eager Zeroed
- Thin Provisioned

Note
The Thick Provision Eager Zeroed option takes longer duration to install but provides better performance. Thick Provisioned will also consume more physical space on the disk.

Step 8 Under **Network Mapping**, allocate one or more virtual network interface card (vNIC) on the destination network using the drop-down list.

For information on interface mapping, see the Mapping the Router Network Interfaces to VM Network Interface Cards, on page 4.

Step 9 In the **Virtual Machine Properties** window configure the properties for the VM if available. This will vary depending on release.

Step 10 Under the VM Options tab, select the boot option as EFI.

- By default, the boot type will be set to Legacy BIOS. Change it to EFI if required.

Step 11 Select **Power on after deployment** to automatically power on the VM.

Step 12 Click **Finish** to deploy the OVA.

The OVA deploys the .iso file and, if the **Power on after deployment** setting is selected, automatically powers on the VM. When the VM is powered on, Cisco IOS XRv 9000 Router begins the installation and boot process. If a bootstrap configuration file was included in the OVA, the router configuration is automatically enabled. For information on bootstrap configuration file, see the CVAC - Bootstrap Configuration Support, on page 68

What to do next

Access the console on the Cisco IOS XRv 9000 Router. For details see the Console Mapping, on page 45 section.

Installing the Cisco IOS XRv 9000 Router to the VM using ISO Template

The following procedure provides a general guideline for how to deploy Cisco IOS XRv 9000 Router using VMware vSphere. However, the exact steps that you need to perform may vary depending on the characteristics of your VMware environment and setup. The steps in this procedure are based on VMware ESXi 5.5 version.

Starting with Cisco IOS XR Software Release 25.2.1, the XRv9K image can be launched on an ESXi 8.x or later hypervisor using the ESXi Host Portal. This support was initially introduced in Release 24.4.1 for ESXi 7.x. For flashing and loading the OE image, you need to do an extra configuration in the VM options. The default boot type needs to be changed from legacy BIOS to EFI. The commands to create the cd-rom EFI image is mentioned in the EFI standard iso from IOSXR build iso image section. Also for vrr,vga, and vpe image launch, you can directly use the iso image from the PIMS portal as it is already in the EFI standard format.

Before you begin

Make sure that:

- The vSphere Client is installed on your machine.
- You have set the correct Firewall Options to allow VM serial port to be connect over network.

Procedure

Step 1 Download the Cisco IOS XRv 9000 Router ISO file and copy the file to the VM Datastore.

Note
Use the _vga version of ISO file to map the XR console to VGA console, else the XR console will be mapped to the first serial port. See Mapping the Router Network Interfaces to VM Network Interface Cards, on page 4.

Step 2 In the vSphere client, select **Create a New Virtual Machine**.

Step 3 Under Configuration, select **Create a Custom configuration**, and click **Next**.

Step 4 Specify the name of the VM, and click **Next**.

Step 5 Under Storage, select the datastore for the VM files, and click **Next**.

Step 6 Select Virtual Machine version 8, and click **Next**.

Note
Cisco IOS XRv 9000 Router is not compatible with VMware ESXi Server versions prior to 5.0.

Step 7 Select Linux and the **Other Linux (64-bit) setting** from the drop-down, and click **Next**.

Step 8 Under **CPUs**, select the following settings:

- Number of virtual sockets (virtual CPUs)
- Number of cores per socket

The number of cores per socket must always be set to 1, regardless of the number of virtual sockets selected. For example, a Cisco IOS XRv 9000 Router with the 4 vCPU configuration must be configured as 4 sockets and 1 core for each socket.

For information on the supported number of virtual CPUs and the corresponding required RAM allocation for your release, see the Installation Requirements for VMware ESXi, on page 9 section .

Click **Next**.

Step 9 Configure the VM's memory size. Click **Next**.

Note
The supported memory size is 64 GB for the default image and a minimum of 115 GB for VRR scenarios.

Step 10 Under **Network**, allocate at least 4 virtual network interface cards (vNICs).

a. From the drop-down select the number of vNICs.

Note
The VMware ESXi 5.5 interface allows for creating 4 vNICS during the initial VM creation. You can add more vNICs after the VM is created and Cisco IOS XRv 9000 Router is first booted.

b. Add the vNICs.

- Select a different network for each vNIC. Note the 2nd and 3rd NICs are reserved.
- Select the adapter type from the drop-down menu. Select E1000 adapter for the first three NICs and later you may select physical interfaces (passthrough) or VMXNET3. For information on the supported adapter type in your release, see the Installation Requirements for VMware ESXi, on page 9 section.

  Note
  From Release 6.0, VMXNET3 NIC is supported.

c. Select all vNICs to connect at power-on, and click **Next**.

Note
You can add vNICs into the VM using vSphere while Cisco IOS XRv 9000 is running. For more information about adding vNICS to an existing VM, see the vSphere documentation.

Step 11 Under the VM Options tab, select the boot option as EFI.

- By default, the boot type will be set to Legacy BIOS. Change it to EFI if required.

Step 12 Under **SCSI Controller**, select **LSI Logic Parallel**, and click **Next**.

Step 13 Under **Select a Disk**, click **Create a new virtual disk**.

Step 14 Under **Create a Disk**, select the following:

- Capacity: Disk Size

  See the Installation Requirements for VMware ESXi, on page 9 section for information on the virtual hard disk size required in your release.

- Disk Provisioning

  Select one of the following:

- Thick Provision Lazy Zeroed
- Thick Provision Eager Zeroed
- Thin Provisioned

Note
The Thick Provision Eager Zeroed option takes a longer duration to install but provides better performance. Thick Provisioned also consumes more physical space on the disk.

- Location

  Store with the virtual machine.

Click **Next**.

Step 15 Under **Advanced Options**, select **IDE (0:0)** for the virtual device node.

Step 16 On the **Ready to Complete** screen, select **Edit the virtual machine settings before completion.**

Step 17 Click **Continue checkbox**.

Step 18 Click **New CD/DVD Drive** and do the following:

a. Select the Device Type from which the VM boots:

Select **Datastore ISO file** option to boot from the Cisco IOS XRv 9000 .iso file. Browse to the location of the .iso file on the datastore set in Step 1.

b. In the **Device Status** field, select **Connect at power on** checkbox.

c. Select the **Virtual Device Node CD/DVD** drive on the host from which the VM boots.

Step 19 To add a serial port (Console Port), click **Add** under the Hardware tab.

Note
Setting up the serial ports is required for a non-vga (default) image.

For information on configuring serial console access, see the Configuring Serial Console Access in VMware ESXi , on page 47

Step 20 Select **Serial Port** and click **Next**.

Step 21 Select **Connect via Network** and click **Next**.

Step 22 Select **Server** and add a telnet address of the host and an unused port higher than 1000. Click **Next**.

Step 23 On Ready to Complete screen, click **Finish**.

Step 24 Repeat Step 17 till Step 22 to add three serial ports. The three ports are XR auxiliary port, admin port, and admin auxiliary port.

Step 25 In the **Resources** tab, click the **CPU setting** and set the **Resource Allocation** setting to **Unlimited**.

Step 26 Click **OK**.

Step 27 Click **Finish**.

The VM is now configured for Cisco IOS XRv 9000 Router and is ready to boot. Cisco IOS XRv 9000 is booted when the VM is powered on.

What to do next

To access and configure Cisco IOS XRv 9000 Router from the serial port on the ESXi host instead of the VM console, provision the VM to use this setting before powering on the VM and booting the router. For more information, see the Console Mapping, on page 45 section.

Note By default the XR console is mapped to the first serial port of the VM. However, if a VGA image is used for Cisco IOS XRv 9000 Router deployment, the XR console is mapped to the VGA console. The VGA console is accessed at the Console Tab in the vSphere Client. On ESXi, VGA console is console which opens by itself once the installation is complete. You don't need any specific procedure to connect to VGA console.

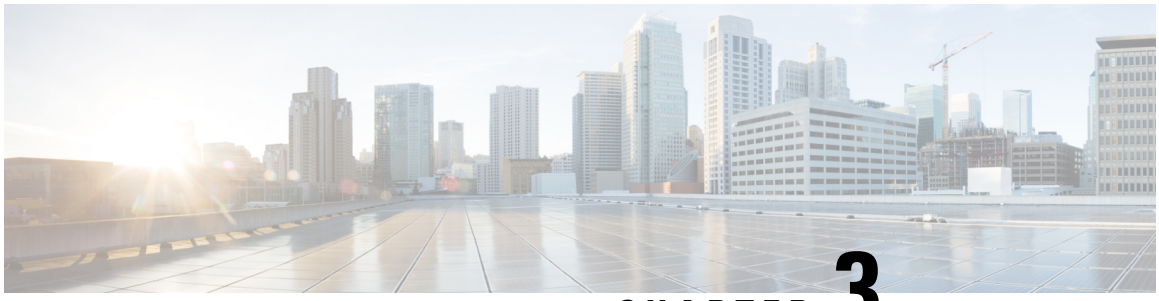
Increasing Performance on VMware ESXi Configurations

You can improve performance on VMware ESXi configurations by performing the following:

- Disable VMware ESXi power management.

  Choose the High Performance setting to disable power management in VMware ESXi. For more information, see the VMware Documentation.

The downside of improving performance on VMware ESXi is that it requires dedicated system resources.



CHAPTER 3

Installing the Cisco IOS XRv 9000 Router in KVM Environments

These file types are needed to install Cisco IOS XRv 9000 Router on the KVM hypervisor:

- .qcow2—Used for booting the software image in KVM environments. The qcow2 disk image has an instance of the Cisco IOS XRv 9000 Router pre-installed.
- .iso and .qcow2—Used to manually create the Cisco IOS XRv 9000 Router VM using Virsh application. You must also have virsh.xml file that has sample XML configuration used to launch the Cisco IOS XRv 9000 Router in KVM environments using Virsh commands.

- Installation Requirements for KVM, on page 17
- Installing the Cisco IOS XRv 9000 Router Using KVM Command Line , on page 18
- Installing the Cisco IOS XRv 9000 Router in KVM Using Virsh, on page 20
- Creating the Cisco IOS XRv 9000 Router KVM Instance on OpenStack , on page 22
- Configure PCI Passthrough, on page 25
- Increasing Performance on KVM Configuration, on page 28

Installation Requirements for KVM

Cisco IOS XRv 9000 Router is supported on top of Ubuntu and Red Hat Distrutions using the Kernel Virtual Machine (KVM). The Cisco IOS XRv 9000 Router installation on KVM requires the creation of a VM and installation using the either .iso file or .qcow2 image. The VM can be launched either using the KVM command line, Virsh, or OpenStack.

For information on the installation requirements for KVM, refer to the latest *Release Notes for Cisco IOS XRv 9000 Router.*

Note

- Ensure your VM configuration aligns with the release-specific storage requirements:
  - IOS-XR software release 6.0 to 25.1.x: Minimum 45 GB
  - IOS-XR software release 25.2.1 and later: Minimum 64 GB
- The **du** command is used to check the real amount of disk space consumed by the virtual disk image.
- The option **intel_iommu=on**, **iommu=pt**, and **iommu_default_dma_mode=1** command must be set in the grub configuration for PCIe-passthrough interfaces to work in a VM.
- The CPU flags must be passed into the VM and must include the **sse4_2** flag in order for the embedded Dataplane to work.

For information on KVM support on OpenStack, see the section KVM support on OpenStack.

Installing the Cisco IOS XRv 9000 Router Using KVM Command Line

This procedure provides a general guideline for manually creating the VM for the Cisco IOS XRv 9000 router; the exact steps that you need to perform may vary depending on the characteristics of your KVM environment and setup. For more information, see the Red Hat Linux or Ubuntu documentation.

Procedure

This procedure explains how to create ISO boot configuration with 3 traffic interfaces and 3 required interfaces (one is for XR management, and two are reserved).

a. `/usr/bin/kvm \`

Invokes the KVM

b.
```
-smbios type=1,manufacturer="cisco",product="Cisco IOS XRv
9000",uuid=97fc351b-431d-4cf2-9c01-43c283faf2a3 \
```

Sets Universally Unique Identifier (UUID) for the VM instance. The UUID is used as part of licensing to identify VM instance.

c. `-cpu host \`

Pass host CPU flags into guest.

d.
```
-drive file=/home/username/bnbMay13/workdir-username/disk1.raw,if=virtio,media=disk,index=1 \
-drive file=/home/username/bnbMay13/workdir-username/xrv9k-fullk9-x.iso.baked,media=cdrom,index=2
 \
```

Empties harddisk and boots ISO.

e. `-m 16384 \`

Creates memory.

f. `-smp cores=4,threads=1,sockets=1 \`

Creates four virtual CPUs and one socket.

g. `-enable-kvm \`

Enables hardware acceleration.

h. `-display none \`

Emulates VGA console when using serial ports for console access. This step is recommended.

i. `-rtc base=utc \`

Sets the real time clock (RTC).

j.

```
-netdev tap,id=host1,ifname=usernameLx1,script=no,downscript=no \
-netdev tap,id=host2,ifname=usernameLx2,script=no,downscript=no \
-netdev tap,id=host3,ifname=usernameLx3,script=no,downscript=no \
-device virtio-net-pci,romfile=,netdev=host1,id=host1,mac=52:46:84:57:A0:DA \
-device virtio-net-pci,romfile=,netdev=host2,id=host2,mac=52:46:C4:F4:36:0F \
-device virtio-net-pci,romfile=,netdev=host3,id=host3,mac=52:46:A5:C0:D0:C5 \
```

Creates 3 NICs; the first is mapped to XR management interface, the second and third are reserved.

k.

```
-netdev tap,id=data1,ifname=usernameXr1,script=no,downscript=no \
-netdev tap,id=data2,ifname=usernameXr2,script=no,downscript=no \
-netdev tap,id=data3,ifname=usernameXr3,script=no,downscript=no \
-device e1000,romfile=,netdev=data1,id=data1,mac=52:46:87:18:62:DF \
-device e1000,romfile=,netdev=data2,id=data2,mac=52:46:32:02:90:6F \
-device e1000,romfile=,netdev=data3,id=data3,mac=52:46:34:93:52:1F \
```

The fourth to eleventh NICs are mapped to traffic ports. A minimum of one traffic interface is recommended.

l.

```
-serial telnet:0.0.0.0:10621,nowait,server \
-serial telnet:0.0.0.0:14713,nowait,server \
-serial telnet:0.0.0.0:18090,nowait,server \
-serial telnet:0.0.0.0:17181,nowait,server \
```

Creates four serial ports to access console. The first serial port is mapped to XR console. See Console Mapping section for additional information. A minimum of one serial port is recommended.

For information on configuring the serial console access, see Configuring the Serial Console Access in KVM using QEMU , on page 49 section.

m. `-boot once=d &`

Boot the ISO only once.

Example:

```
/usr/bin/kvm \
      -smbios type=1,manufacturer=\"cisco\",product=\"Cisco IOS XRv
9000\",uuid=73c8134e-2f51-40c1-a8d5-27687d717857 \
      -cpu host \
      -drive file=/root/gorajama/workdir-root/disk1.raw,if=virtio,format=raw,media=disk,index=1 \
      -drive if=pflash,format=raw,readonly,file=OVMF_CODE.fd \
      -drive if=pflash,format=raw,file=OVMF_VARS.fd \
      -drive file=/root/gorajama/workdir-root/xrv9k-full-x-24.4.2.14I.iso.baked,media=cdrom,index=2
```

```
 \
      -m 20480 \
      -smp cores=4,threads=1,sockets=1 \
      -enable-kvm \
      -daemonize \
      -rtc base=utc \
      -name IOS-XRv-9000:root \
      -netdev tap,id=host1,ifname=rootLx1,script=no,downscript=no \
      -netdev tap,id=host2,ifname=rootLx2,script=no,downscript=no \
      -netdev tap,id=host3,ifname=rootLx3,script=no,downscript=no \
      -device
virtio-net-pci,romfile=,netdev=host1,id=host1,bus=pci.0,mac=52:46:28:2B:47:AC,csum=off,guest_csum=off
 \
      -device
virtio-net-pci,romfile=,netdev=host2,id=host2,bus=pci.0,mac=52:46:C2:A9:4B:D7,csum=off,guest_csum=off
 \
      -device
virtio-net-pci,romfile=,netdev=host3,id=host3,bus=pci.0,mac=52:46:B1:65:5B:40,csum=off,guest_csum=off
 \
      -netdev tap,id=data1,ifname=rootXr1,script=no,downscript=no \
      -netdev tap,id=data2,ifname=rootXr2,script=no,downscript=no \
      -netdev tap,id=data3,ifname=rootXr3,script=no,downscript=no \
      -device e1000,romfile=,netdev=data1,id=data1,bus=pci.0,mac=52:46:6D:DA:1D:EA \
      -device e1000,romfile=,netdev=data2,id=data2,bus=pci.0,mac=52:46:73:D7:9E:53 \
      -device e1000,romfile=,netdev=data3,id=data3,bus=pci.0,mac=52:46:77:F6:B5:03 \
      -monitor telnet:0.0.0.0:16908,server,nowait \
      -serial telnet:0.0.0.0:13775,nowait,server \
      -serial telnet:0.0.0.0:13329,nowait,server \
      -serial telnet:0.0.0.0:18865,nowait,server \
      -serial telnet:0.0.0.0:14514,nowait,server \
      -boot once=d
```

Note
The disk labeled disk1.raw in the example above can be created with **qemu-img**. The **qemu-img** is an utility to convert the virtual hard disk format. Instead of a raw disk format, qcow2 disk format can be used in above example.

A preinstalled qcow2 image can also be used; in that case the cdrom parameter is removed.

You can also use PCI Passthrough to assign hardware NICs to the VM. Please see Configure PCI Passthrough for detailed steps.

Installing the Cisco IOS XRv 9000 Router in KVM Using Virsh

This procedure provides a general guideline for manually creating the VM for the Cisco IOS XRv 9000; the exact steps that you need to perform may vary depending on the characteristics of your KVM environment and setup. For more information, refer the Red Hat Linux, Ubuntu and Virsh documentation.

Before you begin

The VM memory ballooning is not supported and hence the 'memory' and 'currentmemory' unit values (as shown below) must be the same in Virsh file.

```
<memory unit='MB'>XXX</memory>
<currentMemory unit='MB'>XXX</currentMemory>
```

Procedure

Step 1 From Cisco.com download the .iso or .qcow2 image along with the sample Virsh XML file.

Step 2 Edit the XML file to point to the .iso or .qcow2 image, and edit the interface sources to designate the connectivity desired.

```
<!-- CDROM HDB Disk -->
     <disk type='file' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <source file='/home/<username>/xrv9k-fullk9-x.iso'/>
      <target dev='hdc' bus='ide'/>
      <alias name='ide-cdrom'/>
     </disk>
```

Step 3 If qcow2 image is used, comment out the CDROM section.

Step 4 If iso image is used, then:

a. Create an empty qcow2 disk:

```
qemu-img create -f qcow2 xrv9000.qcow2 45G
```

Starting from IOS-XR release 25.2.1 minimum requirement is 64G

```
qemu-img create -f qcow2 xrv9000.qcow2 64G
```

Note
The minimum supported VM hard disk size is 64GB.

b. Edit the HDA Disk section in the XML to point to the empty qcow2 disk just created:

```
<!-- HDA Disk -->
    <disk type='file' device='disk'>
     <driver name='qemu' type='qcow2'/>
     <source file='/home/<username>/xrv9000.qcow2'/>
     <target dev='vda' bus='virtio'/>
     <alias name='virtio-disk0'/>
    </disk>
```

Step 5 Edit the Path of OVMF Binaries Based on the OS

Modify the OVMF Code and VARS files under the OS block to specify the BIOS binaries for UEFI boot with OVMF

Use the default paths based on your operating system:

- For Ubuntu: `/usr/share/OVMF/`
- For RHEL: `/usr/share/edk2/ovmf/`

Step 6 Create the Virsh domain.

```
virsh create xrv9k-fullk9-x.virsh.xml
```

Step 7 Validate the Virsh domain created in Step 7:

```
virsh list xrv9k-fullk9-x.virsh.xml
Id    Name                           State
----------------------------------------------------
```

```
 149   IOS-XRv-9000_username_virsh        running
```

Step 8 Use the Virsh commands to manage the VM, such as:

- **reboot**

```
virsh reboot IOS-XRv-9000_USER1_virsh
```

- **shutdown**

```
virsh shutdown IOS-XRv-9000_USER1_virsh
```

- **destroy**

```
virsh destroy IOS-XRv-9000_USER1_virsh
```

For information on configuring the serial console access, see Configuring the Serial Console Access in KVM using Virsh , on page 48 section.

See Virsh command reference documentation for more details.

You can also use PCI Passthrough to assign hardware NICs to the VM. Please see Configure PCI Passthrough for detailed steps.

Creating the Cisco IOS XRv 9000 Router KVM Instance on OpenStack

This procedure shows how to bring the Cisco IOS XRv 9000 Router up on the RHEL 7 and later version and OpenStack 5 and later version . The procedure uses the OpenStack command line interface. It is expected that the reader is familiar with OpenStack commands and procedures. See the OpenStack documentation for further information.

At the end of the procedure, you will have a single Cisco IOS XRv 9000 Router running with 3 data plane interfaces connected to 3 neutron networks.

Note In case trunk interfaces are required, please use Neutron ML2 core plugin with Nexus 1kv mechanism plugin. The following procedure creates only Access (non-tagging) type of interfaces.

Before you begin

You must have:

- OpenStack 5, 6 or 7 with Neutron ML2 core plugin, Open vSwitch as a mechanism plugin and VLAN as Tenants network type.
- Cisco IOS XRv 9000 Router ISO image (VGA type).

Procedure

Step 1 **Image preparation**

For Cisco IOS XRv 9000 Router deployment on OpenStack you need:

- Cisco IOS XRv 9000 ISO image (VGA type)
- a blank 45GB cinder volume (virtual hard disk)

Note
The minimum supported VM hard disk size is 45GB for release 6.0 and 55GB for release 5.4.

When the ISO image is booted, the Cisco XRv 9000 Router gets installed on the second disk (blank 45GB volume). Later, the router can be booted from the created cinder volume. Optionally, a pre-installed Cisco XRv 9000 qcow2 disk can be downloaded in place of the ISO image.

a. Cisco IOS XRv 9000 ISO image (VGA type) must be imported into OpenStack glance. Use the following command line to import the image into glance:

```
glance image-create --name xrv9k-fullk9_vga-x --disk-format iso --container bare \
--file xrv9k-fullk9_vga-x.iso
```

b. Verify the image in OpenStack glance:

```
 glance image-list
+--------------------------------------+---------------------------------------------+----------+--------------+----------+--------+
| ID                                   | Name                                        | Disk Format |
 Container Format | Size       | Status |
+--------------------------------------+---------------------------------------------+----------+--------------+----------+--------+
| 71b44355-32a8-45e7-abfd-f52593f2dc1a | csr1000v-universalk9.03.15.00.S.155-2.S | qcow2       |
 bare             | 1335427072 | active |
| e779245a-9491-4bee-bc85-a73e9394b981 | xrv9k-fullk9_vga-x                          | iso         |
 bare             | 775370752  | active |
| 3b3ade31-fae6-4354-9902-fb77452a65ab | xrvr-initial-config                         | iso         |
 bare             | 358400     | active |
+--------------------------------------+---------------------------------------------+----------+--------------+----------+--------+
```

Step 2 **Cinder volumes preparation**

Use the following command line to create Cinder volumes:

a. Create Cinder volume of Cisco IOS XRv 9000 Router's disk image and make it bootable:

```
cinder create --display-name xrv9k-disk 45
cinder set-bootable volume-id True
```

The **cinder list** command displays the volume ID of router's ISO image.

b. Check Cinder volumes:

```
cinder list
+--------------------------------------+--------+--------------+------+-------------+----------+-------------------------------+
|                  ID                  | Status |    Display Name    | Size | Volume Type |
```

```
Bootable |             Attached to              |
+--------------------------+--------+-------------+----+---------+-------+-------------------------+
|
| cbef86dd-9819-4daa-81b2-4b905b287974 | Available |    xrv9k-disk     |  45  |     None    |
true   | 5262e1fe-37f5-4535-bf76-aab26cb86366 |
+--------------------------+------+-------------+----+---------+-----------------------------------
```

You must see Cinder volume with name xrv9k-disk with **Status-Available** and **Bootable-True.**

Step 3 **Configuring Nova Flavor**

The virtual hardware templates are called **flavors** in OpenStack. The **nova flavor-create** command allows users to create new flavors. The nova's flavor is used to pass the information about RAM size, disk, and number of cores to the Nova scheduler process when new instances of VM are launched.

The Cisco IOS XRv 9000 Router requires 16GB of RAM, 45GB hard disk, and 4 vCPUs.

```
nova flavor-create xrv9k-flavor auto 16384 45 4
```

xrv9k-flavor is flavor's name.

Step 4 **Configuring network**

Cisco IOS XRv 9000 Router requires a minimum of 4 network interfaces. The first NIC or vNIC is mapped with the XR management interface, the second and third NICs are reserved, and the remaining NICs are mapped to traffic interfaces. For more information on interface mapping, see the Mapping the Router Network Interfaces to VM Network Interface Cards section.

For a sample network configuration, we can have 6 neutron networks, 6 neutron subnetworks and then pass 6 network IDs as parameters to VM instance. This equates to 3 traffic interfaces. Use **neutron net-create <*neutron-network-name*>** to create networks in Neutron.

For example:

```
neutron net-create management-xr
neutron net-create management-other
neutron net-create management-host
neutron net-create datalink-1
neutron net-create datalink-2
neutron net-create datalink-3
```

In the above example, first three commands create control plane networks, and last three commands create data plane networks.

Now using corresponding network names, create subnetworks in Neutron. Use **neutron subnet-create <*neutron-network-name*> <*IP-subnet*> --name <*neutron-network-name*>** command. For consistency, keep the neutron names and subnetwork names same.

For example:

```
neutron subnet-create management-xr  10.50.70.0/26 --name management-xr
neutron subnet-create management-other  10.50.70.64/26 --name management-other
neutron subnet-create management-host  10.50.70.128/26 --name management-host
neutron subnet-create datalink-1  10.57.11.0/24 --name datalink-1
neutron subnet-create datalink-2  10.57.12.0/24 --name datalink-2
neutron subnet-create datalink-3  10.57.13.0/24 --name datalink-3
```

Step 5 **Attach management-xr and management-host subnets to the neutron router**

Use these commands to attach the management-xr and management-host subnets to the neutron router:

```
neutron router-interface-add <Neutron router name> <subnet id of management-xr>
neutron router-interface-add <Neutron router name> <subnet id of management-host>
```

Note

- This step must be performed before launching the Cisco IOS XRv 9000 Router to avoid DHCP issues.
- The **neutron router-list** command displays list of Neutron router name.
- The **neutron subnet-list** command displays list of Neutron subnet.

Step 6 Launch the Cisco IOS XRv 9000 Router

Follow this command line to boot the router.

```
nova boot
 --flavor xrv9k-flovor \
 --nic net-id={Control plane network ID of management-xr} \
 --nic net-id={Control plane network ID of management-other} \
 --nic net-id={Control plane network ID of management-host} \
 --nic net-id={Data plane network ID of datalink-1} \
 --nic net-id={Data plane network ID of datalink-2} \
 --nic net-id={Data plane network ID of datalink-3} \
 --block-device id={glance ID of Cisco IOS XRv 9000 router's CD volume available in Step 1},\
 --source=image,dest=volume,bus=ide,device=/dev/hdc,size=1,type=cdrom,bootindex=1 \
 --block-device source=volume,id={cinder ID of Cisco IOS XRv 9000 Router's disk volume available in
 Step 1}, dest=volume,size=50,bootindex=0 xrv9k-1
nova list (your instance should be in Active state)
nova get-vnc-console xrv9k-1 novnc
```

The **nova get-vnc-console** command returns an URL that is used to access the Cisco IOS XRv 9000 Router console.

Note

The **config-drive** allows the VM to boot with an initial configuration so that when the VM is powered on, it can perform services, configured in the config-drive. As the Cisco IOS XRv 9000 Router is running on the VM, the router accepts a valid XR configuration through a plain text file. During the boot process the text file is parsed by the command parser, just as if the commands were entered through CLI.

If **config-drive** support is desired, a plaintext file with the initial XR configuration can be passed by adding this line under **nova boot** command:

```
config-drive true user-data <path>/iosxr_config.txt file /iosxr_config.txt=<path>/iosxr_config.txt
```

For information on Cisco Virtual Appliance Configuration (CVAC), see section CVAC - Bootstrap Configuration Support.

Configure PCI Passthrough

Configure PCI passthrough for Cisco IOS XRv9000 routers, whether you install them using the KVM command line or the VM's XML configuration.

Procedure

Step 1 Enable input-output memory management unit (IOMMU) on the host.

a) Edit the GRUB command line to include these parameters:

- intel_iommu=on
- iommu=pt
- iommu_default_dma_mode=1

Example:

```
GRUB_CMDLINE_LINUX="crashkernel=1GB-4GB:192M,4GB-64GB:256M,64GB and above:512M
resume=/dev/mapper/vg0-swap rd.lvm.lv=vg0/root rd.lvm.lv=vg0/swap rhgb quiet intel_iommu=on iommu=pt
 iommu_default_dma_mode=1"
```

b) Rebuild and reboot the GRUB command.

Example:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
reboot
```

c) Verify the change using the **cat/proc/cmdline** updates.

Step 2 After the system reboots, verify that IOMMU is enabled.

Step 3 Bind or Run the PCI device to the `vfio-pci` driver.

If..	Then..
You are using KVM Command Line	Run the **vfio-pci** command.
You are using KVM Virsh	Bind the PCI device to the **vfio-pci** driver.

a) Check which driver is currently bound to the device. Note the PCI address.

Example:

```
virsh nodedev-dumpxml pci_0000_d8_00_0
```

b) If required, unbind the device from its current driver and bind it to `vfio-pci`.

Example:

```
echo '0000:d8:00.0' > /sys/bus/pci/devices/0000:d8:00.0/driver/unbind
      echo '8086 1583' > /sys/bus/pci/drivers/vfio-pci/new_id
      echo '0000:d8:00.0' > /sys/bus/pci/drivers/vfio-pci/bind
```

c) For multiple devices, additional PCI device, repeat these steps.

Step 4 Add PCI passthrough.

If you are using VM's XML:

a) Open the VM’s XML file or use virsh edit *vr-name*. In the devices section, add a block for each PCI device

Example:

```
xml
      <hostdev mode='subsystem' type='pci' managed='yes'>
        <source>
```

```
      <address domain='0x0000' bus='0xd8' slot='0x00' function='0x0'/>
    </source>
  </hostdev>
```

b) Repeat the block for additional devices for example, `function='0x1'`.

If you are using KVM Command Line:

a) Fetch PCI ID of the NIC by using **lspci** command.
b) Use the **device vfio-pci,host=xx.x** and add the PCI-ID of the NIC.

```
/usr/libexec/qemu-kvm \
device vfio-pci,host=d8:00.0 \
device vfio-pci,host=d8:00.1 \
smbios type=1,manufacturer="cisco",product="Cisco IOS XRv
9000",uuid=bcb40603-510e-4ba4-bfdf-fa880cf9dcd0 \
cpu host \
```

Step 5 If you are using virsh, do one of the following to apply the updated VM definition and start the VM.

- Create or update the VM definition.

  ```
  virsh define /path/to/your-updated-vm.xml
  ```

- Start the VM.

  ```
  virsh create /path/to/your-updated-vm.xml
  ```

- Use a persistent VM.

  ```
  virsh start vm-name
  ```

Step 6 Verify PCI passthrough inside the VM.

Example:

Log in to the VM's shell and run the **lspci | grep-i ethernet** command.

```
[virssh]# lspci -nn | grep -i ethernet
3b:00.0 Ethernet controller [0200]: Intel Corporation Ethernet Controller X550 [8086:1563] (rev 01)
3b:00.1 Ethernet controller [0200]: Intel Corporation Ethernet Controller X550 [8086:1563] (rev 01)
5e:00.0 Ethernet controller [0200]: Intel Corporation Ethernet Controller E810-C for SFP [8086:1593]
 (rev 02)
5e:00.1 Ethernet controller [0200]: Intel Corporation Ethernet Controller E810-C for SFP [8086:1593]
 (rev 02)
5e:00.2 Ethernet controller [0200]: Intel Corporation Ethernet Controller E810-C for SFP [8086:1593]
 (rev 02)
5e:00.3 Ethernet controller [0200]: Intel Corporation Ethernet Controller E810-C for SFP [8086:1593]
 (rev 02)
d8:00.0 Ethernet controller [0200]: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+
[8086:1583] (rev 02) → XL710 NIC
d8:00.1 Ethernet controller [0200]: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+
[8086:1583] (rev 02) → XL710 NIC
```

The specified PCI device(s) are now directly accessible inside the VM. This enables advanced network use cases (such as LLDP, SR-IOV, or direct Layer 2 packet operations) that are not possible with default virtual interfaces.

What to do next

- Install or configure drivers for the passed-through NIC or GPU inside the VM, if required.

- Validate connectivity and desired features, for example, LLDP neighbor formation, direct packet capture.
- Optionally, check VM logs for any error or warning related to device initialization.

Increasing Performance on KVM Configuration

You can increase the performance for a Cisco IOS XRv 9000 Router in the KVM environment by changing settings on the KVM host. These settings are independent of the Cisco IOS XR configuration settings on the router. This option is available in Red Hat Enterprise Linux 7.0 KVM.

You can improve performance on KVM configurations by enabling CPU pinning.

Note The Cisco IOS XR Software Release 5.4 does not support jumbo packets larger than 1518 bytes for KVM on a VirtIO interface. The packets larger than 1518 bytes are dropped.

Enabling CPU Pinning

To increase the performance of the Cisco IOS XRv 9000 Router in KVM environments, you can use the KVM CPU affinity option to assign a VM to a specific processor. To use this option, configure CPU pinning on the KVM host.

Follow these steps to configure CPU pinning:

1. In the KVM host environment, verify the host topology to find out how many vCPUs are available for pinning:

   ```
   virsh nodeinfo
   ```

2. Verify the available vCPU numbers:

   ```
   virsh capabilities
   ```

3. Pin the vCPUs to sets of processor cores:

   ```
   virsh vcpupin <vm-name> <vcpu-number> <host-core-number>
   ```

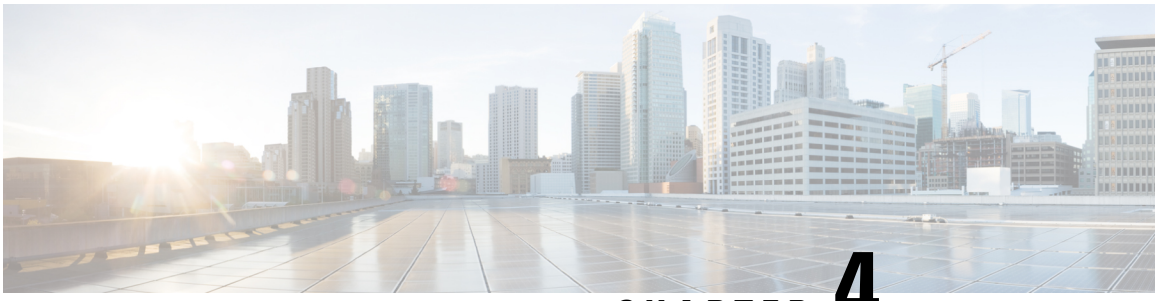
   The **virsh vcpupin** command must be executed for each vCPU on your Cisco IOS XRv 9000 Router. The following example shows the KVM commands needed if you have a Cisco IOS XRv 9000 Router configuration with four vCPUs and the host has eight cores:

   ```
   virsh vcpupin xrv9000 0 2
   virsh vcpupin xrv9000 1 3
   virsh vcpupin xrv9000 2 4
   virsh vcpupin xrv9000 3 5
   ```

   The host core number can be any number from 0 to 7. For more information, see the KVM documentation.

Note When configuring CPU pinning, carefully consider the CPU topology of the host server. If using a Cisco IOS XRv 9000 Router configured with multiple cores, do not configure CPU pinning across multiple sockets.

The downside of improving performance on KVM configuration is that it requires dedicated system resources.



CHAPTER 4

Customize Installation using Golden ISO

Table 1: Feature History Table

Feature Name	Release Information	Description
Golden ISO (GISO)	Release 7.3.1	Golden ISO is a customizable .iso image file that contains the deployment-ready minimal software image, base configuration files and packages that can be installed using iPXE or system upgrade. The GISO file can be created to include the optional packages and SMUs based on requirement. This capability is now supported on the Cisco IOS XRv 9000 router.

Golden ISO (GISO) is a customized ISO that a user can build to suit the installation requirement. The user can customize the installable image to include the standard base image with the basic functional components, and add additional RPMs, SMUs and configuration files based on requirement.

The ease of installation and the time taken to seamlessly install or upgrade a system plays a vital role in a cloud-scale network. An installation process that is time-consuming and complex affects the resiliency and scale of the network. The GISO simplifies the installation process, automates the installation workflow, and manages the dependencies in RPMs and SMUs automatically.

GISO is built using a build script `gisobuild.py` available on the github location Github location. For more information about the build script and the steps to build GISO, see Build Golden ISO, on page 30.

When a system boots with GISO, additional SMUs and RPMs in GISO are installed automatically, and the router is pre-configured with the XR configuration in GISO. For more information about downloading and installing GISO, see Install Golden ISO, on page 33.

The capabilities of GISO can be used in the following scenarios:

- Migration from IOS XR 32-bit to IOS XR 64-bit
- Initial deployment of the router
- Software disaster recovery

- System upgrade from one base version to another
- System upgrade from same base version but with additional SMUs
- Install update to identify and update dependant packages

- Limitations, on page 30
- Golden ISO Workflow, on page 30
- Build Golden ISO, on page 30
- Install Golden ISO, on page 33
- Install Replace with Golden ISO, on page 34

Limitations

The following are the known problems and limitations with the customized ISO:

- Building and booting GISO for asynchronous package (a package of different release than the ISO) is not supported.
- Verifying the XR configuration is not supported in the GISO build script `gisobuild.py`.
- Renaming a GISO build and then installing from the renamed GISO build is not supported.
- The Cisco IOS XRv 9000 Router does not support GISO creation with optional files like custom configuration and ZTP.ini.
- The Cisco IOS XRv 9000 Router does not support GISO creation for XRv9K vRR profile.
- Automatic GISO post build scripts are discontinued.

Golden ISO Workflow

The following image shows the workflow for building and installing golden ISO.

Build Golden ISO

The customized ISO is built using Cisco Golden ISO (GISO) build script `gisobuild.py` available on the Github location.

GISO Tool Link: Golden ISO Creation for SMUs

The GISO build script supports automatic dependency management, and provides these functionalities:

- Builds RPM database of all the packages present in package repository.
- Skips and removes Cisco RPMs that do not match the xrv9k-mini-x.iso version.
- Skips and removes third-party RPMs that are not SMUs of already existing third-party base package in mini-x.iso.
- Displays an error and exits build process if there are multiple base RPMs of same release but different versions.

- Performs compatibility check and dependency check for all the RPMs. For example, the child RPM is dependent on the parent RPM . If only the child RPM is included, the Golden ISO build fails.

To build GISO, provide the following input parameters to the script:

- Base mini-x.iso (mandatory)
- XR configuration file (optional)
- one or more Cisco-specific SMUs for host, XR and System admin (mandatory)
- one or more third-party SMUs for host, XR and System admin (mandatory)
- Label for golden ISO (optional)

Note Golden ISO can be built only from mini ISO. The full or fullk9 bundle ISO is not supported.

Use the following naming convention when building GISO:

GISO Build	Format	Example
GISO without k9sec RPM	`<platform-name>-golden-x.iso-<version>.<label>` `<platform-name>-golden-x-<version>.iso.<label>`	<platform-name>-golden-x64.iso-<version>.v1 <platform-name>-golden-x64-<version>.iso.v1
GISO with k9sec RPM	`<platform-name>-goldenk9-x.iso-<version>.<label>` `<platform-name>-goldenk9-x-<version>.iso.<label>`	<platform-name>-goldenk9-x64.iso-<version>.v1 <platform-name>-goldenk9-x64-<version>.iso.v1

Note To successfully add k9sec RPM to GISO, change the permission of the file to `644` using the **chmod** command.

```
chmod 644 [k9 sec rpm]
```

To build GISO, perform the following steps:

Before you begin

- To upgrade from non-GISO to GISO version, it is mandatory to first upgrade to mini ISO with GISO support.
- The system where GISO is built must meet the following requirements:
- - System must have Python version 2.7 and later.
  - System must have free disk space of minimum 3 to 4 GB.
  - Verify that the Linux utilities `mount`, `rm`, `cp`, `umount`, `zcat`, `chroot`, `mkisofs` are present in the system. These utilities will be used by the script. Ensure privileges are available to execute all of these Linux commands.
  - Kernel version of the system must be later than 3.16 or later than the version of kernel of Cisco ISO.

- Verify that a `libyaml rpm` supported by the Linux kernel is available to successfully `import yaml` in the tool.
- User should have proper permission for security rpm(k9sec-rpm) in rpm repository, else security rpm would be ignored for Golden ISO creation.

- The system from where the gisobuild script is executed must have root credentials.

Procedure

Step 1 Copy the script `gisobuild.py` from the Github location to an offline system or external server where the GISO will be built. Ensure that this system meets the pre-requisites described above in the *Before You Begin* section.

Step 2 Run the script `gisobuild.py` and provide parameters to build the golden ISO off the router. Ensure that all RPMs and SMUs are present in the same directory. The number of RPMs and SMUs that can be used to build the Golden ISO is 128.

Note
The `-i` option is mandatory, and either or both `-r` or `-c` options must be provided.

```
[directory-path]$ gisobuild.py [-h] [-i <mini-x.iso>] [-r <rpm repository>]
[-c <config-file>] [-l <giso label>] [-m] [-v]
```

Golden ISO label is not specified so defaulting to 0. The following example shows the script output:

where:

- -i is the path to mini-x.iso
- -r is the path to RPM repository
- -c is the path to XR config file
- -l is the golden ISO label
- -h shows the help message
- -v is the version of the build tool `gisobuild.py`
- -m is to build the migration tar to migrate from IOS XR to IOS XR 64 bit

GISO is built with the RPMs placed in respective folders in the specified directory and also includes the log files `giso_summary.txt` and `gisobuild.log-<timestamp>`. The XR configuration file is placed as `router.cfg` in the directory.

Note The GISO script does not support verification of XR configuration.

What to do next

Install the golden ISO on the router.

Install Golden ISO

Golden ISO (GISO) automatically performs the following actions:

- Installs host and system admin RPMs.
- Partitions repository and TFTP boot on RP.
- Creates software profile in system admin and XR modes.
- Installs XR RPMs. Use **show instal active** command to see the list of RPMs.
- Applies XR configuration. Use **show running-config** command in XR mode to verify.

Procedure

Step 1 Download GISO image to the router using one of the following options:

- **PXE boot:** when the router is booted, the boot mode is identified. After detecting PXE as boot mode, all available ethernet interfaces are brought up, and DHClient is run on each interface. DHClient script parses HTTP or TFTP protocol, and GISO is downloaded to the box.
- **USB boot or Disk Boot:** when the USB mode is detected during boot, and GISO is identified, the additional RPMs and XR configuration files are extracted and installed.
- **System Upgrade** when the system is upgraded, GISO can be installed using **install add**, **install activate**, or using **install replace** commands.

  The options to upgrade the system are as follows:

  - **system upgrade from a non-GISO (image that does not support GISO) to GISO image:** If a system is running a version1 with an image that does not support GISO, the system cannot be upgraded directly to version2 of an image that supports GISO. Instead, the version1 must be upgraded to version2 mini ISO, and then to version2 GISO.
  - **system upgrade in a release from version1 GISO to version2 GISO:** If both the GISO images have the same base version but different labels, **install add** and **install activate** commands does not support same version of two images. Instead, using **install update** command installs only the delta RPMs. System reload is based on restart type of the delta RPMs.
  - **system upgrade across releases from version1 GISO to version2 GISO:** Both the GISO images have different base versions. Use **install add** and **install activate** commands, or **install replace** command to perform the system upgrade. The router reloads after the upgrade with the version2 GISO image.

Step 2 Run the **show install repository all** command in System Admin mode to view the RPMs and base ISO for host, system admin and XR.

Step 3 Run the **show install package <golden-iso>** command to display the list of RPMs, and packages built in GISO.

The ISO, SMUs and packages in GISO are installed on the router.

Install Replace with Golden ISO

Golden ISO (GISO) upgrades the router to a version that has a predefined list of software maintenance update (SMUs) with a single operation. However, to update to the same version with a different set of SMUs requires a two-step process.

To avoid this two-step process, use the **install replace** command to replace the currently active version with the full package including the image an SMUs from the newly added GISO.

The process involves upgrading the GISO to add the delta SMUs, and manually deactivating the SMUs that are not in use. In addition, this is the only method to upgrade to GISO containing different optional RPMs, which is a subset of the running set of optional RPMs. For example, consider V1 of GISO is the running version with V1 mini and optional RPMs V1 mpls, V1 mpls-te, V1 mgbl, and V1 k9sec. If V2 of GISO does not contain V2 k9sec, then use **install replace** to upgrade to the optional RPMs in V2.

Procedure

Step 1 **install replace** *<GISO-location>* [**commit** | **noprompt**]

Example:

```
Router#install replace harddisk:/<giso-image>.iso
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Install operation 11 started by root:
exec-timeout is suspended.
No install operation in progress at this moment
Label = More_Pkgs
ISO <giso-iso-image>.iso in input package list. Going to upgrade the system to

version <new-giso-image>.
System is in committed state
Current full-label: <giso-image>_R_Commit
Current only-label: R_Commit
Current label: R_Commit
Updating contents of golden ISO
Scheme : localdisk
Hostname : localhost
Username : None
SourceDir : /ws
Collecting software state..
Getting platform
Getting supported architecture
Getting active packages from XR
Getting inactive packages from XR
Getting list of RPMs in local repo
Getting list of provides of all active packages
Getting provides of each rpm in repo
Getting requires of each rpm in repo
Fetching .... <giso-image>.iso
Label within GISO: More_Pkgs
Skipping <platform>-mgbl-3.0.0.0-<release>.x86_64.rpm from GISO as it's active
Adding packages
        <platform>-golden-x-<release>-<Label>.iso
RP/0/RP0/CPU0:Jun 20 14:43:59.349 UTC: sdr_instmgr[1164]: %INSTALL-INSTMGR-2-OPERATION_SUCCESS :

Install operation 12 finished successfully
Install add operation successful
```

```
Activating <platform>-golden-x-<release>-<Label>
Jun 20 14:44:05 Install operation 13 started by root:
  install activate pkg <platform>-golden-x-<release>-<Label> replace noprompt
Jun 20 14:44:05 Package list:
Jun 20 14:44:05     <platform>-golden-x-<release>-<Label>.iso
Jun 20 14:44:29 Install operation will continue in the background
exec-timeout is resumed.
Router# Install operation 13 finished successfully
Router: sdr_instmgr[1164]: %INSTALL-INSTMGR-2-OPERATION_SUCCESS :

Install operation 13 finished successfully
```

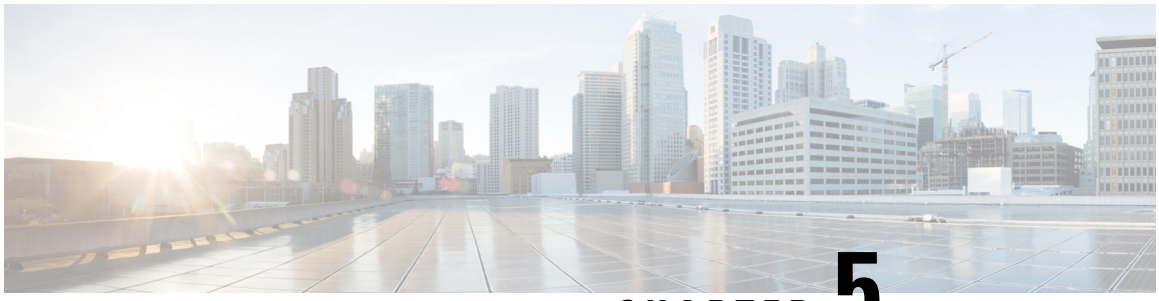
The version and label of the newly added GISO is compared with the version and label of the currently active version. If a mismatch is identified, a new partition is created and the full package is installed. After installation, the system reloads with the image and packages from the newly added GISO.

Step 2 **show version**

Example:

```
Router#show version
Wed Jun 20 15:06:37.915 UTC
Cisco IOS XR Software, Version <new-giso-image>
Copyright (c) 2013-2018 by Cisco Systems, Inc.

Build Information:
Built By     : <user>
Built On     : <date>
Build Host   : <host-name>
Workspace    : <workspace-name>
Version      : <version>
Location    : <path>
Label        : <label-name>

cisco <platform> () processor
System uptime is 3 hours 51 minutes
```

The system loads with the image and packages from the newly added GISO.



CHAPTER 5

Cisco IOS XRv 9000 Router Smart Licensing

Beginning with Cisco IOS XR Release 5.4, Cisco IOS XRv 9000 Router supports activation using Cisco Smart Licensing. Cisco Smart Licensing removes the requirement of having to install node locked licenses into Cisco IOS XRv 9000 instances.

- Cisco Smart Licensing, on page 37
- Cisco IOS XRv 9000 Router Licensing Model, on page 38
- Restrictions and Guidelines for Smart Licensing, on page 40
- Configure Cisco Smart Licence, on page 40
- Registering the Cisco IOS XRv 9000 Router with the Cisco Licensing Cloud, on page 40
- Managing Smart License , on page 41
- Virtual Unique Device Identifier (vUDI) , on page 42
- Troubleshooting Cisco Smart License Issues, on page 43

Cisco Smart Licensing

Cisco Smart Licensing is a cloud-based, software license management solution that enables you to automate time-consuming, manual licensing tasks. The solution lets you to easily track the status of your license and software usage trends.

Cisco Smart Licensing helps simplify three core functions:

- Purchasing—The software that you have installed in your network can automatically self-register, without Product Activation Keys (PAKs).
- Management—You can automatically track activations against your license entitlements. Additionally, there is no need to install the license file on every node. You can create license pools (logical grouping of licenses) to reflect your organization structure. Smart Licensing offers you Cisco Smart Software Manager, a centralized portal that enables you to manage all your Cisco software licenses from one centralized website.
- Reporting—Through the portal, Cisco Smart Licensing offers an integrated view of the licenses you have purchased, and what has been actually deployed in your network. You can use this data to make better purchase decisions, based on your consumption.

Cisco IOS XRv 9000 Router only support activation using Cisco Smart Licensing. Cisco Smart Licensing removes the requirement of having to install node locked licenses into Cisco IOS XRv 9000 Router instances. Instead, the Cisco IOS XRv 9000 Router communicates to the Cisco Licensing Cloud (directly, through a

proxy, or through a Smart Licensing Satellite) to provide a report of which features and to what scale the system is being used.

Cisco Smart Licensing uses Cisco Smart Call Home feature to communicate with the Cisco Smart Software Manager. Smart Call Home is auto-configured for default Smart Licensing Setup. For more information about Cisco Smart Call Home and non-default configuration, refer *Cisco ASR 9000 Series Aggregation Services Router System Management Configuration Guide*, chapter *Configuring Call Home on the Cisco ASR 9000 Series Router*.

Cisco Smart Licensing uses the Cisco Smart Software Manager for managing licenses. To access the Cisco Smart Software Manager, click here.

For information on manually renewing smart license, and deregistering a device from Cisco Smart Licensing, see the section Managing Smart License , on page 41 .

For more information about Cisco Smart Software Manager, see the Cisco Smart Software Manager User Guide, which is accessible from the Cisco Smart Software Manager tool.

Cisco IOS XRv 9000 Router Licensing Model

The Cisco IOS XRv 9000 Router licensing includes demo and production modes.

Table 2: Cisco IOS XRv Router Licensing Mode

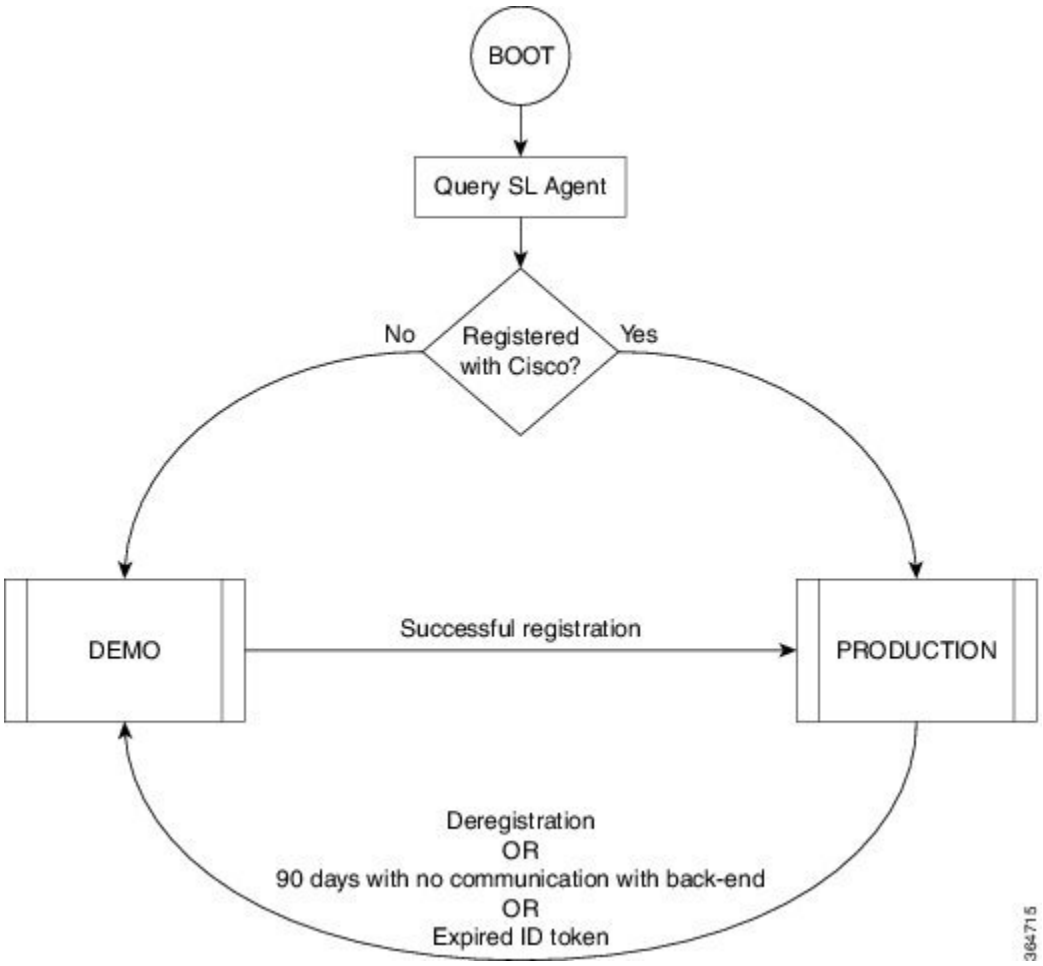
Mode	Description
Demo	• This is the default mode when the router is launched. • No cloud connectivity is required. • No feature level enforcement. • Rate limitations of 200 Kbps for throughput for all interfaces.
Production	• This mode requires registration. • No enforcement applied.

Unlike other Cisco products that have dedicated hardware (ASR9K, NCS 4K, and NCS 6K), because of its virtual nature, Cisco IOS XRv 9000 Router cannot boot into the evaluation mode; this provides users with unlimited access to features without requiring licenses, just by reinstalling the software every few months.

For this reason, when in demo mode, Cisco IOS XRv 9000 Router will impose throughput scale limitations that are sufficiently strong to render the system unusable in a deployment scenario, while remaining fully functional for demonstration purposes.

Note When the router is in an unregistered state, the licenses are in EVAL (evaluation) mode. Evaluation period will last for 90 days.

Figure 3: Licensing Modes



This figure illustrates the basic state transitions that support demo and production modes:

- The system boots into either demo or production mode, depending on whether the Smart Licensing Agent's state indicates that the system is currently registered with the Cisco back-end, or not
- When in demo mode, the system imposes rate limitations of 200 Kbps for throughput.
- When in production mode, no limitations are imposed, and no enforcement is applied, regardless of whether the customer is in compliance or not, from a licensing point of view. (This is in keeping with the smart licensing principles: each router only reports usage to the back-end, and any out-of-compliance accounts trigger back-end processes to reach out to customers to purchase or renew licenses).

For information on the Cisco IOS XRv 9000 Router license PIDs, refer the latest *Cisco IOS XRv 9000 Router Release Notes*, section *License Ordering Information*.

Managing Evaluation Periods

Because of the limits imposed in the demo mode, an alternate approach needs to be taken to support customers who wish to evaluate Cisco IOS XRv 9000 Router in an unfettered fashion. This procedure explains the alternate approach:

1. The smart licensing back-end supports virtual accounts as a means for a customer to subdivide their licenses. Usually this applies mostly to larger customers; however it can be leveraged at any place.
2. The customer creates a “virtual account” within their licensing portal, intended only for evaluation purposes.
3. A Cisco representative populates limited-time entitlements for each feature the customer is interested in, into this virtual account.
4. The customer proceeds with registration of the on-router with the same procedure as for production, except in this case the consumed licenses comes from this virtual account.

This procedure has the added benefit of allowing the customer to practice (or automate or both) the registration procedure during the evaluation phase, since the on-router operations are identical.

Restrictions and Guidelines for Smart Licensing

- Smart Licensing and/or Smart Licensing Using Policy is not supported on the following Cisco XRv9000 appliances:
  - UCS C220 M4S
  - UCS C220 M5SX
  - UCS C220 M7
- Smart Licensing and/or Smart Licensing Using Policy is not required on Cisco IOS XRv 9000 UCS M7 Appliance, as this platform uses a perpetual licensing model.

Configure Cisco Smart Licence

Cisco IOS XRv 9000 Router comes with a default Smart Call Home configuration sufficient to run Smart Licensing. The Smart Call Home default configuration is triggered by Smart Licensing internally, using a built-in CiscoTAC-1 profile. This default configuration requires router connectivity to cisco.com through traffic or management ports.

There is no intervention required by the customer, except to ensure IP connectivity to the Cisco cloud.

If alternate Smart Licensing configurations are desired through a Smart Call Home Gateway, refer *Cisco ASR 9000 Series Aggregation Services Router System Management Configuration Guide*, chapter *Configuring Call Home on the Cisco ASR 9000 Series Router*.

Registering the Cisco IOS XRv 9000 Router with the Cisco Licensing Cloud

Cisco Smart Licensing is always enabled on Cisco IOS XRv 9000 Router. In order to use the router in production mode, you must register the router with Cisco. Using the ID token, the license agent on the router registers the product with Cisco and receives an identity certificate. This certificate is used for all future communications with Cisco. The license agent on the router automatically renews the registration information with Cisco every 30 days.

This registration step is performed once for each product instance.

Note License substitution is not supported on XRv 9000 router.

Procedure

Step 1 Configure connectivity to cisco.com, either using MgmtEth or other Ethernet interfaces.

Step 2 Go to the Cisco Smart Software Manager website, select the appropriate account, request an ID token, and copy it to clipboard.

Step 3 Execute this command on the router **license smart register idtoken** *id-token* in the configuration mode.

The *id-token* is copied in step 2.

Example:

```
Router(config)# license smart register idtoken YjBkOWM5YTItMDFiOS00ZjBmLTllY2YtODEzMzg1YTMyZDVhLTEz
 ODE0MjE0%0ANzc5NDF8U1BDUTAySWFRTmJqa1NnbmlzRUIyaGlYU
053L0pHZTNvUW9VTFpE%0AekxCOD0%3D%0A
```

The system communicates to the Cisco Smart Licensing servers to obtain authorization for Smart Licensing. For information on License Authorization Status, see the Troubleshooting Cisco Smart License Issues, on page 43

Managing Smart License

The license agent automatically renews the registration information with Cisco every 30 days. However, you can manually renew the Smart License registration. You also have an option to deregister the router from Cisco Smart Licensing. For more information on these options, refer these sections.

Manually Renewing the Smart License Registration

If you need to manually renew the registration information, execute the **license smart renew** command in EXEC mode.

Note Before deleting a VM registered with Cisco, you must deregister it so that it is not counted as one of your entitlements.

Deregistering a Device from Cisco Smart Licensing

To remove the Cisco Smart Licensing registration for the router instance, execute the **license smart deregister** command in EXEC mode. All Cisco Smart Licensing certificates and entitlements are removed.

Virtual Unique Device Identifier (vUDI)

Every Cisco platform has a Unique Device Identifier (UDI), which is comprised of the product ID, version, and a serial number. On physical platforms these are burned into the chassis during device manufacture. For virtual platforms, the system generates a serial number, combines it with product ID and version to create virtual UDI.

To view the UDI information of the router, execute **show license udi** command from EXEC mode.

```
RP/0/RP0/CPU0:ios#show license udi
Tue Aug 25 09:47:09.780 UTC

Product Information
===================
UDI:
PID:R-IOSXRV9000-IMG,SN:DF855094AA4,SUVI:R-IOSXRV9000-IMGDF855094AA4,UUID:1BB98DDC-3EE1-4A60-95A6-530870AC19D9
```

Because Cisco IOS XRv 9000 Router is virtual and represented as a virtual disk image, it can be copied or cloned. This can lead to multiple instances with the same vUDI. Because vUDIs are used as part of licensing to identify instances, care must be taken to ensure no collisions occur. However, in a virtualized environment, the instance itself can only work with the information that the hypervisor provides; so the **virtual-platform udi reset** command is provided as a fail-safe mechanism in addition to boot up logic.

Note The execution of the **virtual-platform udi reset** command causes the VM to reload.

UDI Behavior at boot up

When the Cisco IOS XRv 9000 VM instance boots for the first time, a unique serial number (UDI) is created by the Cisco IOS XRv 9000 Router. If the serial number is not created then random number generator generates a serial number for the VM instance, and combines with the product ID and version to create a vUDI. This is stored in secure storage, and cannot be manually changed by the user.

When a VM is copied or cloned, the vUDI associated with VM is duplicated. Hence to mitigate duplication of vUDIs, hypervisor provides a Universally Unique Identifier (UUID) to a VM instance on every boot that is stored in the virtual hard disk. When the router boots for the first time from the virtual hard disk image, it stores the UUID assigned by the hypervisor inside that virtual hard disk. On sub-sequent boots, it compares the new UUID assigned by the hypervisor to what was used during the last boot. If the same hypervisor is booting the same virtual hard disk image, the UUID must remain same. However, if the disk image was copied and run on a different hypervisor, then the UUID would be different. So the system generates a new serial number and creates new vUDI for the cloned VM.

Identical UDIs are detected by the user because:

- random number generator created two identical serial numbers
- hypervisor did not provide UUID.

In this case users can execute the **virtual-platform udi reset** command from Admin mode to generate a new vUDI.

When you execute the **virtual-platform udi reset** command, you will be prompted to confirm, and then you will receive a series of system messages confirming the request. This will cause Smart Licensing to re-register

with the Cisco Smart Software Manager and a new vUID will be assigned to the Cisco IOS XRv 9000 VM instance.

Troubleshooting Cisco Smart License Issues

To troubleshoot Cisco Smart license issues, these commands are available on all Smart Licensing platforms. These commands can be used to check which entitlements are being consumed, device compliance, etc:

- show license all
- show license status
- show license summary
- show license tech support
- show license udi
- show license usage

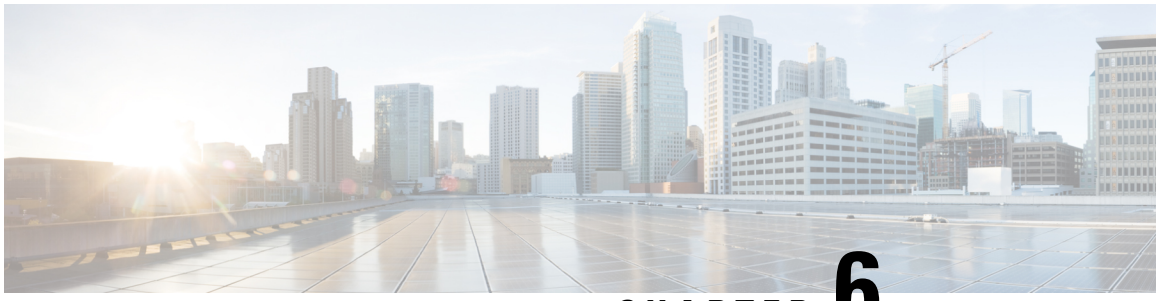
These are the Cisco IOS XRv 9000 Router platform specific commands used by Cisco support:

- show license platform detail
- show license platform summary
- show license platform trace

License Authorization Status

The license authorization status has 4 primary available states:

Status	Description
Registered	Device registration is complete and an ID certificate is received. The ID certificate is used for future communication with the Cisco licensing authority.
Authorized	Registration is complete with a valid Smart Account. The license consumption has commenced. This is the indication of being in compliance.
Out of Compliance	Consumption exceeds available licenses in the Smart Account.
Authorization Expired	The device is unable to communicate with the Cisco Smart Software Manager (CSSM) for an extended period of time. Typically after 90 days this state will be present. The device will attempt to contact the CSSM every hour in order to renew the authorization until the registration period expires.



CHAPTER 6

Accessing the Console Port

This chapter discusses how to access the console on Cisco IOS XRv 9000 router.

- Console Mapping, on page 45
- Access the Cisco IOS XRv 9000 Router Through the VM Console , on page 46
- Access the Cisco IOS XRv 9000 Router Through the Virtual Serial Port , on page 46
- Access the XR Console from the System Admin Console, on page 49
- Create User Profiles and Assign Privileges, on page 50

Console Mapping

Cisco IOS XRv 9000 Router boots when the VM is powered on. Depending on the installation image used (with or without VGA), you can monitor the installation process on the VM console or on the console on the virtual serial port.

This table describes the console mapping if the installation image type *with* VGA is used:

Table 3: Console Mapping for Image Type with VGA

VM Device	Maps to
VM Console	XR Console
First Serial Port	XR Auxiliary
Second Serial Port	Admin Console
Third Serial Port	Admin Auxiliary
Fourth Serial Port	Unused

This table describes the console mapping if the installation image type *without* VGA is used:

Table 4: Console Mapping for Image Type without VGA

VM Device	Maps to
VM Console	Unused
First Serial Port	XR Console

VM Device	Maps to
Second Serial Port	XR Auxiliary
Third Serial Port	Admin Console
Fourth Serial Port	Admin Auxiliary

Access the Cisco IOS XRv 9000 Router Through the VM Console

By default, the XR console is mapped to first serial port. However, under the following circumstances, XR console is mapped to VM console:

- If you are running the VM under the ESXi hypervisor or OpenStack platform and using a VGA image type
- If you are using VGA image for Cisco IOS XRv 9000 Router deployment

The VGA console is accessed at the Console Tab in the vSphere Client. On ESXi, VGA console is console which opens by itself once the installation is complete. You don't need any specific procedure to connect to VM console

After loading the vga image, you can login to VM console. VM console offers the benefit of direct login. However, while using VM console, you might find that copying is a little difficult on a VM console. Also navigation is restricted to a level. Serial console has better copying options and navigation control.

Cisco IOS XRv 9000 Router boots when the VM is powered on. Depending on the installation image used (with or without VGA), you can monitor the installation process on the VM console or on the console on the virtual serial port.

The installation procedure remains same with or without VM console.

See VMware or OpenStack documentation on how to access the VM console through the appropriate GUI interface.

Access the Cisco IOS XRv 9000 Router Through the Virtual Serial Port

By default, Cisco IOS XRv 9000 Router is accessed using the VM console. You can configure the VM to use the virtual serial port as the console port for Cisco IOS XRv 9000 Router. See the following sections to configure the virtual serial port on your hypervisor.

Following are the default console settings:

- Baud rate 115200 bps
- no Parity
- 2 Stop bits and
- 8 Data bits

Note These steps should have been performed during installation of Cisco IOS XRv 9000 Router.

Configuring Serial Console Access in VMware ESXi

This procedure explains how to configure the serial console access in VMware ESXi using VMware vSphere. For more information, refer to the VMware vSphere documentation.

Procedure

Step 1 Power-down the VM.

Step 2 Select the VM and configure the virtual serial port settings:

a) Choose **Edit Settings** > **Add**
b) Choose **Device Type** > **Serial port** and click **Next.**
c) Choose **Select Port Type**, select **Connect via Network** and then click **Next**.

Step 3 Choose the **Select Network Backing** and then do the following:

- Select **Server (VM listens for connection)**
- Enter the **Port URI** using the following syntax: telnet://esxi-host-ipaddress:*portnumber*

  where *portnumber* is the port number for the virtual serial port.
- Under **I/O mode**, select **Yield CPU on poll.**
- Click **Next**.

Step 4 Power-on the VM.

When the VM is powered on, access the virtual serial port console. If you are unable to access the virtual serial port, then perform step 5.

Step 5 Configure the security settings for the virtual serial port.

- Select the **ESXi host** for the virtual serial port.
- Click the **Configuration** tab and then click **Security Profile**.
- In the Firewall section, click **Properties**, and then select the **VM serial port connected over Network** value.

What to do next

You can now access the Cisco IOS XR console using the Telnet port URI: telnet esxi-host-ip-address <*portnumber*>

When you configure the virtual serial port, Cisco IOS XRv 9000 Router is no longer accessible from the VMware ESXi console.

Configuring the Serial Console Access in KVM using Virsh

The serial console access configuration for KVM environment is explained in the sample Virsh XML file available along with the installation images downloaded from Cisco.com. However, you must edit the serial port numbers in the XML source configuration.

This procedure explains how to edit the serial port number for four serial ports and telnet to the serial port to obtain access to the serial console.

Procedure

Step 1 Assign service values to each serial ports. You must pick unused ports while assigning the service values to each port.

The first serial port (port 0) is assigned 11768 as the port number.

```
<!-- Use virsh qemu-monitor-command IOS-XRv-9000_vpe_rwelnode10_virsh - -hmp "info chardev" to view
 or create serial ports -->
    <serial type='tcp'>
       <source mode="bind" host="127.0.0.10" service="11768"/>
       <protocol type="telnet"/>
       <target port="0"/>
    </serial>
```

The second serial port (port 1) is assigned 12251 as the port number.

```
    <serial type='tcp'>
       <source mode="bind" host="127.0.0.10" service="12251"/>
       <protocol type="telnet"/>
       <target port="1"/>
    </serial>
```

The third serial port (port 2) is assigned 17161 as the port number.

```
    <serial type='tcp'>
       <source mode="bind" host="127.0.0.10" service="17161"/>
       <protocol type="telnet"/>
       <target port="2"/>
    </serial>
```

The fourth serial port (port 3) is assigned 16998 as the port number.

```
    <serial type='tcp'>
       <source mode="bind" host="127.0.0.10" service="16998"/>
       <protocol type="telnet"/>
       <target port="3"/>
    </serial>
```

Step 2 To obtain the serial console access, telnet to the port, use the **telnet localhost <*portnumber*>** command.

This example shows how to access second serial port that has port number 12251:

```
telnet localhost 12251
```

Step 3 If the Virsh console is desired, uncomment the console section and comment out the last serial port.

Here is an example:

```
<!-- <console type='pty'> -->
<!--  <target type='serial' port='0'/> -->
<!-- </console> -->
```

Configuring the Serial Console Access in KVM using QEMU

Access to Cisco IOS XRv 9000 Router using KVM-QEMU command line can be obtained with telnet.

Procedure

The sample command line creates 4 serial ports that can be connected through telnet

```
-serial telnet:127.0.1.10:10621,nowait,server \
-serial telnet:127.0.1.10:14713,nowait,server \
-serial telnet:127.0.1.10:18090,nowait,server \
-serial telnet:127.0.1.10:17181,nowait,server \
```

In order to access one of the ports, use the **telnet localhost** <*port-number*> command.

For example, in order to access the first port shown in the sample CLI above:

```
telnet localhost 10621
```

Access the XR Console from the System Admin Console

In cases where there is an inband connectivity problem and you are able to log in only to the System Admin console, you can use the following method to access the XR console.

Procedure

Step 1 Login to the System Admin console as the root user.

Step 2 **show vm location 0/RP0**

Example:

The following example shows the command output with all the Virtual Machines (VM) displayed.

```
sysadmin-vm:0_RP0# show vm location 0/RP0
Location: 0/RP0
Id                Status        IP Address       HB Sent/Recv
--------------------------------------------------------------
sysadmin          running       192.0.0.1        NA/NA
default-sdr       running       192.0.0.4        6304304/6304304
default-sdr       running       192.0.0.6        315193/315193
```

Step 3 **run ssh** *ip address*.

Example:

The following example shows the connection to the first **default-sdr**:

```
sysadmin-vm:0_RP0# run ssh 192.0.0.4
Last login: Fri Apr  6 20:53:47 2018 from 192.0.0.1
[xr-vm_node0_RP0_CPU0:~]$
```

Step 4 **exec**

Example:

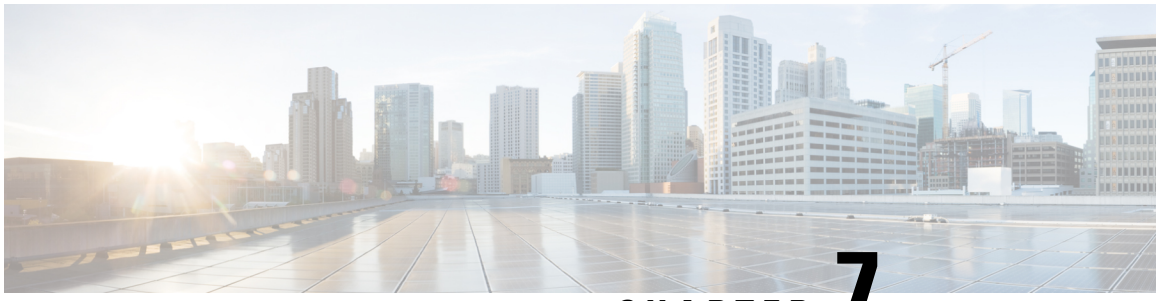
The following example shows accessing the XR console with XR login credentials:

```
[xr-vm_node0_RP0_CPU0:~]$exec
User Access Verification

Username: iox

Password:

RP/0/RP0/CPU0:router#
```

Create User Profiles and Assign Privileges

To provide controlled access to the XR and System Admin configurations on the router, user profiles are created with assigned privileges. The privileges are specified using command rules and data rules. The authentication, authorization, and accounting (aaa) commands are used for the creation of users, groups, command rules, and data rules. The `aaa` commands are also used for changing the disaster-recovery password.

For more information on creating user profiles and assign privileges, see the System Setup and Software Installation Guide for Cisco ASR 9000 Series Routers, chapter *Create User Profiles and Assign Privileges*.



CHAPTER 7

Cisco IOS XRv 9000 Router Control Plane Specific Features

This chapter covers information about the Cisco XRv 9000 Control Plane Specific features.

- BGP Optimal Route Reflector, on page 51
- Support for bgp bestpath igp-metric ignore Command, on page 66
- BFD for Multihop Paths, on page 66
- CVAC - Bootstrap Configuration Support, on page 68
- ORR Support for FlexAlgo, on page 71
- Enabling Segment Routing Flexible Algorithm, on page 71
- IPv4 and IPv6 Traffic Redirect using Policy based Routing, on page 81
- gNMI Bundling of Telemetry Updates, on page 84
- QoS on IPv4 GRE Tunnels, on page 85
- Accessing the Networking Stack, on page 87
- Application Hosting on the Cisco IOS XR Linux Shell, on page 91
- Class-based MPLS traffic steering, on page 101
- IS-IS auto-cost reference bandwidth, on page 105

BGP Optimal Route Reflector

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
BGP ORR 6PE	Release 7.3.1	This feature is introduced. If there is no path selectable as bestpath for a given ORR table, you can assign the default table's bestpath as ORR group's bestpath. This feature enables IPv6 label-unicast with IPv4 nexthop and fallback default path. New keyword introduced in this release: • **fallback-default-bestpath**

BGP-ORR (optimal route reflector) enables virtual route reflector (vRR) to calculate the best path from a route reflector (RR) client's point of view.

BGP ORR calculates the best path by:

1. Running SPF multiple times in the context of its RR clients or RR clusters (set of RR clients)
2. Saving the result of different SPF runs in separate databases
3. Using these databases to manipulate BGP best path decision and thereby allowing BGP to use and announce best path that is optimal from the client’s point of view

In an autonomus system, a BGP route reflector acts as a focal point and advertises routes to its peers (RR clients) along with the RR's computed best path. Since the best path advertised by the RR is computed from the RR's point of view, the RR's placement becomes an important deployment consideration.

With network function virtualization (NFV) becoming a dominant technology, service providers (SPs) are hosting virtual RR functionality in a cloud using servers. A vRR can run on a control plane device and can be placed anywhere in the topology or in a SP data center. Cisco IOS XRv 9000 Router can be implemented as vRR over a NFV platform in a SP data center. vRR allows SPs to scale memory and CPU usage of RR deployments significantly. Moving a RR out of its optimal placement requires vRRs to implement ORR functionality that calculates the best path from a RR client's point of view.

BGP ORR offers these benefits:

- calculates the bestpath from the point of view of a RR client.
- enables vRR to be placed anywhere in the topology or in a SP data center.
- allows SPs to scale memory and CPU usage of RR deployments.

Note Enabling the ORR feature increases the memory footprint of BGP and RIB. With increased number of vRR configured in the network, ORR adversely impacts convergence for BGP.

BGP ORR Configuration Methods

You can configure the BGP ORR as follows:

- BGP ORR without path advertisement policy.
- BGP ORR with path advertisement policy.

BGP ORR without Path Advertisement Policy

The vRR performs the following steps to calculate and advertise the BGP ORR best path to all the vRR sites and ORR groups:

1. The vRR calculates the BGP ORR best path for each ORR group from the global Address Family Indicator (AFI) policy.
2. The vRR advertises the calculated best path along with selected additional paths.

If you haven't assigned a policy to the ORR group, by default, the vRR calculates the best paths using the Global-AFI's default selection policy for that ORR group. For more information, refer to BGP ORR without Path Advertisement Policy Topology, on page 53.

BGP ORR with Path Advertisement Policy

The vRR performs the following steps to calculate and advertise the BGP ORR best and backup/additional paths to all the vRR sites and ORR groups:

1. You can configure an ORR group add path policy using the **optimal-route-reflection** command. For more information, refer to BGP Optimal Route Reflector with Path Advertisement Policy, on page 61.
2. The vRR calculates the best path for each ORR group based on the assigned policy.
3. The vRR advertises the best and backup/additional paths to PE routers within the ORR group that has addpath route-policy.

Note If required you can enable the vRR to advertise all the paths to the ORR groups using the *addpath-all policy* in the **optimal-route-reflection** command.

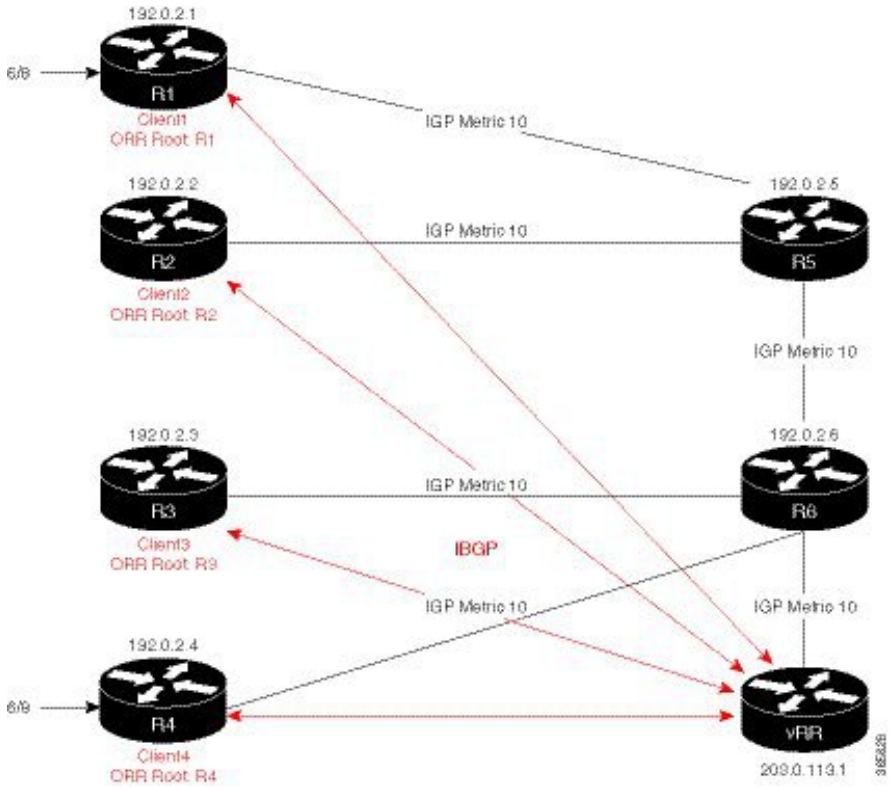
4. The vRR advertises the calculated best path along with all the known paths to all vRR sites within the Intra-Region Groups.
5. The vRR advertises only the client path and the necessary peering path to all the vRR sites within the Inter-Region Group.

BGP ORR without Path Advertisement Policy Topology

Consider a BGP ORR without path advertisement policy topology where:

- Router R1, R2, R3, R4, R5 and R6 are route reflector clients
- Router R1 and R4 advertise 6/8 prefix to vRR

Figure 4: BGP-ORR without Path Advertisement Policy Topology



vRR receives prefix 6/8 from R1 and R4. Without BGP ORR configured in the network, the vRR selects R4 as the closest exit point for RR clients R2, R3, R5, and R6, and reflects the 6/8 prefix learned from R4 to these RR clients R2, R3, R5, and R6. From the topology, it is evident that for R2 the best path is R1 and not R4. This is because the vRR calculates best path from the RR's point of view.

When the BGP ORR is configured in the network, the vRR calculates the shortest exit point in the network from R2's point of view and determines that R1 is the closest exit point to R2. vRR then reflects the 6/8 prefix learned from R1 to R2.

Restrictions and Limitations for BGP ORR without Path Advertisement Policy

The following are the restrictions and limitations for BGP ORR without path advertisement policy:

- If there are multiple paths added to a ORR group, the vRR chooses only one path among them to calculate the best path for a vRR site.
- The vRR calculates and advertises same type and set of additional paths to all the clients.

Configuring BGP ORR without Path Advertisement Policy

Procedure

Perform the following steps to configure the BGP ORR without path advertisement policy.

Step 1 Configure an ORR globally under router BGP mode.

Example:

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# optimal-route-reflection ipv4 foo 10.1.1.1 10.1.1.2 10.1.1.3
Router(config-bgp)# optimal-route-reflection ipv6 bar abcd::1 abcd::2 abcd::3
```

Step 2 Enable the ORR group under address-family mode.

Example:

```
Router(config-bgp-route)# address-family ipv4 unicast
Router(config-bgp-route-af)# optimal-route-reflection apply foo
Router(config-bgp-route-af)# exit
Router(config-bgp-route)# address-family ipv6 unicast
Router(config-bgp-route-af)# optimal-route-reflection apply bar
Router(config-bgp-route-af)# allocate-label {all | route-policy <>}
Router(config-bgp-route-af)# commit
```

Step 3 Specify a neighbor as an ORR client.

Example:

```
Router# configure
Router(config)# neighbor 2.2.2.2
Router(config-nbr)# remote-as 100
Router(config-nbr)# address-family ipv4 unicast
Router(config-nbr-af)# optimal-route-reflection foo
Router(config-nbr-af)# exit
Router(config-nbr)# address-family ipv6 label-unicast
Router(config-nbr-af)# optimal-route-reflection foo
Router(config-nbr-af)# commit
```

Step 4 Enable the selection of default table's bestpath in the absence of bestpath for an ORR group.

Example:

```
Router# configure
Router(config)# router bgp 65000
Router(config-bgp)# bgp router-id 10.1.1.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# optimal-route-reflection fallback-default-bestpath
```

Step 5 View the running configuration to verify the configuration that you have configured.

Example:

```
router bgp 100
 optimal-route-reflection ipv4 foo 10.1.1.1 10.1.1.2 10.1.1.3
 optimal-route-reflection ipv6 bar abcd::1 abcd::2 abcd::3
 address-family ipv4 unicast
  optimal-route-reflection apply foo
 address-family ipv6 unicast
```

```
 optimal-route-reflection apply foo
 allocate-label {all | route-policy <>}
neighbor 2.2.2.2
 remote-as 100
 address-family ipv4 unicast
  optimal-route-reflection foo
 address-family ipv6 label-unicast
  optimal-route-reflection foo
```

Step 6 Perform the following steps to verify the BGP ORR configuration for IPv4:

a) Verify whether R2 received the best exit, execute the **show bgp <prefix>** command (from R2) in EXEC mode. In the above example, R1 and R4 advertise the 6/8 prefix; run the **show bgp 6.0.0.0/8** command:

Example:

```
Router# sh bgp 200.1.1.0/24 path-elements
BGP routing table entry for 200.1.1.0/24
Versions:
    Process     bRIB/RIB SendTblVer
    Speaker           84         84
        Flags: 0x043e3028+0x00000000;
Last Modified: Sep 7 20:21:38.000 for 00:11:45
Paths: (2 available, best #1)
Path-elements: 2
    Path ID: 1
        Gateway metric 0, Version 84
    Path: Nexthop 11.1.1.2, flags 0x4000000001440207
          Neighbor 11.1.1.2, Received Path ID 0
    Flags: 0x00000001
           status: valid
           path type: bestpath
           add-path action:
    Opaque: pelem=0xebf81d10
            net=0xabe435c, tblattr=0xa962f64 (ver 84)
            path=0xabf4450, path-tblattr=0xa962f64 (ver 84)
                       nobestpath-tblattr=0xa963234 (ver 0)
                        noaddpath-tblattr=0xa9631e4 (ver 0)
            bitfields=0xace9f64 (val=0x1a, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0xac145d8 (val=0x0, size=0)
            orr-ap-bitfields=0xac145d8 (val=0x2, size=0). <<<<<<<<<<<<<<<
            net-next=0xebf81cd4, tblattr-prev=0xebf81cd4, tblattr-next=0xebf81c98
Path ID: 2
    Gateway metric 0, Version 107
    Path: Nexthop 192.168.0.10, flags 0x4000000000040007
          Neighbor 192.168.0.2, Received Path ID 2
    Flags: 0x00000042
           status: valid
           path type: backup
           add-path action: advertise
    Opaque: pelem=0x10a0bba8
            net=0x109cbb14, tblattr=0x10399c84 (ver 156)
            path=0x109ebcd4, path-tblattr=0x10399c84 (ver 156)
                        nobestpath-tblattr=0x10399fa4 (ver 0)
                            noaddpath-tblattr=0x10399f54 (ver 0)
            bitfields=0x10c00624 (val=0x10, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0x10a2b8cc (val=0x0, size=1)
            orr-ap-bitfields=0x10a2b8a8 (val=0x6, size=1) <<<<<<<<<<<<<<<
            net-next=0x10a0b528, tblattr-prev=0x10a0bbe8, tblattr-next=0x0
```

The above show output states that the best path for R2 is through R1, whose IP address is 192.0.2.1 and the metric of the path is 20.

b) Execute the **show bgp** command from the vRR to determine the best path calculated for R2 by ORR. R2 has its own update-group because it has a different best path (or different policy configured) than those of other peers:

Example:

```
VRR# show bgp 6.0.0.0/8
Thu Apr 28 13:36:42.744 UTC
BGP routing table entry for 6.0.0.0/8
Versions:
Process bRIB/RIB SendTblVer
Speaker 13 13
Last Modified: Apr 28 13:36:26.909 for 00:00:15
Paths: (2 available, best #2)
Advertised to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
ORR bestpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.1 (metric 30) from 192.0.2.1 (192.0.2.1)
Origin incomplete, metric 0, localpref 100, valid, internal, add-path
Received Path ID 0, Local Path ID 2, version 13
Path #2: Received by speaker 0
Advertised to update-groups (with more than one peer):
0.2
ORR addpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.4 (metric 20) from 192.0.2.4 (192.0.2.4)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
Received Path ID 0, Local Path ID 1, version 13
```

Note
Path #1 is advertised to update-group 0.1. R2 is in update-group 0.1.

c) Execute the **show bgp** command for update-group 0.1 verify whether R2 is in update-group 0.1.

Example:

```
VRR# show bgp update-group 0.1
Thu Apr 28 13:38:18.517 UTC

Update group for IPv4 Unicast, index 0.1:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 65000
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
ORR root (configured): g1; Index: 0
4-byte AS capable
Non-labeled address-family capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 5, replicated: 5
All neighbors are assigned to sub-group(s)
```

```
Neighbors in sub-group: 0.2, Filter-Groups num:1
Neighbors in filter-group: 0.2(RT num: 0)
192.0.2.2
```

d) For further verification, check the contents of the table created on vRR as a result of configuring the g1 policy. From R2's point of view, the cost of reaching R1 is 20 and the cost of reaching R4 is 30. Therefore, the closest and best exit for R2 is through R1.

Example:

```
VRR# show orrspf database g1
Thu Apr 28 13:39:20.333 UTC

ORR policy: g1, IPv4, RIB tableid: 0xe0000011
Configured root: primary: 192.0.2.2, secondary: NULL, tertiary: NULL
Actual Root: 192.0.2.2, Root node: 2000.0100.1002.0000

Prefix Cost
203.0.113.1 30
192.0.2.1 20
192.0.2.2 0
192.0.2.3 30
192.0.2.4 30
192.0.2.5 10
192.0.2.6 20

Number of mapping entries: 8
```

Step 7 Perform the following steps to verify the BGP ORR configuration for IPv6 Provider Edge (6PE):

a) Verify the BGP ORR configuration using the **show bgp** command.

Example:

```
show bgp ipv6 labeled-unicast 1111::1/128
Tue Mar 2 10:25:00.748 PST
BGP routing table entry for 1111::1/128
Versions:
Process bRIB/RIB SendTblVer
Speaker 4 4
Last Modified: Mar 2 10:18:53.000 for 00:06:08
Paths: (3 available, best #3)
Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
ORR bestpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.168.0.3 (metric 75) from 192.168.0.3 (192.168.0.3)
Received Label 24007
Origin incomplete, metric 0, localpref 100, valid, internal, add-path, labeled-unicast
Received Path ID 0, Local Path ID 2, version 4
Path #2: Received by speaker 0
Not advertised to any peer
Local, (Received from a RR-client)
192.168.0.4 (metric 190) from 192.168.0.4 (192.168.0.4)
Received Label 24007
Origin incomplete, metric 0, localpref 100, valid, internal, labeled-unicast
Received Path ID 0, Local Path ID 0, version 0
Path #3: Received by speaker 0
Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
0.2
Local, (Received from a RR-client)
```

```
192.168.0.5 (metric 65) from 192.168.0.5 (192.168.0.5)
Received Label 24007
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, labeled-unicast
Received Path ID 0, Local Path ID 1, version 3
```

b) Verify the ipv6 group configuration using the **show bgp** command.

Example:

```
show bgp ipv6 labeled-unicast update-group
Tue Mar 2 10:25:51.308 PST

Update group for IPv6 Labeled-unicast, index 0.1:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 1
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
ORR root (configured): orr-grp-1; Index: 0
4-byte AS capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 1, replicated: 2
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1
Neighbors in filter-group: 0.2(RT num: 0)
192.168.0.2 192.168.0.4

Update group for IPv6 Labeled-unicast, index 0.2:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 1
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
4-byte AS capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 1, replicated: 4
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.1, Filter-Groups num:1
Neighbors in filter-group: 0.1(RT num: 0)
192.168.0.3 192.168.0.5
```

c) Verify the IPv6 unicast routes in the BGP configuration using the **show bgp** command.

Example:

```
show bgp ipv6 unicast orr-group all
Tue Mar 2 10:26:41.072 PST
```

```
Name Tableid Nbrcnt Index Root
orr-grp-1 0xe0000019 2 0 192.168.0.3
```

d) Verify the BGP speaker global ORR policy group.

Example:

The following show command displays the BGP speaker global ORR policy group table.

```
Router# show bgp orr-group global all
Wed Apr  8 16:46:29.929 PDT
Name            Policy-afi Global     Tableid          AFI-count  Root
orr-grp-3       IPv4       Yes        0xe0000014        1         1.1.2.1
orr-grp-2       Ipv6       Yes        0xe0800013        0         1::1
orr-grp-1       IPv4       Yes        0xe0000012        2         192.168.0.3
```

The following show command displays the details of a global ORR group entry for the given ORR name.

```
Router# show bgp orr-group global orr-grp-1
Wed Apr  8 16:46:51.596 PDT
  ORR Name : orr-grp-1
       policy afi : IPv4
global Defined : Yes
          tableid : 0xe0000012
           aficnt : 2
IPv4 unicast used : Yes
IPv6 unicast used : Yes
           root : 192.168.0.3
```

The following show command displays the BPM ORR policy group table:

```
Router# show bgp orr-group bpm all
Wed Apr  8 16:49:44.223 PDT
Name            Policy-afi Global     AFI-cnt    Nbr-af-cnt      Root
orr-grp-3       IPv4       Yes        1          0               1.1.2.1
orr-grp-2       IPv6       Yes        0          0               1::1
orr-grp-1       IPv4       Yes        2          4               192.168.0.3
```

The following show command displays the detail of a BPM ORR group entry for the given ORR name.

```
Router# show bgp orr-group bpm orr-grp-1
Wed Apr  8 16:50:02.437 PDT
    ORR Name : orr-grp-1
        v4 policy : Yes
   global Defined : Yes
        AFI count : 2
 total nbr af cnt : 4
IPv4 unicast used : Yes
IPv6 unicast used : Yes
  IPv4 nbr af cnt : 2
  IPv6 nbr af cnt : 2
           root : 192.168.0.3
```

BGP Optimal Route Reflector with Path Advertisement Policy

Table 6: Feature History Table

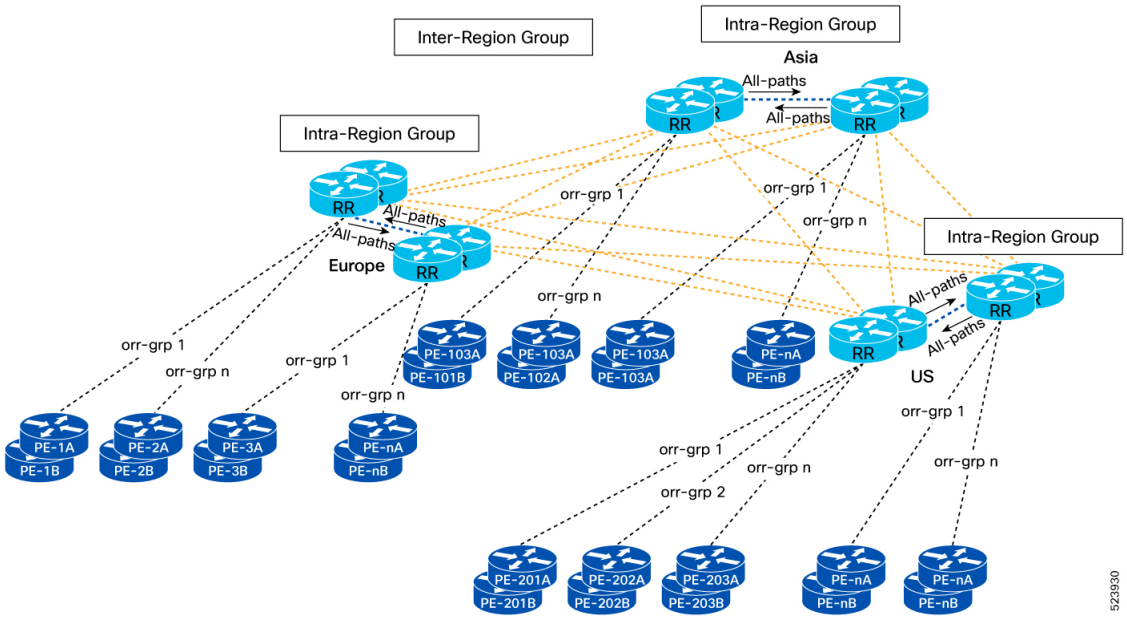
Feature Name	Release Information	Feature Description
BGP Optimal Route Reflector with Path Advertisement Policy	Release 24.2.1	This feature allows you to assign an independent routing policy to an individual Optimal Route Reflector (ORR) group. The virtual route reflector (vRR) calculates the additional paths for that ORR group based on the assigned policy. This results in having a different best and backup path for each ORR group which is the optimal path from the client point of view. This feature introduces these changes: **CLI:** • The *route-policy* keyword is introduced in the **optimal-route-reflection** command. **YANG Data Model:** • `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg.yang` (see GitHub, YANG Data Models Navigator)

BGP-ORR (optimal route reflector) enables the virtual route reflector (vRR) to calculate the best path from a route reflector (RR) client's point of view. For more information, refer to BGP Optimal Route Reflector, on page 51.

Traffic Flow Topology

Consider a topology where the network traffic flows from ORR groups, through vRR sites, through an Intra-Region Group, and finally through an Inter-Region Group. Refer to the figure Figure 5: Advertising BGP-ORR Best Path, on page 62.

Figure 5: Advertising BGP-ORR Best Path



The following are the definition of the hierarchical structure of network elements within a BGP ORR environment, as shown in the figure Figure 5: Advertising BGP-ORR Best Path, on page 62:

- ORR Groups: Multiple sets of routers and provider edge (PE) routers connect to form an ORR group. For example, orr-grp n.
- vRR Sites: Multiple ORR groups connect to form a vRR site.
- Intra-Region Group: Multiple vRR sites connect to form an Intra-Region Group. For example, Europe.
- Inter-Region Group: Multiple Intra-Region groups connect to form an Inter-Region Group.

This feature allows you to configure add-path selection policies for each vRR site and ORR group. The vRR calculates and advertises the BGP ORR best and backup/additional paths to all the vRR sites and ORR groups. For more information, refer to the BGP ORR with Path Advertisement Policy, on page 53.

You can configure the following router policy to enable the vRR to advertise the best, backup, or additional paths:

- **"Is-destination"**: This is a term assigned to the policy that advertises a specific type of path, such as best, backup, or add-path, to a designated neighbor. This policy doesn't apply to ORR neighbors.
- **"Set path-selection backup 1 advertise"**: This is a term assigned to the policy that enables the vRR to advertise only the best and backup path to the vRR sites and ORR groups.
- **"Set path-selection all advertise"**: This is a term assigned to the policy that enables the vRR to advertise the best and additional paths to the vRR sites and ORR groups.

Restrictions and Limitations for BGP ORR with Path Advertisement Policy

The following are the restrictions and limitations for BGP ORR with path advertisement policy:

- By default, the vRR advertises the calculated best and backup paths to the ORR group. If required you can enable vRR to advertise all the paths to the ORR group using the *addpath-all* keyword in the **additional-paths selection** command.
- By default, the vRR advertises all the paths to all vRR sites.
- If you have not assigned a policy to the ORR group and if the AFI router policy is not assigned with a default path, then the vRR does not calculate the best paths for that ORR group.
- The router policy **"Is-destination"** option is not supported with ORR route-policy attach-point. For example, if you have configured the "is-destination" path, the "is-destination" path is advertised only to the vRR but not for PEs.
- The **"Set path-selection backup 1 install"** option doesn't support ORR groups as vRR is not part of the forwarding path; therefore, you cannot install ORR and backup paths on the forwarding table. If you configure this policy for an ORR group, the system ignores the install command and, by default, advertises the backup paths to the specified ORR groups.

Configuring BGP ORR with Path Advertisement Policy

Perform the following steps to configure the BGP ORR with path advertisement policy.

Procedure

Step 1 Configure RPL policy "Is-destination".

Example:

```
Router# configure
Router(config)# route-policy Bestpath
Router(config-rpl)# if destination is-best-path then
Router(config-rpl)# pass
Router(config-rpl)# else
Router(config-rpl)# drop
Router(config-rpl)# endif
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

This policy allows vRR to pass the path provided by you only if it is the best path; otherwise, it will drop the route. This policy allows vRR to advertise only the best path to the vRR sites and it doesn't apply to ORR neighbors.

Step 2 Configure a RPL policy to advertise the best and back up path.

Example:

```
Router# configure
Router(config)# route-policy Backup
Router(config-rpl)# set path-selection backup 1 advertise
Router(config-rpl)# end-policy
Router(config-rpl)# exit
```

This policy allows vRR to advertise the best and backup path.

Step 3 Configure a RPL policy to advertise all paths.

Example:

```
Router# configure
Router(config)# route-policy addpath-all
```

```
Router(config-rpl)# set path-selection all advertise
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# route-policy pass
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# route-policy addpath-backup
Router(config-rpl)# set path-selection backup 1 advertise
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# end
Router#
```

This policy allows vRR to advertise all the paths to the ORR groups.

Step 4 Configure an ORR group add path policy using the **optimal-route-reflection** command, and assign the address-family to the ORR group using the **address-family** command.

Example:

```
Router# configure
Router(config)# router bgp 13
Router(config-bgp)# optimal-route-reflection ipv4 orr-grp-1 1.1.1.1 route-policy addpath-backup
Router(config-bgp-router)# address-family ipv4 unicast
Router(config-bgp-router-af)# additional-paths selection route-policy addpath-all
Router(config-bgp-router-af)# optimal-route-reflection apply orr-grp-1
Router(config-bgp-router-af)# commit
```

Step 5 Configure the neighbor groups for BGP using the **neighbor-group** command.

Example:

```
Router# configure
Router(config)# neighbor-group PR
Router(config-nbrgrp)# address-family ipv4 unicast
Router(config-nbrgrp-af)# route-reflector-client
Router(config-nbrgrp-af)# optimal-route-reflection orr-grp-1
Router(config-nbrgrp-af)# route-policy pass in
Router(config-nbrgrp-af)# commit
```

Step 6 Apply the configured BGP neighbor group to the BGP neighbors using the **neighbor** command.

Example:

```
Router# configure
Router(config)# neighbor 10.10.10.10
Router(config-nbr)# use neighbor-group PR
```

Step 7 View the running configuration to verify the configuration that you have configured.

Example:

```
/*Enable to BGP Policies*/
route-policy addpath-all
  set path-selection all advertise
end-policy
route-policy pass
  pass
end-policy
route-policy addpath-backup
  set path-selection backup 1 advertise
end-policy

/*Configure ORR Add-Path and create ORR Groups*/
```

```
router bgp 13
  optimal-route-reflection ipv4 orr-grp-1 1.1.1.1 route-policy addpath-backup
  optimal-route-reflection ipv4 orr-grp-2 2.2.2.2
  address-family ipv4 unicast
    additional-paths selection route-policy addpath-all
    optimal-route-reflection apply orr-grp-1
    optimal-route-reflection apply orr-grp-2
!

/* Configure BGP neighbor groups*/
neighbor-group PR
  address-family ipv4 unicast
    route-reflector-client
    optimal-route-reflection orr-grp-1
    route-policy pass in
  !
!
neighbor-group BB
  address-family ipv4 unicast
    route-reflector-client
    route-policy PR_BB_COMM in
  !
!

/*Apply the neighbor groups*/
neighbor 10.10.10.10
  use neighbor-group PR
neighbor 3.3.3.3
  use neighbor-group BB
```

Step 8 Verify the BGP ORR with the add path policy configuration using the **show bgp** command.

Example:

```
Router# sh bgp 200.1.1.0/24 path-elements
BGP routing table entry for 200.1.1.0/24
Versions:
    Process     bRIB/RIB SendTblVer
    Speaker           84         84
        Flags: 0x043e3028+0x00000000;
Last Modified: Sep 7 20:21:38.000 for 00:11:45
Paths: (2 available, best #1)
Path-elements: 2
    Path ID: 1
        Gateway metric 0, Version 84
    Path: Nexthop 11.1.1.2, flags 0x4000000001440207
          Neighbor 11.1.1.2, Received Path ID 0
    Flags: 0x00000001
           status: valid
           path type: bestpath
           add-path action:
    Opaque: pelem=0xebf81d10
            net=0xabe435c, tblattr=0xa962f64 (ver 84)
            path=0xabf4450, path-tblattr=0xa962f64 (ver 84)
                       nobestpath-tblattr=0xa963234 (ver 0)
                        noaddpath-tblattr=0xa9631e4 (ver 0)
            bitfields=0xace9f64 (val=0x1a, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0xac145d8 (val=0x0, size=0)
            orr-ap-bitfields=0xac145d8 (val=0x2, size=0). <<<<<<<<<<<<<<<
            net-next=0xebf81cd4, tblattr-prev=0xebf81cd4, tblattr-next=0xebf81c98
Path ID: 2
    Gateway metric 0, Version 107
    Path: Nexthop 192.168.0.10, flags 0x4000000000040007
```

```
      Neighbor 192.168.0.2, Received Path ID 2
Flags: 0x00000042
       status: valid
       path type: backup
       add-path action: advertise
Opaque: pelem=0x10a0bba8
        net=0x109cbb14, tblattr=0x10399c84 (ver 156)
        path=0x109ebcd4, path-tblattr=0x10399c84 (ver 156)
                    nobestpath-tblattr=0x10399fa4 (ver 0)
                        noaddpath-tblattr=0x10399f54 (ver 0)
        bitfields=0x10c00624 (val=0x10, size=1)
        pe-bitfields=0x0 (val=0x0, size=0)
        orr-bitfields=0x10a2b8cc (val=0x0, size=1)
        orr-ap-bitfields=0x10a2b8a8 (val=0x6, size=1) <<<<<<<<<<<<<<<
        net-next=0x10a0b528, tblattr-prev=0x10a0bbe8, tblattr-next=0x0
```

Support for bgp bestpath igp-metric ignore Command

The **bgp bestpath igp-metric ignore** command enables the system to ignore Interior Gateway Protocol (IGP) metric while selecting the best path.

By default, BGP always prefers a path with the lowest IGP metric. When there are two paths, one with the IGP metric and the other without, then executing the **bgp bestpath igp-metric ignore** command results in BGP performing best path computation as if neither paths has the IGP metric.

The following example shows how to configure the software to ignore the interior gateway protocol (IGP) metric when performing best-path selection:. In this example, the command is configured in router BGP VRF configuration mode.

```
RP/0/0/CPU0:router#configure
RP/0/0/CPU0:router(config)#router bgp 50000
RP/0/0/CPU0:router(config-bgp)#vrf 1
RP/0/0/CPU0:router(config-bgp-vrf)#bgp bestpath igp-metric ignore
```

BFD for Multihop Paths

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Multihop BFDv4 and BFDv6 for iBGP and eBGP	Release 7.3.1	This feature provides sub-second forwarding failure detection for a destination more than one hop, and up to 255 hops away. This feature is supported on all the media-types that support BFD singlehop.

BFD multihop (BFD-MH) is a BFD session between two addresses that are not on the same subnet. An example of BFD-MH is a BFD session between PE and CE loopback addresses or BFD sessions between routers that are several TTL hops away. External and internal BGP applications support BFD multihop. BFD multihop supports BFD on arbitrary paths, which can span multiple network hops.

The BFD for Multihop Paths feature is supported on all the media-types that support BFD singlehop.

Note The Multihop BFDv4 and BFDv6 for iBGP and eBGP feature is not supported over MPLS/GRE Tunnel/SR.

Setting up BFD Multihop Session

A BFD multihop session is set up between a unique source-destination address pair provided by the client. A session can be set up between two endpoints that have IP connectivity. IPv4 addresses in both global routing table and in a VRF table are supported.

Note Aggressive timer is not recommended to use for the BFD Multipath sessions and the Multihop sessions. The recommend time is more than 100 ms x 3 = 300 ms.

Configure BFD IPv6 Multihop Session

When BFD is used with BGP, the BFD session type (singlehop or multihop) is configured based on the BGP configuration. If you configure eBGP-multihop keyword, the BFD session will also run in multihop mode; otherwise the session will run in singlehop mode.

Use the **bfd multihop ttl-drop-threshold** command to drop BFD packets coming from neighbors exceeding a certain number of hops.

- Configure BFD IPv6 multihop for eBGP neighbors
- Configure BFD IPv6 multihop for iBGP neighbors
- Enable BFD on BGP neighbor

Configure BFD IPv6 multihop for eBGP neighbors

```
Router# configure
Router(config)# bfd multipath include location 0/7/CPU0
Router(config)# router bgp 65001
Router(config-bgp)# neighbor 21:1:1:1:1:1:1:2 ebgp-multihop 255
Router(config-bgp)# neighbor 21:1:1:1:1:1:1:2 bfd fast-detect
```

Configure BFD IPv6 multihop for iBGP neighbors

```
Router# configure
Router(config)# bfd multipath include location 0/7/CPU0
Router(config)# router bgp 65001
Router(config-bgp)# neighbor 21:1:1:1:1:1:1:2
```

Enable BFD on a BGP neighbor

```
Router# configure
Router(config)# router bgp 120
Router(config-bgp)# bfd minimum-interval 6500
Router(config-bgp)# bfd multiplier 7
Router(config-bgp)# neighbor 172.168.40.24
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# bfd fast-detect
```

Running Configuration

The following is the running configuration for BFD IPv6 Multihop for eBGP neighbors:

```
Router# show running-configuration
bfd multipath include location 0/7/CPU0
router bgp 65001
neighbor 21:1:1:1:1:1:1:2 ebgp-multihop 255
neighbor 21:1:1:1:1:1:1:2 bfd fast-detect
```

The following is the running configuration for BFD IPv6 Multihop for iBGP neighbors:

```
Router# show running-configuration

bfd multipath include location 0/7/CPU0
router bgp 65001
neighbor 21:1:1:1:1:1:1:2
```

The following is the running configuration for BFD IPv6 Multihop for iBGP neighbors:

```
Router# show running-configuration

router bgp 120
 bfd minimum-interval 6500
 bfd multiplier 7
 neighbor 172.168.40.24
 remote-as 2002
 bfd fast-detect
```

Verification

```
Router# show bfd session
Tue Apr  7 06:16:36.982 UTC

Src Addr  Dest Addr VRF Name H/W NPU Local det time(int*mult)State

10.1.1.1  192.0.2.1 default  No  n/a  n/a  150ms(50ms*3)    UP
10.1.1.2  192.0.2.2 default  No  n/a  n/a  150ms(50ms*3)    UP
10.1.1.3  192.0.2.3 default  No  n/a  n/a  150ms(50ms*3)    UP
10.1.1.4  192.0.2.4 default  No  n/a  n/a  150ms(50ms*3)    UP


Router# show bfd ipv6 session
Tue Apr  7 06:16:45.012 UTC
Src Addr     Dest Addr         VRF Name Local det time(int*mult) State  Echo  Async
----------- -----------------  -------- ----------------------  -----  ----  -----
2001:DB8::1 2001:DB8:0:ABCD::1 default  0s(0s*0) 150ms(50ms*3)    UP
2001:DB8::2 2001:DB8:0:ABCD::2 default  0s(0s*0) 150ms(50ms*3)    UP
2001:DB8::3 2001:DB8:0:ABCD::3 default  0s(0s*0) 150ms(50ms*3)    UP
2001:DB8::4 2001:DB8:0:ABCD::4 default  0s(0s*0) 150ms(50ms*3)    UP
```

CVAC - Bootstrap Configuration Support

Cisco Virtual Appliance Configuration (CVAC) is an out-of-band configuration mechanism supported by several Cisco virtual routers. CVAC receives the configuration injected into the virtual router environment on a CD-ROM, disk image, or USB drive provided by the hypervisor. The configuration is detected and applied at startup time.

This allows the user to combine a new virtual router with a startup (bootstrap) configuration for initial deployment and is a very quick and convenient way of configuring many basic requirements (for example, the management ip address) that typically would have to be done manually.

Note CVAC works if there is no existing configuration, including the initial username and password you are prompted for on a new system.

Cisco IOS XRv 9000 Router fully supports CVAC with native KVM, Openstack Config Drive, and Virsh. CVAC is not supported on VMware ESXi.

Building the Bootstrap Configuration ISO

Cisco IOS XRv 9000 Router supports a plain-text configuration file on a single CD-ROM drive:

- **iosxr_config.txt**—provides standard XR configuration

This text file provides a simple list of configuration CLIs for CVAC to apply automatically. This operation is functionally equivalent to manually issuing a **copy iosxr_config.txt running-config** command.

Given one or more configuration files, you can create an ISO image suitable for insertion into Cisco IOS XRv 9000 Router with the following command:

```
mkisofs -output bootstrap.iso -l -V config-1 --relaxed-filenames --iso-level 2
iosxr_config.txt
```

Here is the sample output for **mkisofs** command on Ubuntu:

```
Warning: creating filesystem that does not conform to ISO-9660.
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
175 extents written (0 MB)
```

CVAC and KVM

In order to have CVAC process the config, add an extra drive to the Qemu command line (as the last drive):

```
-drive file=./bootstrap.iso,if=virtio,media=cdrom,index=3
```

If the configuration files are correctly provided and CVAC runs successfully, you will see these syslog messages:

```
  RP/0/0/CPU0:Dec 14 09:10:22.719 : config[1]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
committed by user  'CVAC'
Use 'show configuration commit changes 1000000001' to view the  changes.


  RP/0/0/CPU0:Dec 14 09:10:23.619 : cvac[2]:  %MGBL-CVAC-5-CONFIG_DONE :
```

Configuration was applied from file /disk0:/iosxr_config.txt.

If any configuration from the config file(s) is rejected, you will additionally see this syslog message:

```
RP/0/0/CPU0:Dec 14 09:10:23.619 : cvac[2]: %MGBL-CVAC-3-CONFIG_ERROR :  Errors were
encountered while applying configs from file  /etc/sysconfig/iosxr_config.txt. Please inspect
 'show configuration  failed' for details
```

Any configuration that did not fail is committed. Further debugging is available in the disk0:/cvac.log file.

CVAC and Virsh

1. Create an ISO image suitable for insertion into Cisco IOS XRv 9000 Router and follow the procedure explained in section Building the Bootstrap Configuration ISO.
2. The Virsh.xml file that is downloaded along with router's image has a Bootstrap section as shown here:

```
<!-- BootstrapSection -->
<!-- Example Bootstrap CLI ISO -->
<!-- <disk type='file' device='cdrom'> -->
<!-- <driver name='qemu' type='raw'/> -->
<!-- <source file='<ISO with file iosxr_config.txt'/> -->
<!-- <target dev='vdc' bus='virtio'/> -->
<!-- <readonly/> -->
<!-- <alias name='bootstrap_CLI'/> -->
<!-- </disk> -->
```

3. Uncomment the Bootstrap section and locate the source file to the absolute path of the bootstrap ISO file on the machine launching the instance. For example:

```
<!--BootstrapSection -->
 <disk type='file' device='cdrom'>
 <driver name='qemu' type='raw'/>
 <source file='/production/bootstrap.iso'/>
 <target dev='vdc' bus='virtio'/>
 <readonly/>
 <alias name='bootstrap_CLI'/>
 </disk>
```

CVAC and OpenStack Config-Drive

Config-drive is a mechanism to bootstrap a VM orchestrated in OpenStack with an initial configuration. Click here to know more about OpenStack config-drive.

If config-drive support is desired, a plaintext file with the initial XR configuration can be passed using this command line:

```
config-drive true user-data /iosxr_config.txt file /iosxr_config.txt=/iosxr_config.txt
```

The file *iosxr_config.txt* is the raw text file with the XR commands, not an ISO file used for the KVM command line or Virsh in earlier sections.

CVAC on Boot

On a newly installed VM instance with no configuration, CVAC behavior is straightforward: all configurations accepted through the parser are committed. CVAC maintains a signature (CRC) of the last applied config file.

On subsequent reboots, any CVAC config file passed into the system has its CRC checked against the last applied CVAC configuration CRC. If there is no change, nothing is done. This means that after an initial CVAC applied configuration, and subsequent configuration changes, if a CVAC configuration file is passed and it hasn't change, then the system configuration is not changed (or reverted).

If there is a change, the new configuration is applied over any existing configuration. This allows an initial CVAC configuration, subsequent configuration changes, and additional CVAC configuration changes on an already configured system.

CVAC does not try to rationalize any of the configurations are just fed into the parser over the already committed configuration. Any commands that pass the parser are committed, and, as mentioned previously, errors are noted in the log file.

ORR Support for FlexAlgo

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
ORR Support for FlexAlgo	Release 7.5.1	This feature allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

Enabling Segment Routing Flexible Algorithm

Segment Routing Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

The SR architecture associates prefix-SIDs to an algorithm which defines how the path is computed. Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraint.

This document describes the IS-IS extension to support Segment Routing Flexible Algorithm on an MPLS data-plane.

Prerequisites for Flexible Algorithm

Segment routing must be enabled on the router before the Flexible Algorithm functionality is activated.

Building Blocks of Segment Routing Flexible Algorithm

This section describes the required building blocks that support the SR Flexible Algorithm functionality in IS-IS.

Flexible Algorithm Definition

Computing a path over a network require the use of many possible constraints. Some networks require the deployment of multiple planes. Simple form of constraint may be to use a particular plane. A more sophisticated form of constraint can include some extended metric, like delay, as described in [RFC7810]. Even more advanced case could be to restrict the path and avoid links with certain affinities. Combinations of these are also possible. You can define the mapping between the algorithm value and its meaning to provide maximum flexibility. When all the routers in the domain have the common understanding what the particular algorithm value represents, the computation for such an algorithm is consistent and the traffic is not subject to looping. In Flexible Algorithm, you define the meaning of the algorithm, not a standard.

Flexible Algorithm Membership

An algorithm defines how IGP computes the best path. Routers advertise the support for the algorithm as a node capability. Routers advertise the prefix-SIDs with algorithm value. Routers tightly couple the prefix-SIDs with the algorithm itself.

An algorithm is a one octet value. Routers reserve values 128 -255 for user-defined values. The routers use them Flexible Algorithm representation.

Flexible Algorithm Definition Advertisement

To guarantee the loop-free forwarding for paths computed for a particular Flexible Algorithm, all routers in the network must share the same definition of the Flexible Algorithm. This is achieved by one or more dedicated routers advertising the definition of each Flexible Algorithm. Such an advertisement is associated with the priority to make sure that all routers agree on a single and consistent definition for each Flexible Algorithm.

Definition of Flexible Algorithm includes:

- Metric type
- Affinity constraints

Use the **advertise-definition** command to enable the router to advertise the definition for the particular Flexible Algorithm. At least one router in the area, preferably two for redundancy, must advertise the Flexible Algorithm definition. The Flexible Algorithm will not be functional unless the router advertises the valid definition.

Flexible Algorithm Link Attribute Advertisement

The router used various link attributes during the Flexible Algorithm path calculation. For example, include or exclude rules that are based on link affinities can be part of the Flexible Algorithm definition, as defined in IETF draft.

Link attribute advertisements that the router uses during Flexible Algorithm calculation must use the Application-Specific Link Attribute (ASLA) advertisements, as defined in RFC8919 (IS-IS). In IS-IS, if the router sets the L-Flag in the ASLA advertisement, then the router uses legacy advertisements (IS-IS Extended availability TLV) instead.

The mandatory use of ASLA advertisements applies to the following link attributes:

- Minimum Unidirectional Link Delay
- TE Default Metric
- Administrative Group

Flexible Algorithm Prefix-SID Advertisement

To forward traffic on a Flexible Algorithm-specific path, all routers participating in the Flexible Algorithm install an MPLS labeled path for the Flexible Algorithm specific SID that the routers advertise for the prefix. Only those prefixes for which the routers advertise, the Flexible Algorithm specific Prefix-SID is subject to Flexible Algorithm-specific forwarding.

Calculation of Flexible Algorithm Path

A router may compute the paths for multiple Flexible Algorithms. Configure a router to support a particular Flexible Algorithm before it can compute any path for such Flexible Algorithm. A router must have a valid definition of the Flexible Algorithm before it uses the Flexible Algorithm.

When computing the shortest path tree for a particular Flexible Algorithm:

- The routers prune from the topology all the nodes that do not advertise support for the Flexible Algorithm.
- If the Flexible Algorithm definition includes excluded affinities, then the routers prune all the links from the topology for which the routers advertise any of such affinities.
- Router uses the metric that is part of the Flexible Algorithm definition. If the routers do not advertise the metric for the particular link, the routers prune the link from the topology.

The routers compute Loop Free Alternate (LFA) paths, TI-LFA backup paths, and Microloop Avoidance paths for a particular Flexible Algorithm. The routers use the same constraints as the calculation of the primary paths for such Flexible Algorithm. These paths use Prefix-SIDs advertised specifically for such Flexible Algorithm in order to enforce a back-up or microloop avoidance path.

Configuring Microloop Avoidance for Flexible Algorithm

By default, the Microloop Avoidance per Flexible Algorithm instance follows the Microloop Avoidance configuration for algo-0.

You can disable Microloop Avoidance for Flexible Algorithm using the following commands:

```
router isis instance flex-algo algo microloop avoidance disable
router ospf process flex-algo algo microloop avoidance disable
```

Configuring LFA/TI-LFA for Flexible Algorithm

By default, a LFA/TI-LFA per Flexible Algorithm instance follows LFA/TI-LFA configuration for algo-0.

You can disable TI-LFA for Flexible Algorithm using the following commands:

```
router isis instance flex-algo algo fast-reroute disable
router ospf process flex-algo algo fast-reroute disable
```

Installation of Forwarding Entries for Flexible Algorithm Paths

The routers must install a Flexible Algorithm path to any prefix in the forwarding using the Prefix-SID that the routers advertised for such Flexible Algorithm. If you do not know the Prefix-SID for Flexible Algorithm, the routers do not install such Flexible Algorithm path in forwarding for such prefix.

The routers install only MPLS to MPLS entries for a Flexible Algorithm path. The routers do not install IP to IP or IP to MPLS entries. These follow the native IPG paths computed based on the default algorithm and regular IGP metrics.

Flexible Algorithm Prefix-SID Redistribution

Previously, the routers limited the prefix redistribution from IS-IS to another IS-IS instance or protocol to SR algorithm 0 (regular SPF) prefix SIDs. The routers did not redistribute SR algorithm 1 (Strict SPF) and SR algorithms 128–255 (Flexible Algorithm) prefix SIDs along with the prefix. The Segment Routing IS-IS Flexible Algorithm Prefix SID Redistribution feature allows redistribution of strict and flexible algorithms prefix SIDs from IS-IS to another IS-IS instance or protocols. The routers automatically enable this feature when you configure redistribution of IS-IS Routes with strict or Flexible Algorithm SIDs.

Flexible Algorithm Prefix Metric

A limitation of the existing Flexible Algorithm functionality in IS-IS is the inability to compute the best path to a prefix in a remote area or remote IGP domain. The routers advertise prefixes between IS-IS areas or between protocol domains, but the existing prefix metric does not reflect any of the constraints of the Flexible Algorithm path. Although you can compute the best Flexible Algorithm path to the interarea or redistribute the prefix inside the area, the path may not represent the overall best path through multiple areas or IGP domains.

The Flexible Algorithm Prefix Metric feature introduces a Flexible Algorithm-specific prefix-metric in the IS-IS prefix advertisement. The prefix-metric provides a way to compute the best end-to-end Flexible Algorithm optimized paths across multiple areas or domains.

Note The Flexible Algorithm definition must be consistent between domains or areas.

Configuring Flexible Algorithm

Use the following IS-IS configuration submode to configure the Flexible Algorithm:

```
router isis instance flex-algo algo
```

Note Always use the IGP metric. If you enable the delay or TE metric, use the advertised delay or TE metric on the link as a metric for Flexible Algorithm computation.

```
router isis  instance flex-algo algo affinity { include-any | include-all | exclude-any}
name1, name2, …
```

name—Name of the affinity map

```
router isis instance flex-algo algo priority priority-value
```

priority value—Priority used during the Flexible Algorithm definition election.

Configure the following command to include the Flexible Algorithm prefix metric in the advertised Flexible Algorithm definition in IS-IS :

```
router isis instance flex-algo algo prefix-metric
```

Configure the following command to enable advertisement of the Flexible Algorithm definition in IS-IS:

```
router isis  instance flex-algo algo advertise-definition
```

Configuring Affinity

Configure the following command for defining the affinity-map. Affinity-map associates the name with the particular bit positions in the Extended Admin Group bitmask.

```
router isis  instance flex-algo algo affinity-map name bit-position bit number
```

- *Name*—Name of the affinity-map.
- *bit number*—Bit position in the Extended Admin Group bitmask.

Configure the following command to associate the affinity with an interface:

```
router isis  instance interface type interface-path-id affinity flex-algo name 1, name 2,
…
```

name—Name of the affinity-map

Configuring Prefix-SID Advertisement

Configure the following command to advertise prefix-SID for the default and strict-SPF algorithm:

```
router isis instance interface type interface-path-id address-family {ipv4 | ipv6} [unicast]
 prefix-sid [strict-spf | algorithm algorithm-number] [index | absolute] sid value
```

- *Algorithm-number*—Flexible Algorithm number
- *SID value*—SID value

Configuring Flexible Algorithm Definitions

Configure the following commands to configure the Flexible Algorithm definition under the flex-algo submode:

```
router isis instance flex-algo algo metric-type {delay | te}
```

Example: Configuring IS-IS Flexible Algorithm

```
Example: Configuring IS-IS Flexible Algorithm
router isis 1
 affinity-map red bit-position 65
 affinity-map blue bit-position 8
 affinity-map green bit-position 201

 flex-algo 128
  advertise-definition
  affinity exclude-any red
  affinity include-any blue
 !
 flex-algo 129
  affinity exclude-any green
 !
!
address-family ipv4 unicast
 segment-routing mpls
!
interface Loopback0
 address-family ipv4 unicast
  prefix-sid algorithm 128 index 100
  prefix-sid algorithm 129 index 101
!
!
interface GigabitEthernet0/0/0/0
 affinity flex-algo red
!
interface GigabitEthernet0/0/0/1
 affinity flex-algo blue red
!
interface GigabitEthernet0/0/0/2
 affinity flex-algo blue
!
```

BGP Routes on PE – Color Based Steering

SR-TE On Demand Next-Hop (ODN) feature can be used to steer the BGP traffic towards the Flexible Algorithm paths.

The following example configuration shows how to setup BGP steering local policy, assuming two router: R1 (2.2.2.2) and R2 (4.4.4.4), in the topology.

Configuration on router R1:

```
vrf Test
address-family ipv4 unicast
  import route-target
   1:150
  !
  export route-policy SET_COLOR_RED_HI_BW
  export route-target
   1:150
  !
!
!
interface Loopback0
ipv4 address 2.2.2.2 255.255.255.255
!
interface Loopback150
vrf Test
ipv4 address 2.2.2.222 255.255.255.255
!
interface TenGigE0/1/0/3/0
description exr1 to cxr1
ipv4 address 10.0.20.2 255.255.255.0
!
extcommunity-set opaque color129-red-igp
  129
end-set
!
route-policy PASS
  pass
end-policy
!
route-policy SET_COLOR_RED_HI_BW
  set extcommunity color color129-red-igp
  pass
end-policy
!
router isis 1
is-type level-2-only
net 49.0001.0000.0000.0002.00
log adjacency changes
affinity-map RED bit-position 28
flex-algo 128
  priority 228
!
address-family ipv4 unicast
  metric-style wide
  advertise link attributes
  router-id 2.2.2.2
  segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 2
   prefix-sid algorithm 128 index 282
  !
```

```
!
interface TenGigE0/1/0/3/0
  point-to-point
  address-family ipv4 unicast
  !
!
!
router bgp 65000
bgp router-id 2.2.2.2
address-family ipv4 unicast
!
address-family vpnv4 unicast
  retain route-target all
!
neighbor-group RR-services-group
  remote-as 65000
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
!
neighbor 4.4.4.4
  use neighbor-group RR-services-group
!
vrf Test
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
segment-routing
traffic-eng
  logging
   policy status
  !
  segment-list sl-cxr1
   index 10 mpls label 16294
  !
  policy pol-foo
   color 129 end-point ipv4 4.4.4.4
   candidate-paths
    preference 100
     explicit segment-list sl-cxr1
     !
    !
   !
  !
!
!

Configuration on router R2:

vrf Test
address-family ipv4 unicast
  import route-target
   1:150
  !
  export route-policy SET_COLOR_RED_HI_BW
  export route-target
   1:150
  !
!
!
interface TenGigE0/1/0/1
```

```
description cxr1 to exr1
ipv4 address 10.0.20.1 255.255.255.0
!
extcommunity-set opaque color129-red-igp
  129
end-set
!
route-policy PASS
  pass
end-policy
!
route-policy SET_COLOR_RED_HI_BW
  set extcommunity color color129-red-igp
  pass
end-policy
!
router isis 1
is-type level-2-only
net 49.0001.0000.0000.0004.00
log adjacency changes
affinity-map RED bit-position 28
affinity-map BLUE bit-position 29
affinity-map GREEN bit-position 30
flex-algo 128
  priority 228
!
flex-algo 129
  priority 229
!
flex-algo 130
  priority 230
!
address-family ipv4 unicast
  metric-style wide
  advertise link attributes
  router-id 4.4.4.4
  segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 4
   prefix-sid algorithm 128 index 284
   prefix-sid algorithm 129 index 294
   prefix-sid algorithm 130 index 304
  !
!
interface GigabitEthernet0/0/0/0
  point-to-point
  address-family ipv4 unicast
  !
!
interface TenGigE0/1/0/1
  point-to-point
  address-family ipv4 unicast
  !
!
router bgp 65000
bgp router-id 4.4.4.4
address-family ipv4 unicast
!
address-family vpnv4 unicast
!
neighbor-group RR-services-group
  remote-as 65000
```

```
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
!
neighbor 1.1.1.1
  use neighbor-group RR-services-group
!
neighbor 2.2.2.2
  use neighbor-group RR-services-group
!
vrf Test
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
  neighbor 25.1.1.2
   remote-as 4
   address-family ipv4 unicast
    route-policy PASS in
    route-policy PASS out
   !
  !
!
!
segment-routing
!
end
```

Configuring Flexible Algorithm

The following ISIS configuration sub-mode is used to configure Flexible Algorithm:

router isis *instance* **flex-algo** *algo*

router ospf *process* **flex-algo** *algo*

algo—value from 128 to 255

Configuring Flexible Algorithm Definitions

The following commands are used to configure Flexible Algorithm definition under the flex-algo sub-mode:

- **metric-type delay**

Note By default the regular IGP metric is used. If delay metric is enabled, the advertised delay on the link is used as a metric for Flexible Algorithm computation.

- **affinity exclude-any** *name1, name2, …*

  name—name of the affinity map

- **priority** *priority value*

  priority value—priority used during the Flexible Algorithm definition election.

The following command is used to enable advertisement of the Flexible Algorithm definition in IS-IS:

```
router isis instance flex-algo algo advertise-definition
```

Configuring Affinity

The following command is used for defining the affinity-map. Affinity-map associates the name with the particular bit positions in the Extended Admin Group bitmask.

```
router isis instance flex-algo algo affinity-map name bit-position bit number
```

```
router ospf process flex-algo algo affinity-map name bit-position bit number
```

- *name*—name of the affinity-map.
- *bit number*—bit position in the Extended Admin Group bitmask.

The following command is used to associate the affinity with an interface:

```
router isis instance interface type interface-path-id affinity flex-algo name 1, name 2, …
```

```
router ospf process area area interface type interface-path-id affinity flex-algo name 1,
name 2, …
```

name—name of the affinity-map

Configuring Prefix-SID Advertisement

The following command is used to advertise prefix-SID for default and strict-SPF algorithm:

```
router isis instance interface type interface-path-id address-family {ipv4 | ipv6} [unicast]
 prefix-sid [strict-spf | algorithm algorithm-number]  [index | absolute] sid value
```

```
router ospf process area area interface Loopback interface-instance prefix-sid [strict-spf
 | algorithm algorithm-number]  [index | absolute] sid value
```

- *algorithm-number*—Flexible Algorithm number
- *sid value*—SID value

IPv4 and IPv6 Traffic Redirect using Policy based Routing

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
IPv4 and IPv6 traffic redirect using Policy Based Routing (PBR)	Release 7.3.3	This feature allows you to redirect IPv4 and IPv6 subscriber traffic to a destination of your choice instead of the one destined originally. With this functionality, you get the flexibility to assign specific traffic through specialized paths, allowing you to efficiently manage service networks carrying voice, video, and data.

In today's converged networks carrying voice, video and data, the requirement to route traffic through specific paths instead of using paths that are from the routing protocols is increasing. The PBR Redirect feature caters to this need by redirecting subscriber traffic to a destination other than the original. This is an additional functionality to the Policy Based Routing (PBR) feature where the packet forwarding decisions are based on the policy configuration, instead of routing protocols.

For example, a service provider may want voice traffic to traverse through certain specialized link and data traffic through the regular routing path. Policy based redirect feature provides a mechanism that lets the service providers to redirect traffic based on a set of preconfigured match criteria into the IPv4 or IPv6 next-hop address.

General Guidelines

- PBR redirect supports IPv4 and IPv6 traffic types.
- PBR redirect and GRE features are mutually exclusive.
- PBR redirect supports destination and source address match.
- The router drops all the matching traffic when the next-hop address for PBR redirect isn't reachable.
- PBR redirect supports the following redirect types:
  - IPv4 and IPv6 next-hop.
  - Default VRF from named VRF
  - Named VRF from default VRF
- PBR redirect supports VRF scaling.
- You can use the following commands to view the PBR policies and associated class maps:
  - **show pbr-pal km policy** *name* **vmr interface** *type* **sw**
  - **show pbr-pal km policy** *name* **info location** *ID*

Configuration

Use the following sample configuration to configure ACLs with PBR.

```
/* Configure an access list */

Router(config)# ipv4 access-list Test
Router(config-ipv4-acl)# 10 permit ipv4 any host 10.1.1.10
Router(config-ipv4-acl)# 20 permit ipv4 any host 10.2.3.4
Router(config-ipv4-acl)# commit
Router(config-ipv4-acl)# exit

/* Configure a class map for the access list */
Router(config)# class-map type traffic match-any Test A
Router(config-cmap)# match access-group ipv4 Test
Router(config-cmap)# end-class-map
Router(config)# commit

/* Configure an PBR policy map with the class map */
Router(config)# policy-map type pbr Test B
Router(config-pmap)# class type traffic Test A
Router(config-pmap-c)# redirect nexthop 192.168.10.1
Router(config-pmap-c)# exit
Router(config-pmap)# end-policy-map
```

Running Configuration

Validate the configuration by using the show command.

```
Router(config)# show running-config
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Mon Nov  6 17:31:23 2017 by UNKNOWN
!
ipv4 access-list Test
 10 permit ipv4 any host 10.1.1.10
 20 permit ipv4 any host 10.2.3.4
!
!
class-map type traffic match-any Test A
 match access-group ipv4 Test
 end-class-map
!
!
policy-map type pbr Test B
 class type traffic Test A
  redirect ipv4 nexthop 192.168.10.1
 !

 end-policy-map
!
```

Verification

The following show commands help in better debuggability for IPv4 and IPv6 traffic redirect using PBR.

```
Router#show pbr-pal km policy policy1 vmr interface GigabitEthernet 0/0/0/0 sw
Wed Feb 16 16:38:03.096 UTC
KM ifname GigabitEthernet0_0_0_0, ul_ifname NULL, policy info name policy1 pnum = 0
ingress_format = 0  egress_format = 1 km policy flags = 0x00001100  class# = 2 ref# = 1
num_intfs = 1 is_bvi 0 is_typhoon_tomahawk 0 np_start 255 np_end 255
================================================================================
```

```
B : type & id      E : ether type     VO : vlan outer     VI : vlan inner
Q : tos/exp/group  X : Reserved       DC : discard class  Fl : flags
F2: L2 flags       F4: L4 flags       SP/DP: L4 ports
T : IP TTL         D : DFS class#      L : leaf class#
Pl: Protocol       G : QoS Grp         M : V6 hdr ext.     C : VMR count

================================================================================
policy name policy1 and format type 0
Total Ingress TCAM entries: 2
 |B    Q  T  Fl Pl SP   DP   G  IPv4/v6 SA IPv4/v6 DA
================================================================================
V|0006 00 00 00 11 0000 0000 00 AC020400 00000000
M|FF00 00 00 80 FF 0000 0000 00 FFFFFF00 00000000
R| C=0 D=0 L=0
V|0006 00 00 00 00 0000 0000 00 00000000 00000000
M|FF00 00 00 00 00 0000 0000 00 00000000 00000000
R| C=1 D=1 L=1

================================================================================
Total Ingress and Egress TCAM entries: 2
```

```
Router#show pbr-pal km policy policy1 info location 0/0/CPU0
Wed Feb 16 16:39:44.503 UTC
KM ifname NULL, ul_ifname NULL, policy info name policy1 pnum = 0 ingress_format = 0
egress_format = 1 km policy flags = 0x00001100  class# = 2 ref# = 1 num_intfs = 1 is_bvi 0
 is_typhoon_tomahawk 0 np_start 255 np_end 255KM policy info
        name policy1
       pnum = 0  ingress_format = 0  egress_format = 1 km policy flags = 0x00001100  class#
 = 2
        ref# = 1 num_intfs = 1
Interface Details
=================
No.  Interface Name                VMR ID   Ingress
1    GigabitEthernet0/0/0/0        1
```

gNMI Bundling of Telemetry Updates

Table 10: Feature History Table

Feature Name	Release Information	Description
gNMI Bundling Size Enhancement	Release 7.8.1	With gRPC Network Management Interface (gNMI) bundling, the router internally bundles multiple gNMI `Update` messages meant for the same client into a single gNMI `Notification` message and sends it to the client over the interface. You can now optimize the interface bandwidth utilization by accommodating more gNMI updates in a single notification message to the client. We have now increased the gNMI bundling size from 32768 to 65536 bytes, and enabled gNMI bundling size configuration through Cisco native data model. Prior releases allowed only a maximum bundling size of 32768 bytes, and you could configure only through CLI. The feature introduces new XPaths to the `Cisco-IOS-XR-telemetry-model-driven-cfg.yang` Cisco native data model to configure gNMI bundling size. To view the specification of gNMI bundling, see Github repository.

To send fewer number of bytes over the gNMI interface, multiple gNMI `Update` messages pertained to the same client are bundled and sent to the client to achieve optimized bandwidth utilization.

The router internally bundles multiple gNMI `Update` messages in a single gNMI `Notification` message of gNMI `SubscribeResponse` message. Cisco IOS XR software Release 7.8.1 supports gNMI bundling size up to 65536 bytes.

Router bundles multiple instances of the same client. For example, a router bundles interfaces `MgmtEth0/RP0/CPU0/0`, `FourHundredGigE0/0/0/0`, `FourHundredGigE0/0/0/1`, and so on, of the following path.

- `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`

Router does not bundle messages of different client in a single gNMI `Notification` message. For example,

- `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters`
- `Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/protocols`

  Data under the container of the client path cannot be split into different bundles.

The gNMI `Notification` message contains a timestamp at which an event occurred or a sample is taken. The bundling process assigns a single timestamp for all bundled `Update` values. The notification timestamp is the first message of the bundle.

Note

- ON-CHANGE subscription mode does not support gNMI bundling.
- Router does not enforce bundling size in the following scenarios:
  - At the end of (N-1) message processing, if the notification message size is less than the configured bundling size, router allows one extra instance which could result in exceeding the bundling size.
  - Data of a single instance exceeding the bundling size.
- The XPath: `network-instances/network-instance/afts` does not support bundling.

QoS on IPv4 GRE Tunnels

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
QoS on IPv4 GRE Tunnels	Release 7.3.3	This feature, which enables the capability to define and control the QoS for both incoming and outgoing customer traffic on provider edge (PE) routers in a service provider network, is introduced.

QoS support on IPv4 GRE tunnels enables applying the policy map directly on the IPv4 GRE interface. This enables aggregate policing and marking on the tunnel. The policy can be applied on the ingress side of the tunnel to mark and police payload traffic going into the tunnel. QoS is not supported on traffic egressing out of the tunnel.

For information on GRE tunnels, see the chapter *Implementing Generic Routing Encapsulation* in the*Cisco ASR 9000 Series Aggregation Services Router MPLS Layer 3 VPN Configuration Guide*.

Note For HQOS license on XRv9000, the license is calculated based on both input and output traffic on the ingress and egress interface only when the service policy is configured on both ingress and egress interface.

Restrictions

The following restrictions apply to QoS on GRE tunnels:

- Any other payload traffic except that of IPv4, IPv6, and MPLS isn't supported.
- QoS on GRE tunnels currently supports only one line card and one network processor.
- The QoS on GRE service policy is supported in the egress direction. You apply an egress service policy at the line card and an ingress policy at the GRE Tunnel.
- The QoS on GRE service policy isn’t supported in the ingress direction.
- The QoS on GRE tunnels functionality doesn't support shaping and queuing actions.

- The QoS on GRE tunnels feature doesn't support marking and conditional marking actions.
- You can't attach a policy map that has percentage-based policing configured.
- The QoS on GRE tunnels feature supports only policing actions. The policer can be a single-rate, two-color policer (1R2C).
- Conform traffic is transmitted, and exceed action is dropped.

Classification for IPv4 GRE tunnel traffic

The following table indicates the support for various payload traffic QoS fields on an IPv4 GRE tunnel for classification.

QoS field	Classification	
	Ingress	Egress
Precedence	No	Yes
Tunnel Precedence	No	No
VLAN	No	No
Class of Service (CoS)	No	No
Drop Eligibility Indicator (DEI)	No	No
IPv4 L3 field	No	Yes

Example

The following example shows application of a two-level hierarchical policy with single-rate, two-color policer (1R2C) on a GRE tunnel.

```
Router(config)#class-map match-any mydata
Router(config-cmap)#match mpls experimental topmost 3 4
Router(config-cmap)#end-class-map
Router(config)#class-map match-any mycontrol
Router(config-cmap)#match mpls experimental topmost 1 2
Router(config-cmap)#end-class-map
Router(config)#policy-map child
Router(config-pmap)#class mycontrol
Router(config-pmap-c)#police rate 2 gbps
Router(config-pmap-c-police)#conform-action transmit
Router(config-pmap-c-police)#exceed-action transmit
Router(config-pmap-c-police)#class class-default
Router(config-pmap-c)#police rate 1 gbps
Router(config-pmap-c-police)#conform-action transmit
Router(config-pmap-c-police)#exceed-action transmit
Router(config-pmap-c-police)#end-policy-map
Router(config)#policy-map parent_gre
Router(config-pmap)#class class-default
Router(config-pmap-c)#service-policy child
Router(config-pmap-c)#police rate 5 gbps
Router(config-pmap-c-police)#child-conform-aware
Router(config-pmap-c-police)#end-policy-map
Router(config)#interface tunnel-ip1
Router(config-if)#service-policy output parent_gre
```

```
Router(config-if)#ipv4 address 12.0.0.1 255.255.255.0
Router(config-if)#tunnel source TenGigE0/0/0/1
Router(config-if)#tunnel destination 15.1.1.2
```

Accessing the Networking Stack

The Cisco IOS XR Software serves as a networking stack for communication. This section explains how applications on IOS XR can communicate with internal processes, and with servers or outside devices.

Note We strongly recommend setting the MTU value of 1514 (default) or higher to avoid dropping fragment packets.

Communication Outside Cisco IOS XR

Table 12: Feature History Table

Feature Name	Release Information	Description
Virtual IP address in the Linux networking stack	Release 7.5.4	Virtual IP addresses allow a single IP address to connect to the current active RP after an RP switchover event. In addition, this functionality enables your network stack to support virtual IP addresses for third-party applications and IOS XR applications that use the Linux networking stack.

To communicate outside Cisco IOS XR, applications use the `fwdintf` interface address that maps to the `loopback0` interface or a configured Gigabit Ethernet interface address. For information on the various interfaces on IOS XR, see Application Hosting on the Cisco IOS XR Linux Shell, on page 91.

To have an iPerf or Chef client on IOS XR communicate with its respective server outside IOS XR, you must configure an interface address as the source address on XR. The remote servers must configure this route address to reach the respective clients on IOS XR.

Virtual addresses can be configured to access a router from the management network such as gRPC using a single virtual IP address. On a device with two or more RPs, the virtual address refers to the management interface that is currently active. This functionality can be used across RP failover without the information of which RP is currently active. This is applicable to the Linux packet path.

This section provides an example of configuring a Gigabit Ethernet interface address as the source address for external communication.

Using a Gigabit Ethernet Interface for External Communication

To configure a GigE interface on IOS XR for external communication, use these steps:

1. Configure a GigE interface.

```
RP/0/RP0/CPU0:ios(config)# interface GigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:ios(config-if)# ipv4 address 192.57.43.10 255.255.255.0
RP/0/RP0/CPU0:ios(config-if)# no shut
RP/0/RP0/CPU0:ios(config-if)# commit
Fri Oct 30 07:51:14.785 UTC
RP/0/RP0/CPU0:ios(config-if)# exit
RP/0/RP0/CPU0:ios(config)# exit
```

2. Verify whether the configured interface is up and operational on IOS XR.

```
RP/0/RP0/CPU0:ios# show ipv4 interface brief
Fri Oct 30 07:51:48.996 UTC

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1         Up                    Up
Loopback1                      8.8.8.8         Up                    Up
GigabitEthernet0/0/0/0         192.164.168.10  Up                    Up
GigabitEthernet0/0/0/1         192.57.43.10    Up                    Up
GigabitEthernet0/0/0/2         unassigned      Shutdown              Down
MgmtEth0/RP0/CPU0/0            192.168.122.197 Up                    Up
RP/0/RP0/CPU0:ios#
```

3. Enter the Linux bash shell and verify if the configured interface is up and running.

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash


[xr-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Gi0_0_0_1 Link encap:Ethernet  HWaddr 52:46:2e:49:f6:ff
          inet addr:192.57.43.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:2eff:fe49:f6ff/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:294 (294.0 B)  TX bytes:504 (504.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
```

```
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:392 (392.0 B)  TX bytes:532 (532.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:672 (672.0 B)  TX bytes:672 (672.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

4. Exit the Linux bash shell and configure the GigE interface as the source address for external communication.

```
[xr-vm_node0_RP0_CPU0:~]$ exit

RP/0/RP0/CPU0:ios# config
Fri Oct 30 08:55:17.992 UTC
RP/0/RP0/CPU0:ios(config)# tpa address-family ipv4 update-source gigabitEthernet 0/0/0/1
RP/0/RP0/CPU0:ios(config)# commit
Fri Oct 30 08:55:38.795 UTC
```

Note By default, the `fwdintf` interface maps to the `loopback0` interface for external communication. This is similar to binding a routing process or router ID to the `loopback0` interface. When you use the `tpa address-family ipv4 update-source` command to bind the `fwdintf` interface to a Gigabit Ethernet interface, network connectivity can be affected if the interface goes down.

5. Enter the Linux bash shell and verify whether the GigE interface address is used by the `fwdintf` interface for external communication.

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash

[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 192.57.43.10
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.197
[xr-vm_node0_RP0_CPU0:~]$
```

External communication is successfully enabled on IOS XR.

East-West Communication for Third-Party Applications

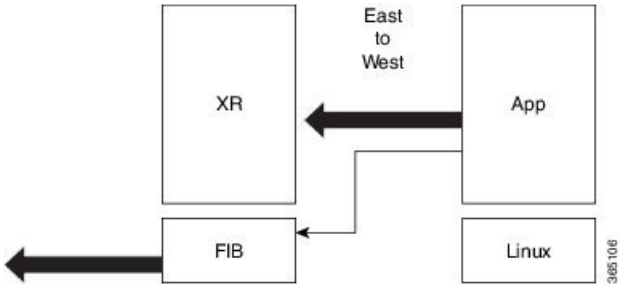
East-West communication on IOS XR is a mechanism by which applications hosted in containers interact with native XR applications (hosted in the XR control plane).

The following figure illustrates how a third-party application hosted on IOS XR interacts with the XR Control Plane.

The application sends data to the Forwarding Information Base (FIB) of IOS XR. The application is hosted in the east portion of IOS XR, while the XR control plane is located in the west region. Therefore, this form of communication between a third-party application and the XR control plane is termed as East-West (E-W) communication.

Third-party applications such as Chef Client and Puppet Agent use this mode of communication to configure and manage containers, packages, and applications on IOS XR. In the future, this support could be extended to IOS XR, configured and managed by such third-party applications.

Figure 6: East-West Communication on IOS XR



For a third-party application to communicate with IOS XR, the Loopback1 interface must be configured. This is explained in the following procedure.

1. Configure the Loopback1 interface on IOS XR.

```
RP/0/RP0/CPU0:ios(config)# interface Loopback1
RP/0/RP0/CPU0:ios(config-if)# ipv4 address 8.8.8.8/32
RP/0/RP0/CPU0:ios(config-if)# no shut
RP/0/RP0/CPU0:ios(config-if)# commit
RP/0/RP0/CPU0:ios(config-if)# exit
RP/0/RP0/CPU0:ios(config)#
```

2. Verify the creation of the Loopback1 interface.

```
RP/0/RP0/CPU0:ios# show ipv4 interface brief
Thu Nov 12 10:01:00.874 UTC

Interface                  IP-Address      Status                Protocol
Loopback0                  1.1.1.1         Up                    Up
Loopback1                  8.8.8.8         Up                    Up
GigabitEthernet0/0/0/0     192.164.168.10  Up                    Up
GigabitEthernet0/0/0/1     192.57.43.10    Up                    Up
GigabitEthernet0/0/0/2     unassigned      Shutdown              Down
MgmtEth0/RP0/CPU0/0        192.168.122.197 Up                    Up
RP/0/RP0/CPU0:ios#
```

3. Enter the third-party network namespace or global VRF depending on the version of IOS XR version you are using for your network.

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash
```

```
/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash
```

4. Verify whether the Loopback1 interface address has been mapped to the E-W interface.

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 192.57.43.10
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.197
[xr-vm_node0_RP0_CPU0:~]$
```

Application Hosting on the Cisco IOS XR Linux Shell

Linux supports an entire ecosystem of applications and tools that have been created, tested, and deployed by system administrators, developers, and network engineers over the last few decades. Linux is well suited for hosting servers with or without applications, because of its stability, security, scalability, reduced cost for licensing, and the flexibility it offers to customize applications for specific infrastructure needs.

With a growing focus on DevOps style workflows that focus on automation and ease of integration, network devices need to evolve and support standard tools and applications that make the automation process easier. A standardized and shared tool chain can boost speed, efficiency, and collaboration. IOS XR is developed from a Yocto-based Wind River Linux 7 distribution. The OS is RPM based and well suited for embedded systems.

IOS XR enables hosting of 64-bit Linux applications on the box, and has the following advantages:

- Seamless integration with configuration management applications
- Easy access to file systems
- Ease of operation

To host a Linux application on IOS XR, you must be familiar with the Linux shell on XR.

A typical Linux OS provides a single set of network interfaces and routing table entries that are shared across the OS. With the introduction of network namespaces, Linux provides multiple instances of network interfaces and routing tables that operate independently.

Note Support for network namespaces varies across different distributions of the Linux OS. Ensure that the distribution you are planning to use for application hosting supports network namespaces.

Network Namespaces on IOS XR

There are two ways of accessing the IOS XR Linux shell, depending on the version of Cisco IOS XR that you are using in your network.

- If you are using **Cisco IOS XR Version 6.0.0**, then you must use the procedure in Accessing the Third-Party Network Namespace on Cisco IOS XR Linux Shell, on page 95. Accessing the XR Linux shell takes you to the default network namespace, XRNNS. You must navigate from this namespace to access the third-party network namespace (TPNNS), where all the third-party application interfaces reside. There is a difference between what you can access and view at the XR router prompt, and what you can access and view at the XR Linux Shell.

- If you are using **Cisco IOS XR Version 6.0.2** and higher, then you must use the procedure in Accessing Global VRF on the Cisco IOS XR Linux Shell, on page 92. Accessing the XR Linux shell takes you directly to the third-party network namespace, renamed as global VRF. You can run bash commands at the XR router prompt itself to view the interfaces and IP addresses stored in global VRF. Navigation is faster and more intuitive in this version of IOS XR.

Accessing Global VRF on the Cisco IOS XR Linux Shell

The Third-Party Network Namespace (TPNNS) is renamed as Global VRF (global-vrf) in Cisco IOS XR Version 6.0.2 and higher. When you access the Cisco IOS XR Linux shell, you directly enter global VRF. This is described in the following procedure.

1. From your Linux box, access the IOS XR console through SSH, and log in.

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

   You have reached the IOS XR router prompt.

2. View the ethernet interfaces on IOS XR.

```
RP/0/0/CPU0:ios# show ipv4 interface brief
...

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1/32      Up                    Up
GigabitEthernet0/0/0/0         10.1.1.1/24     Up                    Up
...


RP/0/RP0/CPU0:ios# show interfaces gigabitEthernet 0/0/0/0
...

GigabitEthernet0/0/0/0 is up, line protocol is up
Interface state transitions: 4
Hardware is GigabitEthernet, address is 5246.e8a3.3754 (bia
5246.e8a3.3754)
Internet address is 10.1.1.1/24
MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation ARPA,
Duplex unknown, 1000Mb/s, link type is force-up
output flow control is off, input flow control is off
loopback not set,
Last link flapped 01:03:50
ARP type ARPA, ARP timeout 04:00:00
Last input 00:38:45, output 00:38:45
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
12 packets input, 1260 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 2 broadcast packets, 0 multicast packets
0 runts, 0 giants, 0 throttles, 0 parity
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
12 packets output, 1224 bytes, 0 total output drops
Output 1 broadcast packets, 0 multicast packets
```

   The output displays the IP and MAC addresses of the `GigabitEthernet0/0/0/0` interface.

3. Verify whether the bash command runs in global VRF by running the **bash -c ifconfig** command to view the network interfaces.

```
RP/0/RP0/CPU0:ios# bash -c ifconfig
...
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

The presence of the following two interfaces confirms that you are in Global VRF:

`fwd_ew` is the interface used for communication (east to west) between third-party applications and IOS XR.

`fwdintf` is the interface used for communication between third-party applications and the network outside IOS XR.

4. Access the Linux shell by running the **bash** command.

```
RP/0/RP0/CPU0:ios# bash
Tue Aug 02 13:44:07.627 UTC
[xr-vm_node0_RP0_CPU0:~]$
```

5. (Optional) View the IP routes used by the `fwd_ew` and `fwdintf` interfaces.

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 1.1.1.1
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.213
```

Alternative Method of Entering Global VRF on IOS XR

To directly enter global VRF on logging to IOS XR, without entering the **bash** command, you can use the `sshd_operns` service, as explained in the steps that follow. The procedure involves the creation of a non-root user in order to access the service. (Root users cannot access this service.)

Note On IOS XR, prior to starting a service that binds to an interface, ensure that the interface is configured, up, and operational.

To ensure that a service starts only after an interface is configured, include the following function in the service script:

```
. /etc/init.d/operns-functions
operns_wait_until_ready
```

The addition of the **operns_wait_until_ready** function ensures that the service script waits for one or more interfaces to be configured before starting the service.

1. (Optional) If you want the operns service to start automatically on reload, add the `sshd_operns` service and verify its presence.

```
bash-4.3# chkconfig --add sshd_operns
bash-4.3# chkconfig --list sshd_operns
sshd_operns      0:off   1:off   2:off   3:on    4:on    5:on    6:off
bash-4.3#
```

2. Start the `sshd_operns` service.

```
bash-4.3# service sshd_operns start
Generating SSH1 RSA host key: [  OK  ]
Generating SSH2 RSA host key: [  OK  ]
Generating SSH2 DSA host key: [  OK  ]
  generating ssh ECDSA key...
Starting sshd: [  OK  ]

bash-4.3# service sshd_operns status
sshd (pid  6224) is running...
```

3. Log into the `sshd_operns` session as the non-root user created in Step 1.

```
host@fe-ucs36:~$ ssh devops@192.168.122.222 -p 57722
devops@192.168.122.222's password:
Last login: Tue Sep  8 20:14:11 2015 from 192.168.122.1
XR-vm_node0_RP0_CPU0:~$
```

4. Verify whether you are in global VRF by viewing the network interfaces.

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

You are ready to use the IOS XR Linux shell for hosting applications.

Accessing the Third-Party Network Namespace on Cisco IOS XR Linux Shell

The Cisco IOS XR Linux shell provides a Third-Party Network Namespace (TPNNS) that provides the required isolation between third-party applications and internal XR processes, while providing the necessary access to XR interfaces for the applications. You can use the steps mentioned in this section to access the IOS XR Linux shell and navigate through the XRNNS (default XR Network Namespace) and the TPNNS.

Note This procedure is applicable only on Cisco IOS XR Versions 5.3.2 and 6.0.0. For accessing this namespace on other versions of Cisco IOS XR, see Accessing Global VRF on the Cisco IOS XR Linux Shell, on page 92.

Use these steps to navigate through the XR Linux shell.

1. From your Linux box, access the IOS XR console through SSH, and log in.

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

You have reached the IOS XR router prompt.

2. View the ethernet interfaces on IOS XR.

```
RP/0/0/CPU0:ios# show ipv4 interface brief
...

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1/32      Up                    Up
GigabitEthernet0/0/0/0         10.1.1.1/24     Up                    Up
...


RP/0/RP0/CPU0:ios# show interfaces gigabitEthernet 0/0/0/0
...

GigabitEthernet0/0/0/0 is up, line protocol is up
Interface state transitions: 4
Hardware is GigabitEthernet, address is 5246.e8a3.3754 (bia
5246.e8a3.3754)
Internet address is 10.1.1.1/24
MTU 1514 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation ARPA,
Duplex unknown, 1000Mb/s, link type is force-up
output flow control is off, input flow control is off
loopback not set,
Last link flapped 01:03:50
ARP type ARPA, ARP timeout 04:00:00
Last input 00:38:45, output 00:38:45
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
12 packets input, 1260 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 2 broadcast packets, 0 multicast packets
0 runts, 0 giants, 0 throttles, 0 parity
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
12 packets output, 1224 bytes, 0 total output drops
Output 1 broadcast packets, 0 multicast packets
```

The output displays the IP and MAC addresses of the `GigabitEthernet0/0/0/0` interface.

3. Enter the **run** command to launch the IOS XR Linux bash shell.

You can also check the version of IOS XR when you are at the bash prompt.

```
RP/0/RP0/CPU0:ios# run
Wed Oct 28 18:45:56.168 IST
```

```
[xr-vm_node0_RP0_CPU0:~]$ uname -a
Linux xr-vm_node0_RP0_CPU0 3.10.19-WR7.0.0.2_standard #1 SMP Mon Jul 6
13:38:23 PDT 2015 x86_64 GNU/Linux
[xr-vm_node0_RP0_CPU0:~]$
```

Note To exit the Linux bash shell and launch the IOS XR console, enter the **exit** command:

```
[xr-vm_node0_RP0_CPU0:~]$ exit
exit
RP/0/RP0/CPU0:ios#
```

4. Locate the network interfaces by running the **ifconfig** command.

```
[xr-vm_node0_RP0_CPU0:~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:280 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31235 (30.5 KiB)  TX bytes:20005 (19.5 KiB)

eth-vf0   Link encap:Ethernet  HWaddr 52:54:00:34:29:44
          inet addr:10.11.12.14  Bcast:10.11.12.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe34:2944/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1566 (1.5 KiB)  TX bytes:1086 (1.0 KiB)

eth-vf1   Link encap:Ethernet  HWaddr 52:54:00:ee:f7:68
          inet6 addr: fe80::5054:ff:feee:f768/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
          RX packets:326483 errors:0 dropped:3 overruns:0 frame:0
          TX packets:290174 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24155455 (23.0 MiB)  TX bytes:215862857 (205.8 MiB)

eth-vf1.1794 Link encap:Ethernet  HWaddr 52:54:01:5c:55:8e
          inet6 addr: fe80::5054:1ff:fe5c:558e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:728 (728.0 B)  TX bytes:1234 (1.2 KiB)

eth-vf1.3073 Link encap:Ethernet  HWaddr e2:3a:dd:0a:8c:06
          inet addr:192.0.0.4  Bcast:192.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::e03a:ddff:fe0a:8c06/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
          RX packets:317735 errors:0 dropped:3560 overruns:0 frame:0
          TX packets:257881 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:18856325 (17.9 MiB)  TX bytes:204552163 (195.0 MiB)

eth-vf1.3074 Link encap:Ethernet  HWaddr 4e:41:50:00:10:01
          inet addr:172.0.16.1  Bcast:172.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::4c41:50ff:fe00:1001/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8996  Metric:1
```

```
          RX packets:8712 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32267 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:723388 (706.4 KiB)  TX bytes:11308374 (10.7 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1635360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1635360 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:182532711 (174.0 MiB)  TX bytes:182532711 (174.0 MiB)

tap123    Link encap:Ethernet  HWaddr c6:13:74:4b:dc:e3
          inet6 addr: fe80::c413:74ff:fe4b:dce3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:998 (998.0 B)
```

The output displays the internal interfaces (`eth0` through `eth-vf1.3074`) used by IOS XR. These interfaces exist in XR Network Namespace (XRNNS) and do not interact with the network outside IOS XR. Interfaces that interact with the network outside IOS XR are found in the Third Party Network Namespace (TPNNS).

5. Enter the TPNNS on the IOS XR bash shell.

```
[XR-vm_node0_RP0_CPU0:~]$ ip netns exec tpnns bash
```

6. View the TPNNS interfaces.

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

The interfaces displayed in the output are replicas of the IOS XR interfaces in the Linux environment. (They have the same MAC and IP addresses.)

- `Gi0_0_0_0` is the IOS XR GigabitEthernet 0/0/0/0 interface.
- `Mg0_RP0_CPU0_0` is the IOS XR management interface, used for administrative operations on XR.
- `fwd_ew` is the interface used for communication (east to west) between third-party applications and IOS XR.
- `fwdintf` is the interface used for communication between third-party applications and the network outside IOS XR.
- `lo:0` is the IOS XR loopback0 interface used for communication between third-party applications and the outside network through the `fwdintf` interface. The loopback0 interface must be configured for applications to communicate outside XR. Alternatively, applications can also configure a GigE interface for external communication, as explained in the Communication Outside Cisco IOS XR, on page 87 section.

All interfaces that are enabled (with the **no shut** command) are added to TPNNS on IOS XR.

7. (Optional) View the IP routes used by the `fwd_ew` and `fwdintf` interfaces.

```
[xr-vm_node0_RP0_CPU0:~]$ ip route
default dev fwdintf  scope link  src 1.1.1.1
8.8.8.8 dev fwd_ew  scope link
192.168.122.0/24 dev Mg0_RP0_CPU0_0  proto kernel  scope link  src 192.168.122.213
```

Alternative Method of Entering the Third Party Network Namespace on IOS XR

To directly enter the TPNNS on logging to IOS XR, without entering the **ip netns exec tpnns bash** command, you can use the `sshd_tpnns` service, as explained in the steps that follow. The procedure involves the creation of a non-root user in order to access the service. (Root users cannot access this service.)

Note On IOS XR, prior to starting a service that binds to an interface, ensure that the interface is configured, up, and operational.

To ensure that a service starts only after an interface is configured, include the following function in the service script:

```
. /etc/init.d/tpnns-functions
tpnns_wait_until_ready
```

The addition of the **tpnns_wait_until_ready** function ensures that the service script waits for one or more interfaces to be configured before starting the service.

1. (Optional) If you want the TPNNS service to start automatically on reload, add the `sshd_tpnns` service and verify its presence.

```
bash-4.3# chkconfig --add sshd_tpnns
bash-4.3# chkconfig --list sshd_tpnns
sshd_tpnns      0:off   1:off   2:off   3:on    4:on    5:on    6:off
bash-4.3#
```

2. Start the `sshd_tpnns` service.

```
bash-4.3# service sshd_tpnns start
Generating SSH1 RSA host key: [  OK  ]
Generating SSH2 RSA host key: [  OK  ]
Generating SSH2 DSA host key: [  OK  ]
  generating ssh ECDSA key...
Starting sshd: [  OK  ]

bash-4.3# service sshd_tpnns status
sshd (pid  6224) is running...
```

3. Log into the `sshd_tpnns` session as the non-root user created in Step 1.

```
host@fe-ucs36:~$ ssh devops@192.168.122.222 -p 57722
devops@192.168.122.222's password:
Last login: Tue Sep  8 20:14:11 2015 from 192.168.122.1
XR-vm_node0_RP0_CPU0:~$
```

4. Verify whether you are in TPNNS by viewing the interfaces.

```
[XR-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
```

```
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

You are ready to use the IOS XR Linux shell for hosting applications.

Class-based MPLS traffic steering

The class-based traffic steering is a network management technique that

- classifies network traffic into distinct categories based on predefined criteria such as protocol header fields or application identification
- applies specific policies and rules to manage and prioritize these classified traffic flows, and
- routes traffic to optimize network performance, reliability, and user experience by reducing congestion and minimizing latency.

Table 13: Feature History Table

Feature Name	Release Information	Feature Description
Class-based MPLS traffic steering	Release 25.3.1	You can now enable precise traffic management to improve network efficiency and control by classifying MPLS traffic using the three-bit EXP field. This feature classifies and steers MPLS traffic based on the EXP value, with the SWAN agent or controller sending MPLS routes to Label Switch Router (LSR) nodes. Each node forwards traffic according to these class-based rules, selecting the next hop based on EXP-defined classes.

Classification and policy configurations

You configure classification and policy by using Quality of Service (QoS), policy manager, and tunnel command-line interface (CLI). When integration with external applications is required, the Service Layer Application Programming Interface (SL-API) component sends commands to the Label Switched Domain (LSD). An agent uses these commands to program classification based on the experimental (EXP) value.

Class-based traffic steering on PBTS

You need to add class-based traffic steering to the existing Policy-Based Tunnel Selection (PBTS). Currently, PBTS works only for the Tunnel case. With class-based traffic steering, the egress interface can be a normal physical interface.

Benefits of class-based MPLS traffic steering

Class-based MPLS traffic steering provides these benefits:

- Enables accurate traffic classification using the three-bit EXP value in MPLS headers, supporting granular service differentiation.
- Improves network efficiency by sorting traffic into classes, such as default or scavenger, to optimize bandwidth usage.
- Integrates seamlessly with existing PBTS, to leverage invested systems and policies.
- Allows flexible traffic steering to both tunnel and physical interfaces, supporting diverse network architectures and operational requirements.

Guidelines and restrictions for class-based traffic steering

Guidelines for class-based traffic steering

These guidelines apply to class-based traffic steering:

- Use class-based traffic steering only on LSR nodes.
- Ensure that EXP-based classification matches your network management policies.
- Use the SL-API to send commands to the LSD for traffic classification.
- Set classification for EXP values from 0 to 7, or specify a default EXP class.
- Integrate class-based traffic steering with your existing QoS, policy manager, and tunnel CLI configurations.

Restrictions for class-based traffic steering

These restrictions apply to class-based traffic steering:

- You can use this feature only on LSR nodes; label edge router (LER) roles are not supported.
- Only EXP-based traffic steering is supported.
- This feature supports only next hop identifier (NHID) cases. As a result, two stage forwarding is not supported.

- This feature allows only one outgoing label at the egress.

How class-based traffic steering works

Summary

This feature enables class-based MPLS traffic steering by using rules that determine how traffic is routed through the network.

Workflow

This feature steers MPLS traffic using class-based rules:

1. The SWAN agent or controller sends MPLS routes to LSR nodes.
2. Class-based steering works on top of existing PBTS infrastructure.
3. Classification separates default traffic from scavenger traffic.
4. The system selects the path based on the traffic class, determined by the EXP value.
5. The next hop matches the defined class to ensure correct path selection.
6. The node uses the MPLS route from the controller to forward network traffic.
7. The egress interface can be either a tunnel or a physical interface.

Configure class-based traffic steering

Class-based traffic steering enables differentiated routing in MPLS networks by mapping traffic classes to MPLS EXP values. This allows you to prioritize critical traffic and assign specific paths to less important traffic, optimizing service quality and resource usage. This approach improves network efficiency and ensures diverse service-level requirements are met.

Before you begin

- Enable Cisco Express Forwarding (CEF) to support MPLS switching.
- Enable MPLS forwarding on all relevant interfaces.
- Verify that network devices can classify and mark MPLS EXP fields.
- Prepare your SWAN agent or network controller to send appropriate MPLS routes to LSDs.

Procedure

Step 1 Send classification and policy configuration commands

Use your CLI interface and the SL-API to send classification and policy commands to each LSD.

Step 2 Define traffic classification rules

Configure rules for MPLS EXP values (0-7) and a default class according to your traffic engineering policy.

If...	Then...
Traffic has an MPLS EXP value of 3	The system routes it along the high-priority path.
Traffic has an MPLS EXP value of 7	The system routes it along the path designated for less critical data.
SL-API rules are defined for specific EXP values	These rules integrate with existing QoS policies for enforcement.

Step 3 Integrate with existing configurations

Ensure that the new traffic steering rules are fully integrated with your current QoS, Policy Management, and Tunnel configurations for seamless operation.

Step 4 Prepare the SWAN agent or controller

Verify that your SWAN controller is configured to advertise MPLS route information to all participating LSDs. Ensure the agent advertises appropriate routes and labels as shown in the example configuration.

Example Configuration:

```
swanagent:
  mpls:
    enabled: true
    advertised_routes:
      - prefix: 198.51.100.0/24
        label: 1001
      - prefix: 203.0.113.0/24
        label: 1002
    lsd_peers:
      - lsd1.example.com
      - lsd2.example.com
```

Step 5 Enable and confirm class-based forwarding on LSDs

Enable the class-based forwarding feature on all LSDs and configure them to match your network's requirements for traffic differentiation and resource allocation.

You can verify the status by viewing logs of the Docker container running the SWAN agent.

Example Configuration:

```
lsd:
  class_based_forwarding:
    enabled: true
    classes:
      - name: 'voice'
        match:
          dscp: 46
        queue: 'high-priority'
      - name: 'video'
        match:
          dscp: 34
        queue: 'medium-priority'
      - name: 'default'
        match:
          dscp: 0
        queue: 'default'
    resource_allocation:
```

```
'high-priority': 40%
'medium-priority': 30%
'default': 30%
```

Note
Adjust the Differentiated Services Code Point (DSCP) values, class names, and resource allocation as required for your deployment.

Class-based traffic steering is now active on the network. Different traffic types are routed along assigned paths as per your policy, providing optimized resource usage and differentiated service levels.

What to do next

- Monitor network logs or dashboards to verify that traffic matches your classification and steering policies.
- Adjust class definitions or resource allocations as business needs change.
- Regularly update your LSD and SWAN configurations to reflect any changes in network topology or policy.

IS-IS auto-cost reference bandwidth

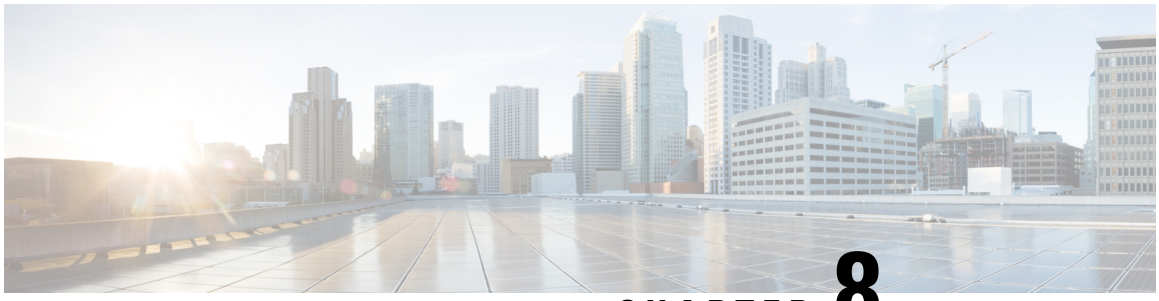
The IS-IS auto-cost reference bandwidth feature is a routing metric mechanism that

- provisions IS-IS metrics based on the physical bandwidth of a link
- adjusts IS-IS metrics for bundle interfaces when bandwidth changes, and
- optimizes route selection by lowering the cost for high-speed links.

This feature enhances network adaptability and ensures that IS-IS routing decisions accurately reflect the current link capacities.

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS auto-cost reference bandwidth	Release 25.4.1	The IS-IS auto-cost reference bandwidth feature automates IS-IS metric provisioning based on physical link bandwidth, optimizing path selection and reducing operational overhead. This feature allows you to configure a reference bandwidth, which IS-IS then uses to automatically calculate interface metrics. It also dynamically adjusts metrics for bundle interfaces when member links change, ensuring accurate and efficient routing without manual intervention. CLI: • **auto-cost (IS-IS)** • **show isis interface**: The Bandwidth field was added in the show output.



CHAPTER 8

Cisco IOS XRv 9000 Data Plane Specific Features

- Early Fast Discard, on page 107
- CEF load balancing with GTP, on page 108
- Data Plane Management, on page 109
- Set interface bandwidth to match SFP speed , on page 114

Early Fast Discard

Table 15: Feature History Table

Feature Name	Release Information	Feature Description
Early Fast Discard	Release 24.1.1	Starting from Cisco IOS XR Release 24.1.1, the early fast discard feature is deprecated and will not be supported in future releases. We recommend not to use this feature starting from Cisco IOS XR Release 24.1.1.

The early fast discard (EFD) is a congestion protection feature to handle cases when the incoming traffic exceeds the router capacity. While the normal congestion control is handled by the traffic manager (TM) and attached QoS policies, the EFD feature is activated in extreme cases. The EFD feature filters high priority traffic (such as keepalives, control, and BFD) and discards the rest at the very early stage of datapath processing, thus maintaining the important flow of control packets.

To activate the EFD feature, use the **early-fast-discard** command in configuration mode. The traffic to be discarded is defined by setting IP precedence, MPLS exp, and VLAN cos values.

Configuring Early Fast Discard

Use Case

Early Fast Discard is configured on the data plane to manage incoming traffic flow by defining the traffic to be discarded based on these conditions:

- IP precedence=4
- MPLS exp=3
- VLAN cos=5

The default value is 6 and ge (greater than or equal to).

Configuration

```
Router# configure
Router(config)# hw-module early-fast-discard
Router(config-early-fast-discard)# ip-prec 4 ip-op [lt | ge]
Router(config-early-fast-discard)# mpls-exp 3 mpls-op [lt | ge]
Router(config-early-fast-discard)# vlan-cos 5 vlan-op [lt | ge]
Router(config-early-fast-discard)# exit
```

Note Use the **no hw-module early-fast-discard** command to deactivate EFD.

Running Configuration

```
RP/0/RP0/CPU0:ios#show run hw-module early-fast-discard
Thu Jul 16 15:51:34.672 UTC
hw-module early-fast-discard
 ip-prec 4 ip-op ge
 mpls-exp 3 mpls-op ge
 vlan-cos 5 vlan-op lt
!
```

Related Topics

- Early Fast Discard, on page 107

CEF load balancing with GTP

This section describes per-flow load balancing.

Per-Flow Load Balancing

Load balancing describes the functionality in a router that distributes packets across multiple links based on Layer 3 (network layer) and Layer 4 (transport layer) routing information. If the router discovers multiple paths to a destination, the routing table is updated with multiple entries for that destination.

Per-flow load balancing performs the following functions:

- Incoming data traffic is evenly distributed over multiple equal-cost connections.
- Layer 3 (network layer) load balancing decisions are taken on IPv4, IPv6, and MPLS flows which are supported for the 5-tuple hash algorithm.
- The same hash algorithm (3-tuple or 5-tuple) is used for load balancing over multiple equal-cost Layer 3 (network layer) paths. The Layer 3 (network layer) path is on a physical interface or on a bundle interface.

- The cef load-balancing fields L3 global command allows you to select the 3-tuple hash algorithm.
- By default, 5-tuple hash algorithm is used for load balancing. If you use the cef load-balancing fields L3 global command, 3-tuple hash algorithm is enabled.

Layer 3 (L3) Routing Information

The 3-tuple load-balance hash calculation contains the following Layer 3 (Network Layer) inputs:

- Source IP address
- Destination IP address
- Router ID

Layer 4 (L4) Routing Information

The 5-tuple load-balance hash calculation contains 3-tuple inputs and these additional following Layer 4 (Transport Layer) inputs:

- Source IP address
- Destination IP address

CEF load balancing for GTP is always enabled by default on XRv9K platform and cannot be disabled. Therefore, the no cef loadbalancing fields l4 gtp command does not disable the inclusion of GTP fields in hash calculation.

XRV9k can detect GTP headers even if it’s behind a GRE header + MPLS Labels and can perform load balancing based on GTP fields.

Data Plane Management

Cisco IOS XRv 9000 Router's data plane starts automatically after the router is booted. In case the control plane does not establish or loses communication with the data plane, then the control plane automatically restarts the data plane.

The data plane can be started, shut-down, and reloaded manually through the admin console for maintenance and troubleshooting purposes. This table lists the commands that are required to manage the data plane.

Table 16: Data Plane Management Commands

Task	Use this command

Check the status of data plane.	show sdr
	`sysadmin-vm:0_RP0# show sdr`
	`Thu May 7 18:38:38.996 UTC`
	`sdr default-sdr`
	`location 0/RP0/VM1`
	`sdr-id 2`
	`IP Address of VM 192.0.0.4`
	`MAC address of VM E2:3A:DD:0A:8C:06`
	`VM State RUNNING`
	`start-time`
	`2015-05-07T17:54:39.457822+00:00`
	`Last Reload Reason FIRST_BOOT`
	`Reboot Count 1`
	`location 0/RP0/VM2`
	`sdr-id 2`
	`IP Address of VM 192.0.0.6`
	`MAC address of VM E2:3A:DD:0A:8C:06`
	`VM State RUNNING`
	`start-time`
	`2015-05-07T18:22:44.136498+00:00`
	`Last Reload Reason FIRST_BOOT`
	`Reboot Count 1`
	You must note the data plane location. In this example status of the data plane is shown under location 0/RP0/VM2.

Start the data plane	sdr default-sdr location <data-plane-location> start
	Example
	`sysadmin-vm:0_RP0# sdr default-sdr location 0/RP0/VM2 start`
	`Mon May 4 17:16:37.867 UTC`
	`Start ? [no,yes] yes`
	`result start sdr default-sdr location 0/RP0 request acknowledged`

Shut the data plane down	sdr default-sdr location <data-plane-location> shut
	Example
	`sysadmin-vm:0_RP0# sdr default-sdr location 0/RP0/VM2 shut`
	`Mon May 4 17:12:32.397 UTC`
	`Shut ? [no,yes] yes`
	`result shutdown sdr default-sdr location 0/RP0 request acknowledged`

Task	Use this command

Reload the data plane	sdr default-sdr location <data-plane-location> reload
	Example
	`sysadmin-vm:0_RP0# sdr default-sdr location 0/RP0/VM2 reload`
	`Mon May 4 17:21:17.390 UTC`
	`Reload ? [no,yes] yes`
	`result graceful reload sdr default-sdr location 0/RP0 request acknowledged`

Note In the normal course of operations users must not manually start and stop the data plane manually.

Data Plane Debugging

Cisco IOS XRv 9000 Router provides a set of commands to check the status and statistics of the data plane, they are:

Note These data plane debug commands may briefly interrupt the traffic forwarding:

- show controller dpa statistics
- show controller dpa fib ipv4|ipv6 [<prefix> | summary]

.

- **show controller dpa version**—displays the version of the data plane.

  For example:

```
RP/0/RP0/CPU0:ios#show controller dpa version
Fri May 29 19:28:16.520 UTC
Image built on 13:29:13 May 29 2015 in workspace /workspace1/shope/ttl_commit
DPA started May 29 18:11:23,  up 0 days, 01:16
```

- **show controller dpa logging**—displays the data plane log. By default only errors and important events are log information is available.

  For example:

```
RP/0/RP0/CPU0:SS_Node1#show controllers dpa logging
 Mon Jun 29 19:47:33.245 UTC
 Jun 29 01:43:32.820: Log File Started
 Jun 29 01:43:32.820: DPA_INFO: DPA beginning initialization
 Jun 29 01:43:32.823: DPA_INFO: Dataplane Agent enabled
 Jun 29 01:43:32.823: DPA_INFO: Image built on 15:02:53 Jun 25 2015
 Jun 29 01:43:32.823: DPA_INFO: Table WRED_STR of size 8388480 is being initialized
 Jun 29 01:43:32.824: DPA_INFO: Table STATIC_POLICER_STR of size 8192 is being initialized
 with data
 Jun 29 01:43:32.824: DPA_INFO: Table HASH_DYN_BUCKET_STR of size 3355264 is being
initialized
```

```
 Jun 29 01:43:32.825: DPA_INFO: Table HASH_BUCKET_STR of size 33552832 is being
initialized
 Jun 29 01:43:32.829: DPA_INFO: Table DYN_FREE_BLOCK_STR of size 16777216 is being
initialized
 Jun 29 01:43:32.832: DPA_INFO: Table INDEX_Q_STR of size 8192 is being initialized
 Jun 29 01:43:32.832: DPA_INFO: Table HASH_HOST_DYN_BUCKET_STR of size 1677504 is being
 initialized
```

- **show controller dpa statistics global**—displays the data plane statistics that includes drop packet counters, packet injected from the control plane and packet punted to the control plane.

  For example:

```
RP/0/RP0/CPU0:ios#show control dpa statistics global
Fri May 29 19:27:35.497 UTC
Index  Punt                                                                    Count
------------------------------------------------------------------------------------
 1575  ARP                                                                        28
 1677  IFIB                                                                     1341
 1701  IPv4 FIB                                                                    5

Index  Inject                                                                  Count
------------------------------------------------------------------------------------
  267  IPv4 from fabric                                                           31
  268  IPv4 from fabric multicast                                               1290
  270  IPv4 from fabric next-hop                                                  55
  275  Inject to fabric                                                         1376
  276  Inject to port                                                             11

Index  Drop                                                                    Count
------------------------------------------------------------------------------------
   63  Egress uIDB in down state                                                   1
  113  IPv6 disabled in uIDB                                                      52
```

- **show controller dpa fib ipv4|ipv6 [<prefix> | summary]**—displays FIB entries on the data plane. The <prefix> and summary keyword are optional.

  For example:

```
RP/0/RP0/CPU0:ios#show controller dpa fib ipv4
Fri May 29 19:54:40.110 UTC

VRF id: 0
Default prefix 0.0.0.0/0 -> leaf:46423
total number of prefix:35
total_node_allocated:29 leaf_inserts:50 leaf_deletes 15 leaf_replaces: 2

Prefix               leaf_index
224.0.0.0/4          46436(0xb564)
224.0.0.0/24         46434(0xb562)
255.255.255.255/32   46429(0xb55d)
0.0.0.0/32           46433(0xb561)
10.1.1.1/32           46513(0xb5b1)
2.2.2.2/32           46510(0xb5ae)
```

  If the summary keyword is used the command shows the total number of prefix in each vrf table and operational statistics. For example:

```
RP/0/RP0/CPU0:R1-PE1#show control dpa fib ipv4 summary

VRF id: 0
Default prefix 0.0.0.0/0 -> leaf:46421
```

```
total number of prefix: 27859
allocated nodes:        1089
leaf_inserts:           27922
leaf_deletes            63
leaf_replaces:          173


VRF id: 1
Default prefix 0.0.0.0/0 -> leaf:46444
total number of prefix: 430
allocated nodes:        26
leaf_inserts:           430
leaf_deletes            0
leaf_replaces:          213
```

If the <prefix> keyword is used the command displays list of all prefix and vrf tables that match the prefix. For example:

```
RP/0/RP0/CPU0:ios#show controller dpa fib ipv4 5.11.23.131/32
VRF id: 0
Prefix              leaf_index
5.11.23.131/32      1164818(0x11c612)

VRF id: 1

VRF id: 2

VRF id: 3
```

- **show controller dpa tm queue <num>**—displays the internal data of a traffic manager queue, including DRR weight, Q-limit, instantaneous packet and byte counts.

  For example:

```
RP/0/RP0/CPU0:ios#show controller dpa tm queue 1
Fri May 29 19:31:25.556 UTC
Queue 1
  Parent Subport:       0
  Weight:               10
  Q-Limit:              625000
  Packets:              0
  Bytes:                0
```

- **show controller dpa tm subport <num>**—displays the internal data of a traffic manager subport, including DRR weight, shaped rate, queue config,and instantaneous packet.

  For example:

```
RP/0/RP0/CPU0:ios#show controller dpa tm subport 3
Fri May 29 19:44:12.993 UTC
Subport 3
  Parent vPort:       3
  Weight:             10200
  Rate:               776726
  Being Deleted:      no
  Configured:         yes
    Queue     24 pkts:        0 bytes:          0
    Queue     25 pkts:        6 bytes:       8376
    Queue     26 pkts:        0 bytes:          0
    Queue     27 pkts:        0 bytes:          0
    Queue     28 pkts:        0 bytes:          0
    Queue     29 pkts:        0 bytes:          0
```

```
  Queue     30 pkts:        0 bytes:          0
  Queue     31 pkts:        0 bytes:          0
Priority Queues:    1
Best effort Queues: 7
```

- **show controller dpa tm vport <num>**—displays the internal data a traffic manager vport, including rate and whether or not flow control is currently active on that port.

  For example:

```
RP/0/RP0/CPU0:ios#show controller dpa tm vport 0
Fri May 29 19:32:39.447 UTC
vPort 0
  Parent port:        0
  Rate:               95
  Flow control:       0
```

Set interface bandwidth to match SFP speed

To ensure the interface is reflecting the SFP bandwidth, you can configure its interface bandwidth same as SFP bandwidth.

SFP modules are available in multiple speed variants, such as 1 Gbps, 10 Gbps, 25 Gbps, etc. To macth the SFP bandwidth with interface bandwidth, you can use this configuration.

Note The interface name typically indicates the physical port's default rate, not the actual speed of the inserted SFP module.

Before you begin

- Verify which SFP module is installed in the interface.
- Check the speed supported by your SFP module.

Procedure

Step 1 Access the device configuration interface.

Step 2 Identify the interface containing the SFP module.

Step 3 Enter interface configuration mode.

Example:

```
Router#configure terminal
Router(config)#interface TF0/0/0/2
```

Step 4 Set the bandwidth to match the SFP speed.

Example:

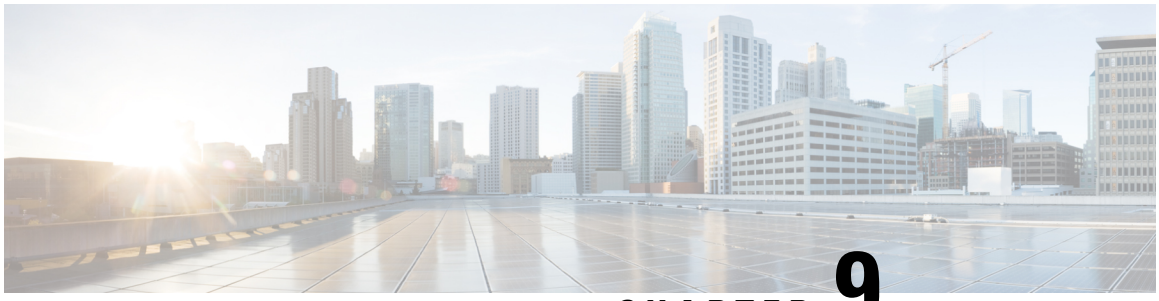
```
Router(config-if)#bandwidth 10000000
```

Step 5 Verify that the interface bandwidth is set correctly.

Example:

```
Router#show interface breif details GigabitEthernet0/1

               Intf       Intf        LineP              Encap  MTU        BW
               Name       State       State               Type (byte)    (Kbps)
--------------------------------------------------------------------------------
                Nu0          up          up               Null  1500          0
     Mg0/RP0/CPU0/0          up          up               ARPA  1514          0
          Hu0/0/0/0  admin-down  admin-down               ARPA  1514  100000000
          Hu0/0/0/1  admin-down  admin-down               ARPA  1514  100000000
          TF0/0/0/2  admin-down  admin-down               ARPA  1514   10000000
          TF0/0/0/3  admin-down  admin-down               ARPA  1514   25000000
```

Confirm the bandwidth matches your SFP module speed.



CHAPTER 9

Cisco IOS XRv 9000 Appliance

This chapter introduces Cisco IOS XRv 9000 as an Appliance and describes the concepts associated with Appliance. This chapter also talks about tasks required to upgrade, downgrade and reinstall the IOS XRv 9000 software on Appliance.

Note Cisco IOS XRv 9000 Appliance is introduced in Cisco IOS XR Release 6.1.2

- Introducing Cisco IOS XRv 9000 Appliance, on page 117
- Appliance Physical Connections Overview, on page 119
- Configuring the Appliance, on page 122
- Software Management, on page 124
- Firmware Management , on page 130
- Cisco IOS XRv 9000 Appliance Hardware Monitoring, on page 140

Introducing Cisco IOS XRv 9000 Appliance

Cisco IOS XRv 9000 Appliance is a package of UCS hardware and Cisco XRv 9000 Router software with all applicable licenses. The Appliance package enables you to virtualize your network routing function without having operational concerns about ownership of hardware and software.

Cisco IOS XRv 9000 Appliance is the pre-installed Cisco IOS XRv 9000 Router software that is sent from the factory on a bare metal UCS server hardware. It supports hyper scalability as it can scale to 70 Million route prefixes when run as a Virtual Route Reflector. Therefore, the extra layer of software (hypervisor) is not required.

The Appliance also supports Zero Touch Provisioning (ZTP) which allows easier insertion into existing networks.

A single PID for the Appliance is inclusive of hardware, software, licenses, and services. The single PID for the Appliance simplifies the support and service experience as it eliminates the need to have separate service contract for software and hardware.

Note

- Licensing is disabled.
- Adding and removing any hardware is not supported.

The below table lists supported UCS server and Appliance PID:

Table 17: Supported UCS Server and Appliance PIDs

Cisco IOS XR Release	Supported UCS Server Model	PIDs for Appliance
Release 6.1.2 and until Release 7.2.2	UCS C220 M4S	ASR-XRV9000-APLN
Release 6.6.2 to Release 24.3.x	UCS C220 M5SX (UCSC-C220-M5SX)	XRV9000-APLN-ROUT
Release 24.4.2	UCS C220 M7	XRV-M7-APLN-25G XRV-M7-APLN-100G

Default Console Settings for UCS C220 M7 Appliance

Table 18: Required Default Console Settings for UCS C220 M7 Appliance

Parameter	Default Setting
Boot Mode	EFI
Serial-over LAN	Disabled
Serial-over LAN Baud Rate	115200
Serial-over LAN COM Port	COM 0
CIMC	• Uses UCS default password. • Password policy is enabled and one-time password is enforced. • Network mode is shared ROM. • Network redundancy is active-active.
BIOS	• UCS is default for others. • Boot-order sequence is CDROM and HDD. • Console redirection is disabled. • UEFI secure boot is enabled. • Hyperthreading is disabled.

Appliance Physical Connections Overview

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
UCS C220 M7 based Appliance Support on XRv9000	Release 25.4.1	The Cisco UCS C220 M7 Rack Server appliance now supports 5th Gen Intel Xeon Scalable CPU with 2.2GHz clock speed, 28 cores and 128GB of RAM with a 4800MT/s transfer rate.
UCS C220 M7 based Appliance Support on XRv9000	Release 24.4.2	The Cisco UCS C220 M7 Rack Server is designed for high-density performance, catering to virtualization, collaboration, and bare metal applications. The M7 based appliance features a 4th Gen Intel Xeon Scalable CPU with 28 cores and 128GB of RAM. Key enhancements include PCIe Gen 5.0, a DDR5 memory bus, and the ability to utilize 480GB M.2 Bootable SATA Intel SSD drives. Furthermore, we now provide M7 appliance support for both 25G and 100G models, effectively succeeding the end-of-life M5 UCS Server.

The rare panel view of the Appliance is similar to UCS server. However, some interfaces that are available on the UCS server are not used in the Appliance. The below topics show the usage and mapping of the interfaces in Appliance.

Note M5 is not supported starting from Release 24.4.2 onwards.

UCS M7 Based Appliance Chassis Rear View (two full-height, 3/4-length PCIe risers)

This figure shows an overview of UCS 25G M7 based Appliance rear view panel features for two full-height, ¾ length PCIe riser.

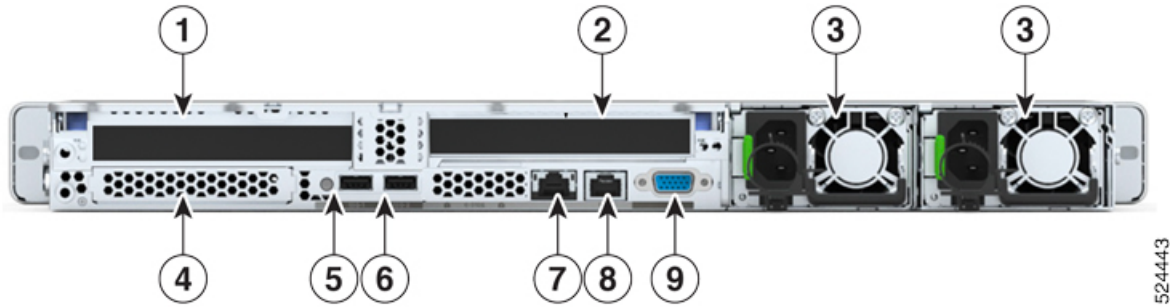


Table 20: Mapping of the Interfaces in Appliance

	Interface Description	In Appliance
1.	Riser 1C PCIe Gen5 (CPU1 control) • Supports one PCIe slot (slot 1) • Slot 1 is full-height, 3/4 length, x16,NCSI, Single Wide GPU	Contains Intel E810 NIC with 4x25G ports
2.	Riser 3C PCIe Gen5 x16 (CPU2 control) • Supports one PCIe slot (slot 3) • Slot 3 is Full-height, 3/4 length, x16, NCSI, Single Wide GPU	-
3.	Power supplies (two, redundant as 1+1)	-
4.	Modular LAN on motherboard (mLOM)/OCP 3.0 slot	Used to connect Intel X710 OCP card for management connectivity.
5.	System ID pushbutton/LED	Mapped to Cisco Integrated Management Controller (CIMC)
6.	USB 3.0 ports (two)	Used as USB ports.
7.	1GBE dedicated Ethernet management port	Mapped to Cisco Integrated Management Controller (CIMC)
8.	COM port (RJ45 connector)	Mapped to XR console The serial port should be cabled to a device that will allow you keyboard/video access over that serial port.
9.	VGA display port (DB15 connector)	Used as VGA display port

Interface Enumeration and Physical Mapping

3	2	1	0
PCIe01 4X25G E810			

UCS M7 Based Appliance Chassis Rear View (three half-height, 3/4 length PCIe riser)

This figure shows the rear view of UCS 100G M7 based Appliance panel features for three half-height, ¾ length PCIe riser.

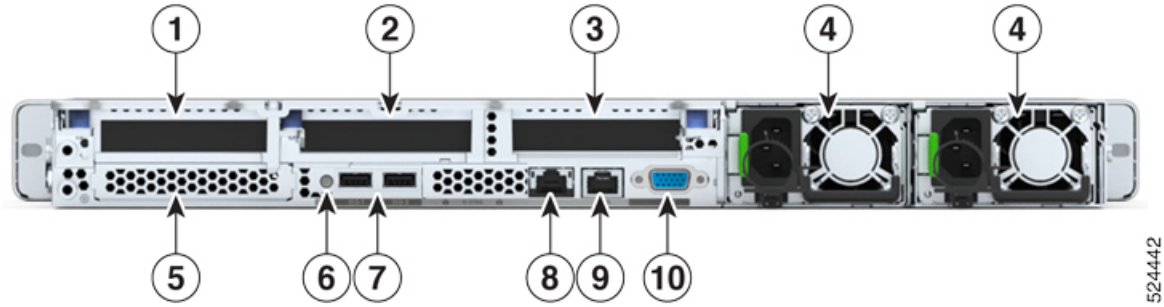


Table 21: Mapping of the Interfaces in Appliance

	Interface Description	In Appliance
1.	There are two half height riser options: • Riser 1A PCIe Gen4 (CPU1control) • Supports one PCIe slot (slot 1) • Slot 1 is half-height, 3/4 length, x16, NCSI, Single Wide GPU	Contains Mellanox ConnectX-6 NIC with 2x100G ports.
2.	There are two half height riser 2 options: • Riser 2A PCIe Gen4 x16 (CPU1 control) - • Supports one PCIe slot (slot 2) • Slot 2 is half-height, 3/4 length, x16, Single Wide GPU	Contains Mellanox ConnectX-6 NIC with 2x100G ports.
3.	There is one half height riser 3 option: Riser 3A PCIe Gen4 x16 (CPU2control) • Supports one PCIe slot (slot 3) • Slot 3 is half-height, 3/4 length, x16, NCSI, Single Wide GPU	-

	Interface Description	In Appliance
4.	Power supplies (two, redundant as 1+1)	-
5.	Modular LAN on motherboard (mLOM)/OCP 3.0 slot	Used to connect Intel X710 OCP card for management connectivity.
6.	System ID pushbutton/LED	Mapped to Cisco Integrated Management Controller (CIMC)
7.	USB 3.0 ports (two)	Used as USB ports.
8.	1GBE dedicated Ethernet management port	Mapped to Cisco Integrated Management Controller (CIMC)
9.	COM port (RJ45 connector)	Mapped to XR console The serial port should be cabled to a device that will allow you keyboard/video access over that serial port.
10.	VGA display port (DB15 connector)	Used as VGA display port

Interface Enumeration and Physical Mapping

The PCIe02 adapter is physically inserted upside down relative to PCIe01. Therefore the last two ports of PCIe02 interface are upside down. Hence the physical to XR port mapping is from left to right as shown in the below table:

1	**0**	**3**	**2**
PCIe01 2X100G MLNX		PCIe02 2X100G MLNX	

Configuring the Appliance

The Appliance can be configured in three ways:

Manual Configuration Using CLIs

To start the manual configuration:

1. Connect to the XR console (or a controller) through the serial port (COM0) which connects directly to CIMC.

   Alternately, you can also access console output from CPU through CIMC, SOL, or KVM using a console redirection setting, which can be modified in BIOS. The same can be accessed at following path from the CIMC GUI: **Compute > BIOS > Configure BIOS > Server Management**.

   Once this setting is modified, reboot the host so that the new settings are applied. After the host reboot, all the output from BIOS, CPU is redirected to KVM or SOL (Serial-over LAN).

2. Login to the XR console using admin password

Note You can use default credentials username as <*root*> and password as <*lab*>. If you still cannot login, use <*root*> as the user name <*Cisco123*> as password.

3. Configure the router manually using CLIs

   For information on specific IOS XR configuration refer, ASR 9000 System Management Configuration Guide.

   For information on specific IOS XR configuration CLIs refer, ASR 9000 System Management Command Reference.

IOS XRv 9000 does not support all the features supported on IOS XR. Refer the latest IOS XRv 9000 Router Release Notes to know the features supported on IOS XRv 9000 Router.

Automated Configuration Through Zero Touch Provisioning

Zero Touch Provisioning (ZTP) helps in auto provisioning after the software installation of the router.

ZTP boot modes

ZTP can be used to configure the router as a part of first boot in one of the following modes:

- Appliance Mode
- KVM mode on RHEL
- ESXI mode on VMware

ZTP auto provisioning methods

ZTP on first boot from the factory can be initiated either using configuration or script files.

- **Configuration**—Downloads and executes the configuration files. The first line of the file must contain the code *!! IOS XR* for ZTP to process the file as a configuration.
- **Script**—Downloads and executes the script files. These script files include a programmatic approach to complete a task. For example, scripts created using IOS XR commands to perform patch upgrades. The first line of the file must contain the code *#! /bin/bash* or *#! /bin/sh* for ZTP to process the file as a script.

For ZTP to work, a DHCP server and a TFTP or HTTP-based file server containing Day-0 configuration should be present on the network interface being used for configuration.There should be a configuration on DHCP server for specific UID or MAC and path of a valid network file containing the Day-0 configuration.

ZTP priority on selection of interface

ZTP can be run on both management or data plane interfaces. Selection of an interface depends upon which interface is found as ACTIVE, however the selection is based on the order as specified.

1. Management interface
2. Data plane interface

For more information on auto provisioning using ZTP, refer Zero Touch Provisioning section.

Automated Configuration using CVAC and USB

The Cisco IOS XRv 9000 Appliance supports auto configuring using CVAC. You should supply a plain-text configuration file **iosxr_config.txt** that has standard XR configurations on a USB drive to CVAC and boot the Appliance. This only works if no other configuration, including initial username and password has been configured.

For more information on how to boot Appliance using CVAC, refer the section CVAC - Bootstrap Configuration Support, on page 68.

Software Management

As an IOS XR based product, the IOS XRv 9000 Appliance inherits a lot of the capabilities for software management from IOS XR. This section describes the concepts and tasks necessary to upgrade, downgrade and reinstall the IOS XRv 9000 Router software.

Note The FPD related commands are not supported on IOS XRv 9000 Appliance. That includes **fpd auto-update** command.

Software Management through UCS

The Appliance device comes with IOS XRv 9000 software preinstalled. User can reimage the device with the desired version of the software (Release 24.4.2 or later) anytime by one of these methods:

- Reinstall the OS using CIMC
- Reinstall the OS from the USB Port
- Reinstalling the OS using a PXE Installation Server

Note Reinstalling the OS will remove all the existing configurations and system information.

After the OS is installed perform the basic configuration as discussed in the *Configuring the Appliance* section.

Reinstall IOS XRv 9000 software using CIMC

Cisco Integrated Management Controller (CIMC) is used to manage the physical device and can be accessed through a web browser. CIMC is used to:

- remotely power on/off the Appliance,
- remotely access the console,
- reinstall the software, and
- upgrade firmware.

You can use CIMC to reinstall IOS XRv 9000 software remotely on the Appliance. By default CIMC has a dedicated GigE port on the Appliance. You must configure the CIMC port with an IP Address to access CIMC from a web browser. The option to configure CIMC port is available during power-on the device on the VGA console.

After configuring the IP Address on the CIMC port, log into CIMC from a web browser and use KVM (keyboard, video, and mouse) console.

When updating to Cisco IOS XRv version 24.4.2 or later on a UCS bare metal server, the KVM console is no longer available. Instead, you should use the telnet port, which allows access to both the XR and Calvados consoles.

Instead of using CD/DVD or floppy drives physically connected to the server, the KVM console uses virtual media, which are actual disk drives or disk image files that are mapped to virtual CD/DVD or floppy drives.

For more information on launching KVM console, refer KVM Console.

Note The ISO version of the IOS XRv 9000 software must be used for software installation and reinstallation.

Reinstalling the OS using CIMC on M7 UCS based Appliance

Follow the below procedure to reinstall the OS on M7 UCS based Appliance:

Starting Cisco IOS XR Software Release, Xrv9000 OE Images (24.4.2 and higher version images) are able to load only on UCS M7 machines in Baremetal appliance deployment scenario.

Before you begin

- Download the desired ISO image file (Release 24.4.2 or later version) to your machine
- You must log in as the user with admin privileges to install the OS
- You must be running CIMC latest version.
- Before loading the IOS XRv9k image, ensure that you have the required UCS M7 BIOS and CIMC images running on the system.

Procedure

Step 1 Download the required standard image (new K9 VRR image) from CCO and proceed with the installation.

Step 2 Copy OS installation ISO disk image files to your computer.

Step 3 If CIMC is not open, then log in.

Step 4 From the CIMC portal, launch KVM from **Actions** > **System** > **launch KVM**.

Step 5 Select Virtual Media>Activate Virtual Device in the KVM console.

Step 6 Select Virtual Media>vKVM Mapped DVD. Then browse the ISO installation disk image stored locally and click Map Device.

Step 7 Select Power>Power cycle System (warm boot) in the KVM console.

Step 8 Press F6 to enter the boot menu. Once you are in the boot menu, select the first DVD which is nothing but the same iso you selected earlier, mounted as a virtual disk. (Eg: Cisco vKVM Mapped vDVD1).

When the server reboots, it begins the installation process. After the installation process completes, refer the section *Configuring the Appliance* to go through how to configure the device. To retain the console access you can check if COM0 is mapped to SOL. So, you can either disable SOL or map SOL to COM1. After this, the physical Serial Port is map to the XR Console.

Reinstalling the OS from a USB Port

The Appliance supports booting an operating system from any USB port. However, there are a few guidelines that you must keep in mind, prior to booting an BIOS from a USB port.

- The BIOS installation process requires a bootable USB drive. Refer *Creating a Bootable USB Drive* section.
- To maintain the boot order configuration, it is recommended that you use an internal USB port for booting an BIOS.
- The USB port must be enabled prior to booting an OS from it.

Note By default, the USB ports are enabled. If you have disabled a USB port, you must enable it prior to booting an BIOS from it.

- After you boot the BIOS from the USB port, you must set the second-level boot order so that the server boots from that USB source every time.

Installing an operating system from a USB port:

1. Power cycle the Appliance
2. During boot process, select the USB Boot Option, and continue
3. The system will install the image from USB drive to harddisk drive and then reboots.

Note USB drive with large memory size won't boot. Hence, we recommend to use 8GB USB drive.

Reinstall the operating system on a UCS C-Series server using CIMC CLI

Enable a remote reinstallation of the OS on a Cisco UCS C-Series server without relying on physical media or PXE.

The Cisco Integrated Management Controller (CIMC) provides out-of-band management for UCS servers, including virtual media features. You can map an installer ISO via NFS or HTTP(S), enabling you to boot and reinstall the operating system remotely.

If you are hosting the installer ISO and virtual media mapping via HTTPS, skip the NFS setup steps and proceed to mapping the ISO in CIMC.

Before you begin

- Obtain an OS installer ISO file.

- Ensure that either an NFS server share or an HTTP/HTTPS server is set up and accessible from the CIMC network. The OS installer ISO file must be hosted and shared from the same server (NFS or HTTP/HTTPs). For details on setting up remote sharing, see *Ubuntu* or *RHEL remote sharing documentation.*
- Verify ISO checksum for integrity.

Procedure

Step 1 Connect to CIMC via SSH.

Example:

```
shell
 ssh <username>@<cimc_ip_address>
```

Enter your password as prompted.

Step 2 Scope to virtual media.

Example:

```
C220-WZXXXXXXX#scope vmedia
```

Note
Check and unmount any old vmedia mappings, by following steps 8-11.

Step 3 Map the ISO image:

- For NFS

  ```
  C220-WZXXXXXXX /vmedia#map-nfs <map_name> nfs://<nfs_ip>/<location> <iso_file>
  ```

- For HTTP

  ```
  C220-WZXXXXXXX /vmedia#map-www <map_name> http://<server_ip>/<location> <iso_file>
  ```

Step 4 Verify the mapping and exit the vmedia scope.

Example:

```
C220-WZXXXXXXX /vmedia#show mappings detail
C220-WZXXXXXXX /vmedia#exit
```

Step 5 Enter BIOS scope.

Example:

```
C220-WZXXXXXXX#scope bios
```

a) Enable all boot devices.

Example:

```
C220-WZXXXXXXX /bios#enable-all-boot-device
```

When prompted, confirm: `Continue? [y|N]` Enter `y`.

Step 6 Set boot order to prioritize CDROM.

Example:

```
C220-WZXXXXXXX /bios#set boot-order CDROM,HDD,FDD,PXE
```

Reboot server for changes to take effect. Do not disable boot options via BIOS screens. Devices not detected will be removed from boot order.

Step 7 Commit the configuration and reboot the system.

Example:

```
C220-WZXXXXXXX /bios *# commit
```

When prompted, `Do you want to reboot the system? [y|N]`. Enter `y`.The server reloads, and boots from the mounted ISO

bios * indicates the uncommiteed changes are present. Once the changes are committed, the asterisk sign (*) will disappear.

Note
After mapping the ISO and initiating the server reload, the system will boot from the CD-ROM as prioritized in the boot order. The operating system installation begins, and installation logs will be visible on the Host console. On completion, the boot order automatically switches to the installed disk ("CiscoXR"). The server then reboots and loads from the disk. Once the Host successfully boots, XR and admin router consoles become available.

Step 8 Once the host boots up fine, it is recommended to unmap the vmedia entries. Connect to CIMC, scope to virtual media.

Example:

```
C220-WZXXXXXXX#scope vmedia
```

Step 9 List mapped devices.

Example:

```
C220-WZXXXXXXX /vmedia#show mappings
```

Step 10 Unmap volumes.

Example:

```
C220-WZXXXXXXX /vmedia#unmap <volume_name>
```

Confirm: Enter `yes`.

Step 11 Delete saved mappings.

Example:

```
C220-WZXXXXXXX /vmedia # delete-saved-mappings
```

Confirm: Enter `yes`.

The operating system is reinstalled, and the server boots from the new iso loaded into disk via virtual media.

Reinstalling an OS Using a PXE Installation Server

Before you begin

- Verify that the server can be reached over a VLAN.
- You must log in as a user with admin privileges to install an OS.

Procedure

Step 1 Set the boot order to PXE first.

Step 2 Reboot the server.

If a PXE install server is available on the VLAN, the installation process begins when the server reboots. PXE installations are typically automated and require no additional user input. Refer to the installation guide for the OS being installed to guide you through the rest of the installation process.

What to do next

After the OS installation is complete, reset the LAN boot order to its original setting.

Creating a Bootable USB drive

To create a bootable USB drive you need UNetbootin-an external open source software.

Before you begin

- Download the desired Cisco IOS XRv 9000 ISO installation file to your laptop or server.
- Download the UNetbootin app from this link: https://unetbootin.github.io/

Procedure

Step 1 Copy the OS installation disk image files to your computer

Note
We recommend to use ISO version of the IOS XRv 9000 software installation file for reinstallation.

Step 2 Format the USB disk to fat32 format

Step 3 Run UNetbootin and load the ISO installation file

Step 4 Build the USB disk. Follow the instructions provided in this link: https://unetbootin.github.io/

By default, the BIOS displays the list of items that the user has to select.

For Mac OS users, use terminal to go to the mount point, and use Vi editor to edit the file. For example: */Volumes/MYDISK*.

Software Management using IOS XR

The IOS XRv 9000 software can be upgraded or downgraded using any of these methods:

- IOS XR CLI commands
- ZTP bash scripting (of install commands)

- IOS XR supported manageability interfaces

For information on the upgrade and downgrade procedures, see the upgrade document. It is available along with the software images.

Software Upgrade Using CLI

Before you begin

- Download the desired ISO image file to your machine.

Procedure

Step 1 install commit

Example:

```
router# install commit
```

Commit the current version of IOS XRv 9000 software installed on the Appliance.

Step 2 install add source <*filepath*>

Example:

```
router# install add source tftp://192.0.2.4/fakepath/xrv9k-fullk9-x.iso
```

Locate the ISO disk image file that has to be installed on the Appliance.

Step 3 install activate <*filename*>

Example:

```
router# install activate xrv9k-fullk9-x.iso
```

Activates the IOS XRv 9000 new image version. The router reboots.

Step 4 show version

Example:

```
router# show version
```

Verify the new image version installed.

Step 5 install commit

Example:

```
router# install commit
```

Commits the new version.

Firmware Management

Firmware for the following individual components can be updated:

- CIMC
- BIOS

Update the BIOS and Cisco IMC Firmware

Caution When you upgrade the BIOS firmware, you must also upgrade the Cisco IMC firmware to the same version, or the server does not boot. Do not power off the server until the BIOS and Cisco IMC firmware are matching or the server does not boot.

UCS Server can be upgraded only to firmware obtained and certified by Cisco. Cisco provides release notes with each firmware image.

- You can upgrade the Cisco IMC and BIOS firmware by using the Cisco IMC GUI interface.

Firmware Files

The latest version of firmware validated with Release 25.4.1 is bundled as part of following package:

```
xrv9k-ucs-c220m7-huu-container-6.0.1.250127.tar.gz
```

The bundle contains latest CIMC and BIOS firmware which needs to be upgraded on the system as described below. There is a mandatory security fix in CIMC image and hence would require the system to be upgrade with packaged CIMC and BIOS versions.

The CIMC and BIOS firmware can be extracted after downloading the TAR ball from the CCO site, as mentioned in the following steps.

1. Copy the tar ball to a laptop connected on the same network as UCS management network.
2. Untar the package.
3. In the untarred directory, check for **bios.pkg** and **cimc.bin** in following directory:

   ```
   <untar dir>/firmware/bios/bios.pkg
   <untar dir>/firmware/cimc/cimc.bin
   ```

4. Confirm that the checksum for both the files using **sha256sum** command matches as per the hashes specified in the respective release notes.

   Use the firmware package **cimc.bin** for upgrading CIMC version and **bios.pkg** for BIOS as provided in the following procedure.

Upgrading via the CIMC UI

Viewing Firmware Components

Procedure

Step 1 From the **Apps** drop-down list, select **Administration**.

Step 2 In the **Navigation** pane, click **Firmware Management** menu.

Step 3 In the **Firmware Management** area, review the following information for CIMC, BIOS, and other components:

Table 22: Firmware Management

Name	Description
Component checkbox	Allows you to select one ore more components.
Upload button	Opens a dialog box that allows you to install a firmware image file that is available to your local machine or on a remote server. **Note** **Upload** button is visible only when you select a component.
Activate button	Allows you to activate the firmware for the selected component. **Note** **Activate** button is visible only when you select a component.
Component column	List of components available for which you can update the firmware.
Running Version column	The firmware version of the component that is currently active.
Backup Version column	The alternate firmware version installed on the server, if any. The backup version is not currently running. To activate it, click **Activate**. **Note** When you install new firmware, any existing backup version is deleted and the new firmware becomes the backup version. You must manually activate the new firmware if you want the server to run the new version.
Bootloader Version column	The bootloader version associated with the boot-loader software of the component.
Status column	The status of the firmware activation on this server.

Name	Description
Progress in % column	The progress of the operation, in percentage.

Updating the Firmware

You can install the firmware package from a local disk, remote server, or a USB storage device connected to a server. After you confirm the installation, BMC replaces the firmware version in the component's backup memory slot with the selected version.

Procedure

Step 1 From the **Apps** drop-down list, select **Administration**.

Step 2 In the **Navigation** pane, click **Firmware Management** menu.

Step 3 In the **Firmware Management** area, select a component from the **Component** column and click **Upload**.

The **Update Firmware** dialog box appears.

Step 4 In the **Update Firmware** dialog box, update the following properties:

Table 23: Remote

Name	Description
Remote button	Select this option when you want to upload the firmware from a remote location.
Server IP/Hostname	Server details from where you want to download the client certificate.

Name	Description
Upload Protocol button	Select one of the following protocols. • TFTP • FTP • SCP • SFTP • HTTP **Note** If you select FTP, SCP or SFTP, you will be prompted to enter your username and password. • **Note** **Front Panel USB** option is visible only if **Smart Access USB** is enabled and a USB storage device is connected to the server. **Front Panel USB**—This allows you to install the firmware through a USB storage device connected to the server. Enter the firmware file name and location in the **Image Path and Filename** field, and click **Install Firmware**.
Path and Filename field	File path where the firmware file resides on the server along with the filename.
Username field	User name for your remote server.
Password field	Password for your remote server.
Upload button	Allows you to upload the firmware file. **Note** If you chose SCP or SFTP as the remote server type while performing this action, a pop-up window is displayed with the message *Server (RSA) key fingerprint is <server_finger_print _ID> Do you wish to continue?*. Click Yes or No depending on the authenticity of the server fingerprint. The fingerprint is based on the host's public key and helps you to identify or verify the host you are connecting to.

Table 24: Local

Name	Description
Local button	Select this option when you want to updated the firmware from your local machine. Click **Browse** and navigate to the firmware file you wish to use.

Name	Description
Upload button	Allows you to upload the firmare file.

Update CIMC firmware using CLI

Update CIMC firmware directly from the CLI when GUI access is not available.

Before you begin

- Ensure you have SSH access to the CIMC appliance.
- Obtain the relevant firmware (.bin for CIMC,) files and the IP address where they are hosted.
- Confirm network connectivity to the firmware file source.

Procedure

Step 1 Log in to CIMC via SSH.

Step 2 Check the current CIMC firmware version.

Example:

```
C220-WZXXXXX#scope cimc
C220-WZXXXXX /cimc#show firmware
Update Stage Update Progress Current FW Version
------------ --------------- ------------------
NONE         0               4.3(5.250021)
```

Step 3 Start the firmware update using your protocol of choice. For example, to use HTTP, enter:

Example:

```
C220-WZXXXXX /cimc/firmware#update http 192.0.2.1 /cimc.bin
Firmware update initialized.
Please check the status using "show detail".
```

The other protocols that you can use are TFTP, FTP, SCP, and SFTP.

The firmware update initializes.

Step 4 Monitor progress periodically, use the **show detail** command.

Example:

```
C220-WZXXXXX /cimc/firmware#show detail
Firmware Image Information:
    Update Stage: DOWNLOAD
    Update Progress: 5
    Current FW Version: 4.3(5.250021)
    FW Image 1 Version: 4.3(5.250021)
    FW Image 1 State: BACKUP INACTIVATED
    FW Image 2 Version: 4.3(5.250021)
    FW Image 2 State: RUNNING ACTIVATED
    Boot-loader Version: 4.3(5.250021)
```

```
    Secure Boot: DISABLED
C220-WZXXXXX /cimc/firmware#show detail
Firmware Image Information:
    Update Stage: INSTALL
    Update Progress: 45
    Current FW Version: 4.3(5.250021)
    FW Image 1 Version: 4.3(5.250021)
    FW Image 1 State: BACKUP INACTIVATED
    FW Image 2 Version: 4.3(5.250021)
    FW Image 2 State: RUNNING ACTIVATED
    Boot-loader Version: 4.3(5.250021)
    Secure Boot: DISABLED
C220-WZXXXXX /cimc/firmware#show detail
Firmware Image Information:
    Update Stage: NONE
    Update Progress: 100
    Current FW Version: 4.3(5.250021)
    FW Image 1 Version: 4.3(5.250021)
    FW Image 1 State: BACKUP INACTIVATED
    FW Image 2 Version: 4.3(5.250021)
    FW Image 2 State: RUNNING ACTIVATED
    Boot-loader Version: 4.3(5.250021)
    Secure Boot: DISABLED
```

This displays status including update stage, progress, image versions, and secure boot status.

Step 5 Once the update progress reaches 100%, activate the new firmware version.

Example:

```
C220-WZP2812997P /cimc/firmware#activate
This operation will activate firmware 1 and reboot the BMC.
Continue?[y|N]y
```

Confirm when prompted by entering `y`. This reboots the BMC and disconnects you.

Step 6 Reconnect via SSH after reboot to confirm the new firmware version is active.

Example:

```
C220-WZXXXXX /cimc/firmware#show detail
Firmware Image Information:
    Update Stage: NONE
    Update Progress: 0
    Current FW Version: 4.3(5.250021)
    FW Image 1 Version: 4.3(5.250001)
    FW Image 1 State: RUNNING ACTIVATED
    FW Image 2 Version: 4.3(5.250021)
    FW Image 2 State: BACKUP INACTIVATED
    Boot-loader Version: 4.3(5.250001)
    Secure Boot: DISABLED
```

Confirm the new firmware version is shown as RUNNING ACTIVATED.

Step 7 Exit firmware scope, CIMC scope and power cycle the chassis.

Example:

```
C220-WZXXXXX /cimc/firmware#exit
C220-WZXXXXX /cimc#exit
C220-WZXXXXX#scope chassis
C220-WZXXXXX /chassis#power cycle
Do you want to continue?[y|N]
```

Confirm when prompted by entering `y`. This reboots the chasssis and disconnects you.

Update BIOS using CLI

Update BIOS firmware directly from the CLI when GUI access is not available.

Before you begin

- Ensure you have administrator access to the Xrv9k router's CIMC via SSH.
- Obtain the BIOS firmware binary file (e.g., `bios.pkg`).
- The update server (HTTP) is running and accessible.

Procedure

Step 1 Log in to CIMC via SSH and enter BIOS scope:

Example:

```
C220-WZXXXXXXX#scope bios
```

Step 2 Check the current BIOS version.

Example:

```
C220-WZXXXXXXX/bios#show detail
BIOS:
    BIOS Version: C220M7.4.3.5b.0_XRV9K
    Backup BIOS Version: C220M7.4.3.5b.0_XRV9K
    Boot Order: (none)
    FW Update Status: Done, OK
    UEFI Secure Boot: enabled
    Actual Boot Mode: Uefi
    Last Configured Boot Order Source: BIOS
    One time boot device: (none)
C220-WZXXXXXXX /bios #
```

Step 3 Upload the BIOS update file, for HTTP protocol:

Example:

```
C220-WZXXXXXXX /bios # update http://192.0.0.2 /bios.pkg
```

Confirm signature verification and that update has started.

The other protocols that you can use are TFTP, FTP, SCP, and SFTP.

Step 4 Monitor status.

Example:

```
C220-WZXXXXXXX /bios#show detail
BIOS:
    BIOS Version: C220M7.4.3.5b.0_XRV9K
    Backup BIOS Version: C220M7.4.3.5b.0_XRV9K
    Boot Order: CDROM,HDD
```

```
    FW Update Status: Image Download (0 %), OK
    UEFI Secure Boot: enabled
    Actual Boot Mode: Uefi
    Last Configured Boot Order Source: CIMC
    One time boot device: (none)
C220-WZXXXXXXX /bios#show detail
BIOS:
    BIOS Version: C220M7.4.3.5b.0_XRV9K
    Backup BIOS Version: C220M7.4.3.5b.0_XRV9K
    Boot Order: CDROM,HDD
    FW Update Status: Write Host Flash (50 %), OK
    UEFI Secure Boot: enabled
    Actual Boot Mode: Uefi
    Last Configured Boot Order Source: CIMC
    One time boot device: (none)
C220-WZXXXXXXX /bios#show detail
BIOS:
    BIOS Version: C220M7.4.3.5b.0_XRV9K
    Backup BIOS Version: C220M7.4.3.5b.0_XRV9K
    Boot Order: CDROM,HDD
    FW Update Status: Done, OK
    UEFI Secure Boot: enabled
    Actual Boot Mode: Uefi
    Last Configured Boot Order Source: CIMC
    One time boot device: (none)
```

Check FW Update Status and BIOS version progression.

Step 5 Activate the backup BIOS version after update completes.

Example:

```
C220-WZXXXXXXX /bios#activate
System is powered-on. This operation will activate backup BIOS version "C220M7.4.3.5b.0_XRV9K" during
 next boot.
Continue?[y|N]y
```

Step 6 After reconnecting, log back in and confirm the new BIOS version is active.

Example:

```
C220-WZXXXXXXX /bios#scope bios
C220-WZXXXXXXX /bios#show detail
BIOS:
    BIOS Version: C220M7.4.3.5b.0_XRV9K
    Backup BIOS Version: C220M7.4.3.5b.0_XRV9K
    Boot Order: CDROM,HDD
    FW Update Status: Done, OK
    UEFI Secure Boot: enabled
    Actual Boot Mode: Uefi
    Last Configured Boot Order Source: CIMC
    One time boot device: (none)
```

Step 7 Exit BIOS scope and power cycle the chassis.

Example:

```
C220-WZXXXXXXX /bios#exit
C220-WZXXXXX#scope chassis
C220-WZXXXXX /chassis#power cycle
Do you want to continue?[y|N] y
```

Confirm when prompted by entering `y`. This reboots the chasssis and disconnects you.

Activating the Firmware

Procedure

Step 1 From the **Apps** drop-down list, select **Administration**.

Step 2 In the **Navigation** pane, click **Firmware Management** menu.

Step 3 In the **Firmware Management** area, select a component from the **Component** column and click **Activate**.
The **Activate Firmware** dialog box appears.

Step 4 In the **Activate Firmware** dialog box, update the following properties:

Table 25: Activate Firmware

Name	Description
Firmware Version list	A list of the firmware versions available for activating the firmware.
Activate Firmware button	Activates the selected firmware version depending on the component you choose. **Important** While the activation is in progress, do not: • Reset, power off, or shut down the server • Reboot or reset BMC • Activate any other firmware • Export technical support or configuration data **Note** You can activate the BIOS firmware when the host is On. When you do this, **Activation Pending** is displayed in the **Status** field. After the next host boot, the BIOS activation that is in the pending state is activated.

Cancelling Firmware Activation

Before you begin

BIOS firmware must be in **Activation Pending** state for you to cancel the activation.

Procedure

Step 1 From the **Apps** drop-down list, select **Administration**.

Step 2 In the **Navigation** pane, click **Firmware Management** menu.

Step 3 In the **Firmware Management** area, select the firmware for which you want to cancel the activation.

Step 4 Click **Cancel Activation**.

Cancel Activation button is displayed only when you choose a firmware that is in **Activation Pending** state.

Cisco IOS XRv 9000 Appliance Hardware Monitoring

The hardware monitoring on Cisco IOS XRv 9000 Appliance enables you to view harware environmental parameters of the Appliance in the same way as you would see on a conventional hardware router. Based on the interfaces involved in retrieving information, the Appliance hardware information is grouped into these three sections:

- Hardware Environment Monitoring—This includes power supply unit, fan, voltage, current, and temperature information; also includes hardware failure warning and alarms information.
- Host OS Level Monitoring—This includes processor, core, memory, and HDD utilization information.
- SFP Optic Monitoring—This includes optical diagnostics and SFP OIR (online insertion and removal) monitoring information.

Hardware Environment Monitoring

In Cisco IOS XRv 9000 Appliance, the system continuosly monitor hardware to collect information on power consumption and report hardware failure. You can view these information using below commands in system admin mode.

Task	Use this command

To view the chassis fan information.	`sysadmin-vm:0_RP0#show environment fan`
	`Mon Mar 17 10:19:04.846 UTC+00:00`
	`Fan speed (rpm)`
	`Location FRU Type FAN_0 FAN_1 FAN_2 FAN_3 FAN_4 FAN_5 FAN_6 FAN_7`
	`0/FT0 XRV-FAN-C220M7= 5200 5330 5330 5330 5330 5330 5330 5330`
	There are six fans in the Cisco IOS XRv 9000 Appliance. These fans do not support OIR, hence you must shut down the Appliance in order to replace fans.
	Unlike other hardware platforms, the Cisco IOS XR software running on the Appliance does not manage the fan speed. Instead, they are controlled by UCS Cisco Integrated Management Controller (CIMC) system.

Task	Use this command

To view the power tray information.	`sysadmin-vm:0_RP0#show environment power`
	`Mon Mar 17 10:19:12.380 UTC+00:00`
	`CHASSIS LEVEL POWER INFO: 0`
	`Total output power capacity (-) : 0W + 0W`
	`Total output power required : 0W`
	`Total power input : 0W`
	`Total power output : 206W`
	`Power Shelf 0:`
	`Power Module / Supply Type / Input Volts / Input Amps / Output Volts / Output Amps / Status`
	`0/PT0-PM0 Cisco 0.0 0.0 12.1 9.0 OK`
	`0/PT0-PM1 Cisco 0.0 0.0 12.1 8.0 OK`
	`Total of Power Shelf 0: 0W/ 0.0A 206W/ 17.0A`
	`Location / Card Type / Power Allocated Watts / Power Used Watts / Status`
	`0/0 R-IOSXRV9000-LC-A 0 - -`
	`0/RP0 R-IOSXRV9000-RP-A 0 - -`
	`0/FT0 XRV-FAN-C220M7= 0 - -`
	In the above command output, only the highlighted field information (Power Module and Output) has a meaningful readings for the Cisco IOS XRv 9000 Appliance.
	The Total Power output is the sum of power output (power output = Volts*Amps) of each power module.
	There are no input Volt/Amp sensors in the Appliance and there are no capacity, required, allocated, and used power data available for the Appliance.

Task	Use this command

To view the temperature information.	`sysadmin-vm:0_RP0# show environment temperature`
	
	`Location TEMPERATURE Value Crit Major Minor Minor Major Crit Sensor (deg C) (Lo) (Lo) (Lo) (Hi) (Hi) (Hi)`
	
	`0/RP0`
	`Front (FP_TEMP_SENSOR) 27 -10 -5 0 40 45 50`
	`Hub (PCH_TEMP_SENS) 37 -10 -5 0 80 85 90`
	`Inlet (RISER1_INLET_TMP) 34 -10 -5 0 60 70 80`
	`Outlet (RISER1_OUTLETTMP) 34 -10 -5 0 60 70 80`
	`Inlet (RISER2_INLET_TMP) 35 -10 -5 0 60 70 80`
	`Outlet (RISER2_OUTLETTMP) 38 -10 -5 0 60 70 80`
	`Processor (P1_TEMP_SENS) 39 -10 -5 0 92 97 100`
	`Processor (P2_TEMP_SENS) 46 -10 -5 0 92 97 100`
	`Memory (DDR4_P1_A1_TEMP) 33 -10 -5 -1 65 85 90`
	`Memory (DDR4_P1_A2_TEMP) 0 -10 -5 -1 65 85 90`
	`Memory (DDR4_P1_A3_TEMP) 0 -10 -5 -1 65 85 90`
	`...`
	`0/PT0-PM0`
	`PM0-Supply (PSU_TEMP) 33 -10 -5 -1 60 65 70`
	`0/PT0-PM1`
	`PM1-Supply (PSU_TEMP) 28 -10 -5 -1 60 65 70`
	
	Note Few temperature readings for memory slots are zero because there are no DDR memory inserted in those memory slots.

Task	Use this command
To view the voltage information.	sysadmin-vm:0_RP0#**show environment voltages** Mon Mar 17 10:19:02.224 UTC+00:00 Location VOLTAGE Value Crit Minor Minor Crit Sensor (mV) (Lo) (Lo) (Hi) (Hi) 0/RP0 Board (PSU1_VOUT) 12100 10150 10788 14000 15000 Board (PSU2_VOUT) 12100 10150 10788 14000 15000 Board (P12V) 11890 10150 10788 13166 13804 Board (P3V_BAT_SCALED) 3088 2246 2543 3588 3760
To view hardware failure information	sysadmin-vm:0_RP0# **show logging | i envmon** Mon Oct 2 09:38:06.390 UTC 0/RP0/ADMIN0:Oct 1 16:58:44.394 : envmon[2332]: %PKT_INFRA-FM-6-FAULT_INFO : Power Module insertion :INFO :0/PT0-PM0: 0/RP0/ADMIN0:Oct 2 09:26:37.657 : envmon[2332]: %PKT_INFRA-FM-6-FAULT_INFO : Power Module insertion :INFO :0/PT0-PM1: **0/RP0/ADMIN0:Oct 2 09:37:03.605 : envmon[2332]: %PKT_INFRA-FM-6-FAULT_INFO : Power Module removal :INFO :0/PT0-PM1:** **0/RP0/ADMIN0:Oct 2 09:37:50.221 : envmon[2332]: %PKT_INFRA-FM-6-FAULT_INFO : Power Module insertion :INFO :0/PT0-PM1:** Before executing the above command, remove and re-insert power module (0/PT0-PM1). The power module supports OIR. In the above command output the highlighted row captures the power module removal and insertion information. **Note** The first two insertions in the above command output are from the system boot.

Task	Use this command

To view alarms	`sysadmin-vm:0_RP0# show alarms`
	
	`Thu Oct 19 12:28:59.400 UTC`
	
	`Active Alarms`
	
	`Location Severity Group Set time Description`
	
	`0/PT0-PM0 major environ 10/19/17 12:27:34 Power Module Error (PM_OUTPUT_STAGE_OT).`
	`0/PT0-PM0 major environ 10/19/17 12:27:34 Power Module Shutdown (PM_OC_SHUTDOWN).`
	`0/PT0-PM1 major environ 10/19/17 12:27:34 Power Module Fault (PM_VOUT_VOLT_OOR).`
	`0/RP0 major environ 10/19/17 12:27:34 Processor (P1_TEMP_SENS): temperature alarm.`
	`0/RP0 major environ 10/19/17 12:27:40 Board (P3V3_AUX): low voltage alarm.`

To view the current environmemt information.	`sysadmin-vm:0_RP0#show environment current`
	`Mon Mar 17 10:18:59.260 UTC+00:00`
	
	`Location CURRENT Sensor Value (mA)`
	
	`0/RP0`
	`Board (PSU1_IOUT) 9000`
	`Board (PSU2_IOUT) 8000`

Task	Use this command
To view consolidated information of temperature, voltage, power, fan, and current sensor values.	

Task	Use this command

```
sysadmin-vm:0_RP0#show environment all
Mon Mar  17 10:19:16.731 UTC+00:00

=============================================================================

   Location     Card Type            Power       Power
    Status
                                     Allocated   Used
                                     Watts       Watts
=============================================================================

   0/0          R-IOSXRV9000-LC-A       0           -
    -
=============================================================================
Location  TEMPERATURE                  Value   Crit Major
Minor Minor Major  Crit
          Sensor                     (deg C)   (Lo) (Lo)
(Lo)   (Hi)  (Hi)   (Hi)
-----------------------------------------------------------------------------
0/RP0
          Front (TEMP_SENS_FRONT)        19    -10    -5
    0    40    45    50
          Processor (P1_TEMP_SENS)       80    -10    -5
    0    92    97    98
          Memory (DDR5_P1_A1_TMP)        33    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_B1_TMP)        31    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_C1_TMP)        29    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_D1_TMP)        27    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_E1_TMP)        34    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_F1_TMP)        31    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_G1_TMP)        29    -10    -5
   -1    65    85    90
          Memory (DDR5_P1_H1_TMP)        27    -10    -5
   -1    65    85    90
=============================================================================
Location  VOLTAGE                      Value   Crit Minor
Minor  Crit
          Sensor                      (mV)     (Lo) (Lo)
(Hi)   (Hi)
-----------------------------------------------------------------------------
0/RP0
          Board (PSU1_VOUT)           12100  10150 10788
14000 15000
          Board (PSU2_VOUT)           12100  10150 10788
14000 15000
          Board (P12V)                11890  10150 10788
13166 13804
          Board (P3V_BAT_SCALED)       3088   2246  2543
 3588  3760
=============================================================================
Location  CURRENT                      Value
          Sensor                      (mA)
-----------------------------------------------------------------------------
0/RP0
        Board (PSU1_IOUT)            9000
```

Task	Use this command
	Board (PSU2_IOUT) 8000 Location Card Type Power Power Status Allocated Watts Used Watts 0/RP0 R-IOSXRV9000-RP-A 0 - - **Fan speed** (rpm) Location FRU Type FAN_0 FAN_1 FAN_2 FAN_3 FAN_4 FAN_5 FAN_6 FAN_7 0/FT0 XRV-FAN-C220M7= 5330 5330 5330 5330 5330 5330 5330 5330 Location Card Type Power Power Status Allocated Watts Used Watts 0/FT0 XRV-FAN-C220M7= 0 - - Power Shelf 0: Power Module Supply Type ------Input---- Volts Amps ------Output--- Volts Amps Status 0/PT0-PM0 Cisco 0.0 0.0 12.1 9.0 OK 0/PT0-PM1 Cisco 0.0 0.0 12.1 8.0 OK Location TEMPERATURE Sensor Value (deg C) Crit (Lo) Major (Lo) Minor (Hi) Minor (Hi) Major (Hi) Crit (Hi) 0/PT0-PM0 PM0-Supply (PSU_TEMP) 24 -10 -5 -1 60 65 70 Location TEMPERATURE Sensor Value (deg C) Crit (Lo) Major (Lo) Minor (Hi) Minor (Hi) Major (Hi) Crit (Hi) 0/PT0-PM1 PM1-Supply (PSU_TEMP) 22 -10 -5

Task	Use this command
	-1 60 65 70 ========== CHASSIS LEVEL POWER INFO: 0 ========== Total output power capacity (-) : 0W + 0W Total output power required : 0W Total power input : 0W Total power output : 206W ========== Location Altitude Value (Meters) Source ---------- 0 0 config

Host Level Monitoring Information

You can monitor the host OS level utilization information for the Appliance and XRv 9000 VM as well. Use the below show commands in system admin mode to view the information.

Task	Use this command
To view the CPU information.	sysadmin-vm:0_RP0# **show virtual-platform cpu** System CPU utilization ---------- Linux 3.14.23-WR7.0.0.2_standard (host) 11/27/17 _x86_64_ (16 CPU) 02:27:49 CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle 02:27:49 all 4.06 0.00 4.66 0.01 0.00 0.06 0.00 0.00 0.00 91.21 02:27:49 0 0.84 0.00 1.72 0.02 0.00 0.30 0.00 0.00 0.00 97.12 02:27:49 1 2.08 0.00 2.31 0.01 0.00 0.10 0.00 0.00 0.00 95.50 02:27:49 2 0.99 0.00 1.73 0.01 0.00 0.05 0.00 0.00 0.00 97.22 ... 02:27:49 14 2.40 0.00 1.64 0.00 0.00 0.00 0.00 0.00 0.00 95.96 02:27:49 15 1.24 0.00 1.41 0.00 0.00 0.00 0.00 0.00 0.00 97.35

Task	Use this command

To view disk information	`sysadmin-vm:0_RP0# show virtual-platform disk`
	
	`System Disk Utilization`
	`---`
	`Filesystem 1K-blocks`
	`Used Available Use% Mounted on`
	`/dev/mapper/panini_vol_grp-host_lv0 991512`
	`425304 498624 47% /`
	`...`
	`/dev/mapper/panini_vol_grp-host_data_scratch_lv0 2007248`
	`3036 1884200 1% /misc/scratch`
	`/dev/mapper/panini_vol_grp-host_data_config_lv0 95088`
	`44 87876 1% /misc/config`
	`/dev/mapper/panini_vol_grp-host_data_log_lv0 479560`
	`8080 435640 2% /var/log`
	`none 512`
	`0 512 0% /mnt`
	`/dev/loop5 6060604`
	`1330192 4399508 24% /lxc_rootfs/panini_vol_grp-xr_lv0`

To view memory information	`sysadmin-vm:0_RP0# show virtual-platform memory`
	`System Memory Usage`
	`---`
	`MemTotal: 131982032 kB`
	`MemFree: 109636132 kB`
	`MemAvailable: 111675924 kB`
	`...`
	`HugePages_Total: 12`
	`...`
	`Hugepagesize: 1048576 kB`
	`...`
	
	In the above command output, the MemFree information is helpful to check if the Cisco IOS XRv 9000 system is experience memory exhaustion. The Hugepages field values help triage for VPE issue.

To view processor information	`sysadmin-vm:0_RP0# show virtual-platform processor`
	`System Processor Information`
	`---`
	`processor : 0`
	`vendor_id : GenuineIntel`
	`cpu family : 6`
	`...`
	`flags : fpu vme de pse tsc msr pae mce cx8 apic`
	`sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr`
	`sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm`
	`constant_tsc arch_perfmon pebs bts rep_good nopl xtopology`
	`nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64`
	`monitor ds_cpl vmx smx est tm2 ssse3 fma cx16 xtpr pdcm`
	`pcid dca sse4_1 sse4_2 x2apic movbe popcnt`
	`tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm`
	`3dnowprefetch ida arat epb xsaveopt pln pts dtherm`
	`tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust`
	`bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx smap`
	`...`
	`processor : 1`
	`...`
	
	The above command displays the detailed information of twenty eight cores in the Appliance. The flags information are useful to check if the CPU is properly set in a XRv9000 VM hypervisor settings.

Environmental Monitoring through UCS Cisco Integrated Management Controller (CIMC)

In Release prior to 6.4.1 , there is no XR based environment monitoring capability in the Appliance.

However, since the Appliance is built on a UCS server, server environmental monitoring can also be done through SNMP or IPMI interfaces directly through CIMC. Refer to the Cisco UCS Server documentation on how to configure and use SNMP or IPMI monitoring via CIMC.

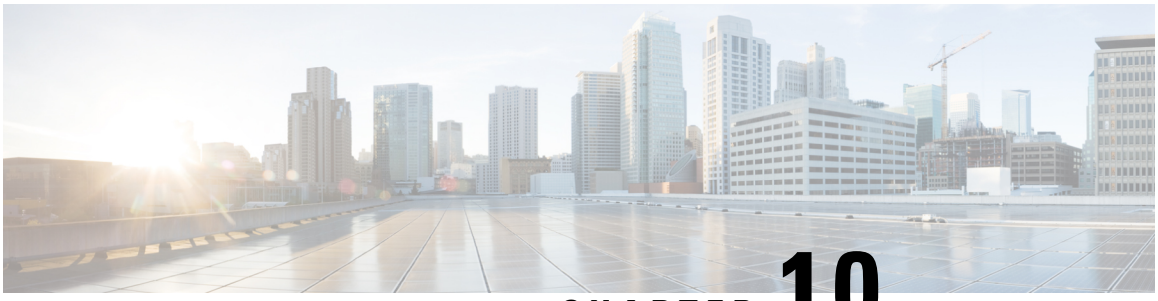
SFP Optic Monitoring Information

Cisco IOS XRv 9000 extracts optic health information from SFPs plugged into a NIC. The information includes vendor name, part number, current receiving power and transmiting power. To view the information use **show controllers <interface> physical**command in XR EXEC mode:

```
RP/0/RP0/CPU0:SS_Node1# show controllers TenGigE 0/0/0/1 physical
SFP EEPROM  port:1
        Xcvr Type: SFP
        Xcvr Code: SFP-10G-SR
        Encoding: 64B66B
        Bit Rate: 10300 Mbps
        Link Reach 50u fiber: 80 meter
        Link Reach 62.5u fiber: 20 meter
        Vendor Name: CISCO-JDSU
        Vendor OUI: 00.01.9c
        Vendor Part Number: PLRXPL-SC-S43-CS (rev.: 1   )
        Laser wavelength: 850 nm (fraction: 0.00 nm)
        Optional SFP Signal: Tx_Disable, Tx_Fault, LOS
        Vendor Serial Number: JUS1734G1L5
        Date Code (yy/mm/dd): 13/10/13  lot code:
        Diagnostic Monitoring: DOM, Int. Cal.,
        Enhanced Options: SW RX LOS Mon., SW TX Fault Mon, SW TX Disable, Alarm/Warning
Flags

...

        Temperature: 28.445
        Voltage: 3.300 Volt
        .
```

SFP OIR (online insertion and removal) information is monitored by polling the status of all SFPs every 5 seconds. The change in state is captured and reflected in a syslog message.

```
RP/0/RP0/CPU0:SS_Node1# show logging | i envmon
```



CHAPTER 10

Cisco IOS XRv 9000 Router Deployment on Amazon Web Services (AWS)

This chapter provides an overview of the Cisco IOS XRv 9000 Router deployment on Amazon Web Services.

Note You can deploy Cisco IOS XRv 9000 Router on AWS from Cisco IOS XR Release 6.3.1.

- Introduction, on page 153
- Cisco IOS XRv 9000 Router AMI Options for Amazon Web Services, on page 154
- Cisco IOS XRv 9000 Router Hourly-Billed AMIs, on page 154
- Supported Cisco IOS XR Technologies, on page 154
- Deploying the Cisco Cisco IOS XRv 9000 Router on Amazon Web Services, on page 155
- Information About Launching Cisco IOS XRv 9000 Router on AWS, on page 156
- Launching the Cisco IOS XRv 9000 Router AMI, on page 157

Introduction

The Cisco IOS XRv 9000 Router can be deployed on Amazon Web Services (AWS) for public and private cloud solutions. The implementation and installation on AWS is different than for the other supported hypervisors. The Cisco IOS XRv 9000 Router is supported on the following AWS platforms:

- AWS Virtual Private Cloud (VPC)
- AWS Elastic Compute Cloud (EC2)

For more information, see the AWS VPC documentation at:

http://aws.amazon.com/documentation/vpc/

Cisco IOS XRv 9000 Router AMI Options for Amazon Web Services

The Cisco IOS XRv 9000 Router for AWS is purchased and launched as an Amazon Machine Image (AMI) on AWS Marketplace .

Note You can upgrade the Cisco IOS XRv 9000 software through normal IOS XR upgrade procedures. But you can not downgrade to a release prior to IOS XR Release 6.3.1.

Cisco IOS XRv 9000 Router Hourly-Billed AMIs

The Cisco IOS XRv 9000 Router for AWS is purchased and launched as an Amazon Machine Image (AMI) on AWS Marketplace. This section describes the Hourly-Billed AMI.

A Cisco IOS XRv 9000 Router hourly-billed AMI, launched directly from AWS Marketplace, is subject to the following conditions:

- You are billed hourly by Amazon Web Services (AWS) for using the Cisco IOS XRv 9000 Router AMI. This hourly usage fee is in addition to the VPC usage fees charged by AWS.
- You do not need to purchase, install, or configure any type of licensing to use an Hourly Billed Cisco IOS XRv 9000 Router.

For information on the not supported IOS XR features, refer the topic *Supported Cisco IOS XR Technologies* .

For information on the supported IOS XR features, refer the latest release notes at this location Release Notes for Cisco IOS XRv 9000 Router.

Supported Cisco IOS XR Technologies

When deployed on an AWS instance, the Cisco IOS XRv 9000 Router supports fewer Cisco IOS XR technologies than are supported by other hypervisors. Some technologies may not be available because they are not supported in an Amazon cloud.

The following restrictions apply to deploying the Cisco IOS XRv 9000 Router on an AWS instance:

- Although CLI commands for unsupported features may be visible on the Cisco IOS XRv 9000 Router, testing by Cisco has determined that these unsupported features do not work in AWS deployments.
- Routing protocols are supported over a tunnel only.

Here is the list of Cisco IOS XR technologies that are supported when deploying the Cisco IOS XRv 9000 Router on an AWS instance.

- Single virtual router VM - vPE

- Routing - limited to static and BGP within AWS but can support dynamic routing using GREv4 tunnels
- IPv4, IPv6 L3 Forwarding
- GREv4 with IPv4/IPv6 payload
- e1000
- QOS - IPv4/IPv6 QOS (Policing/Marking/H-QOS/Egress-TM), Hierarchical Policers (conform aware)
- IPv4/6 ACLs (chained)
- Strict IPv4/IPv6 uRPF
- LPTS based CoPP
- EFD DOS protection
- IPSLA

Here is the list of Cisco IOS XR technologies that are not supported when deploying the Cisco IOS XRv 9000 Router on an AWS instance.

- Virtual Route Reflector (vRR)
- App Hosting
- Bundles
- BFD IPv4 and BFD IPv6 Single Hop (Static and BGP)
- VM serial port and console access
- VLAN

For more information about the features contained in the Cisco IOS XRv 9000 Router technology packages, see the Cisco IOS XRv 9000 Router Installation and Configuration Guide.

The following caveat applies to the Cisco IOS XR technology support on AWS deployments:

- You cannot configure HSRP between the Cisco IOS XRv 9000 Router nodes in an Amazon cloud. Amazon does not allow running HSRP on the hosts in the VPC. Amazon AWS blocks all broadcast and multicast traffic in a VPC.

Deploying the Cisco Cisco IOS XRv 9000 Router on Amazon Web Services

Before attempting to launch the Cisco IOS XRv 9000 Router on AWS, the following prerequisites apply:

- You must have an Amazon Web Services account.
- FireFox is more stable with AWS than other browsers and is recommended.
- An SSH client (for example, Putty on Windows or Terminal on Macintosh) is required to access the Cisco IOS XRv 9000 Router AWS console.

- Determine the instance type that you want to deploy for the Cisco IOS XRv 9000 Router. See the next section for more information.
- If you are planning to launch the AMI using the 1-Click Launch, you must first create a Virtual Private Cloud (VPC). For more information, see Amazon Virtual Private Cloud (VPC) .

Information About Launching Cisco IOS XRv 9000 Router on AWS

Launching the Cisco IOS XRv 9000 Router AMI takes place directly from the AWS Marketplace.

Determine whether the Cisco IOS XRv 9000 Router will be deployed on an Amazon EC2 instance or on an Amazon VPC instance.

If you are using an Amazon VPC instance, see the Launching the Cisco IOS XRv 9000 Router AMI Using the Manual Launch, on page 158. This section also mentions that in order to launch an instance, you need to generate a key pair or use an existing key pair.

Jumbo frames in a VPC have limitations; see this document: Network Maximum Transmission Unit (MTU) for Your EC2 Instance.

Supported Instance Types

The Amazon Machine Image supports different instance types, which determine the size of the instance and the required amount of memory.

Table 26: Feature History Table

Feature Name	Release	Description
Enhanced Networking Features with Elastic Network Adapter (ENA) on Amazon EC2 M5 Instances	Release 7.3.3	You can launch your router with the Elastic Network Adapter (ENA) on Amazon Elastic Compute Cloud (Amazon EC2) M5 instances to deliver high network throughput. Amazon EC2 M5 instances provide more CPU cores, faster disk speeds, and higher network bandwidth that boosts the network performance.

The following AMI instance types are supported for the Cisco IOS XRv 9000 Router:

Table 27: Supported AMI Instance Types

Instance Type	vCPU	Memory (GB)	Maximum NICs	Maximum IPv4 and IPv6 Addresses
m4.xlarge	4	16	4	15
m4.2xlarge	8	32	4	15
m4.4xlarge	16	64	8	30

Instance Type	vCPU	Memory (GB)	Maximum NICs	Maximum IPv4 and IPv6 Addresses
m4.10xlarge	40	160	8	30
c4.xlarge	4	7.5	4	15
c4.2xlarge	8	15	2	15
c4.4xlarge	16	30	8	30
c4.8xlarge	36	60	8	30

The following additional AWS EC2 M5 instance types are supported for the Cisco IOS XRv 9000 Router:

- m5
- m5n
- c5
- c5n

To know more about the maximum number of network interfaces supported per instance type, and the maximum number of private IPv4 addresses and IPv6 addresses per network interface, see the AWS user guide.

To know more about these AWS instanace types, see the Amazon Instance Types page from the Amazon Web Services documentation.

Launching the Cisco IOS XRv 9000 Router AMI

To launch the Cisco IOS XRv 9000 Router AMI, perform the steps in the following sections:

First, see: Selecting the Cisco IOS XRv 9000 Router AMI , on page 157.

If you are using an Amazon EC2 or VPC instance, see: Launching the Cisco IOS XRv 9000 Router AMI Using the Manual Launch, on page 158.

Then, see:Associating the elastic IP Address with Cisco IOS XRv 9000 Router Instance, on page 160 and Connecting to the IOS XRv 9000 Instance using SSH, on page 161.

Selecting the Cisco IOS XRv 9000 Router AMI

To select the Cisco IOS XRv 9000 Router AMI, perform the following steps:

Procedure

Step 1 Log in to Amazon Web Services Marketplace.

Step 2 Search AWS Marketplace for: “Cisco IOS XRv 9000”. A list of AMIs such as the following, appears:

- Cisco IOS XRv 9000 Demo Version (hourly billing)

Step 3 Select the Cisco IOS XRv 9000 Router AMI that you are planning to deploy.

The AMI information page displays, showing the supported instance types and the hourly fees charged by AWS. Select the pricing details for your region.

Click **Continue**.

Step 4 Enter your AWS email address and password, or create a new account.

The "Launch on EC2 page" displays.

Launching the Cisco IOS XRv 9000 Router AMI Using the Manual Launch

Procedure

Step 1 On the Launch with EC2 page, select the **Region** from the drop-down list.

Step 2 Choose the Cisco IOS XRv 9000 Router release version from the **Select a Version** pane.

The hourly usage charges for your region are shown under Pricing Details.

Step 3 Click the **Launch with EC2 Console** button for your region.

The window to select the instance type displays.

Select the General purpose tab for the supported instance types. Select the instance type.

Click the **Next: Configure Instance Details** button.

Step 4 Configure the instance details.

- Select the network from the network drop-down list. Select a VPC subnet, into which you want to deploy the IOS XRv 9000, from the **Subnet** drop-down. Keep in mind that this determines the availability zone of your instance.

  Keep default settings for **Auto-assign Public IP**.

  You can initially create two interfaces on the Instance Details screen. Afterwards, to add more interfaces, click Add Device in **Network Interfaces**. The maximum number of interfaces that are supported depends on the instance type.

- Select additional options available from AWS.

Step 5 (Optional) Click the **Next: Add Storage** button.

Step 6 (Optional) Keep the default hard drive setting.

Note
When operating the Cisco IOS XRv 9000 Router in AWS, the (46 GB EBS) size of virtual hard drives cannot be changed.

(Optional) Click the **Next: Add Tags** button.

Step 7 (Optional) Enter the tag information as needed.

(Optional) Click the **Next: Configure Security Groups** button.

Step 8 (Optional) Choose one of the following:

- Create a new Security Group
- Select an existing Security Group

The Cisco IOS XRv 9000 Router requires SSH for console access. The Cisco IOS XRv 9000 Router also requires that the Security Group, at a minimum, does not block TCP/22. These settings are used to manage the Cisco IOS XRv 9000 Router.

Click the **Review and Launch** button.

Step 9 Review the Cisco IOS XRv 9000 Router instance information.

Click **Launch**.

Step 10 When prompted, enter the key pair information. The key pair consists of a public key stored in AWS and your private key used to authenticate access to the instance. Do one of the following:

a) Choose an existing key pair, or
b) Create a new key by performing the following steps:

- Upload your own public key
- Create a new key pair on AWS:

  Click on **Create Key Pair.** Enter the key pair name and click Create. After the key pair is created, ensure that you have downloaded the private key from Amazon before continuing. A newly created private key can only be accessed once. After the key pair is downloaded, click **Close**.

Note
AWS security policies require that the private key permission level be set to 400. To set this value for the .pem file, open a UNIX shell terminal screen and enter the following command: **chmod 400** *pem-file-name*

Step 11 Click **Launch Instance**.

It takes approximately ten minutes to deploy the AMI instance. You can view the status by clicking on the Instances link on the menu.

Wait for the State to show **Running** and the Status Checks to show **passed**.

At this point, the Cisco IOS XRv 9000 Router AWS instance is booted and ready for software configuration. Proceed to the sections: Associating the elastic IP Address with Cisco IOS XRv 9000 Router Instance, on page 160 and Connecting to the IOS XRv 9000 Instance using SSH, on page 161.

Day Zero Configuration

The day zero configuration also known as bootstrap configuration is the configuration applied when the router boots for the first time. The day zero configuration should be entered into the User Data box as CLI (command line interface). Here is the sample:

```
username root
group root-lr
group cisco-support
secret 5 $1$920D$OrPQMgw1/3WdUe5R3RpLP/
```

```
!
interface TenGigE 0/0/0/0
ipv4 address 192.0.2.2/255.255.255.0
no shutdown
!
router static
address-family ipv4 unicast
0.0.0.0/0 192.0.2.2
!
!
ssh server v2
ssh server vrf default
```

Note If you use your own User data box, you should configure a username in order to connect to the box using SSH.

Associating the elastic IP Address with Cisco IOS XRv 9000 Router Instance

Before you can access the management console using an SSH connection, you must associate an interface on the Cisco IOS XRv 9000 Router with the elastic IP address created with the VPC. Perform the following steps:

Procedure

Step 1 On the Services > EC2 > Instances page, select the Cisco IOS XRv 9000 instance.

Step 2 In the displayed Network interfaces, click **eth0**.

A popup window displays showing detailed information about the **eth0** interface.

Note the interface's private IP address.

Step 3 Copy the **Interface ID** .

Step 4 In EC2 Dashboard > Network & Security, click Elastic IPs.

Step 5 Select the Elastic IP to which you want to associate IP from the list.

Step 6 From **Actions** drop-down, select Associate address.

Step 7 In Assicate Address page, do the following:

a) Select Network Interface as **Resource Type**.
b) In the **Network Interfaces** field paste the interface ID copied in Step 3.
c) Select **Private IP** address assigned by AWS from drop-down and check **Allow Elastic IP to be reassociated if already attached**.
d) Click **Associate**.

This action associates the elastic IP address (Amazon elastic IP) with the private IP address of the network interface. You can now use this interface to access the management console. See the Connecting to the IOS XRv 9000 Instance using SSH, on page 161.

Connecting to the IOS XRv 9000 Instance using SSH

The Cisco IOS XRv 9000 Router instance on AWS requires SSH for console access. To access the Cisco IOS XRv 9000 Router AMI, perform the following steps:

Procedure

Step 1 Once the Cisco IOS XRv 9000 Router status shows that is it is running, select the instance.

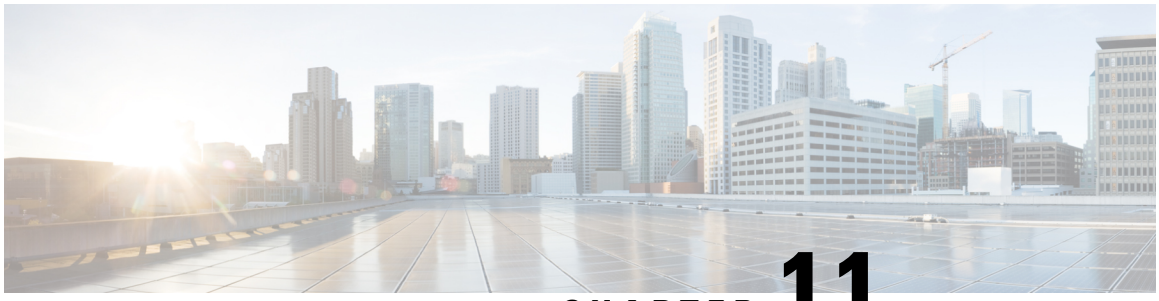
Step 2 Enter the following UNIX shell command to connect to the Cisco IOS XRv 9000 Router console using SSH:

ssh -i *pem-file-name* **root**@[*public-ipaddress* | *DNS-name*]

Note
You must log in as **`root`** the first time you access the instance.

The private key stored in the .pem file is used to authenticate access to the Cisco IOS XRv 9000 Router instance.

Step 3 Start configuring the Cisco IOS XRv 9000 Router.



CHAPTER 11

Appendix

The appendix consists of reference topics.

- Virtual Machine Requirements, on page 163
- Benefits of Virtualization Using Cisco IOS XRv 9000 Router , on page 166
- Cisco IOS XRv 9000 Router Architecture-Differences from Hardware Platforms, on page 166
- Support Information for Platforms and Cisco Software Images , on page 168
- VMware ESXi Support Information, on page 168
- KVM Support on OpenStack, on page 173
- Certificate management for M7, on page 173

Virtual Machine Requirements

Cisco IOS XRv 9000 Router runs only on a virtual machine. This section describes the virtual machine requirements for the router.

Virtual Machine

A virtual machine (VM) is a software implementation of a computing environment in which an operating system (OS) or program can be installed and run. The VM typically emulates a physical computing environment, but requests for CPU, memory, hard disk, network, and other hardware resources. These are managed by a virtualization layer which translates these requests to the underlying physical hardware.

You can use an Open Virtualization Archive (OVA) file to deploy VM. The OVA file package simplifies the process of deploying a VM by providing a complete definition of the parameters and resource allocation requirements for the new VM.

An OVA file consists of a descriptor (.ovf) file, a storage (.vmdk) file and a manifest (.mf) file.

- ovf file—Descriptor file, an xml file with extension .ovf, which consists of all metadata about the package. It encodes all product details, virtual hardware requirements, and licensing.
- vmdk file—File format that encodes a single virtual disk from a VM.
- mf file—Optional file that stores the SHA key generated during packaging.

You can also install Cisco IOS XRv 9000 Router using an .iso file and manually create the VM in the hypervisor.

Hypervisor Support

A hypervisor enables multiple operating systems to share a single host machine. While each operating system appears to have the dedicated use of the host's processor, memory, and other resources, the hypervisor actually controls and allocates only needed resources to each operating system and ensures that the operating systems (VMs) do not disrupt each other.

Installation of Cisco IOS XRv 9000 Router is supported on selected Type 1 (native, bare metal) hypervisors. Installation is not supported on Type 2 (hosted) hypervisors, such as VMware Fusion, VMware Player, or Virtual Box. The following table lists hypervisor versions supported in the latest Cisco IOS XR Software Release.

Table 28: Support Matrix for Hypervisor Versions

Cisco IOS XR Version	VMWare ESXi	Kernel Based Virtual Machine (KVM)
Release 7.3.1	version 6.7 and newer	Linux KVM based on • Red Hat Enterprise Linux 7, 7.1, 7.2, 7.3 and 7.4 • Ubuntu 14.04.03 LTS • Ubuntu 16.04 LTS • CentOS 7, 7.1, 7.2, 7.3, 7.4 • Openstack 10
Release 24.4.1	Version 7	Linux KVM based on • Ubuntu 20.04 and 24.04

The features available in a hypervisor may differ depending on their type. Not all hypervisor features in a given version may be supported. The hypervisor versions listed are those officially tested and supported by Cisco IOS XRv 9000 Router. See the following sections for more information.

- VMware ESXi Support Information

Hypervisor NIC Requirements

The specific Cisco IOS XR Software Release determines the type and maximum number of NICs supported by hypervisor.

NIC Hot Add/Remove allows you to add and remove NICs without the need to power down the VM in some Cisco IOS XR software versions and hypervisors.

This table lists the supported NICs for each VM instance.

Cisco IOS XR Release	5.4	6.0.x, 6.1.x, 6.2.x, 6.3.x, 6.4.x, 6.5.x	24.1.x, 24.2.x, 24.3.x, 24.4.x
VMware ESXi			

NIC Types Supported	E1000	E1000, VMXNET Generation 3 (VMXNET3) for traffic interfaces only.	Physical NIC:E810 (SRIOV VF only), X710 Virtual NIC:E1000, VMXNet3 (for traffic interfaces only)
Maximum number of NICs per VM instance	11 (one for management, two are reserved, and eight for traffic)	11 (one for management, two are reserved, and eight for traffic)	11 (one for management, two are reserved, and eight for traffic)
NIC Hot Add/Remove Support	No	No	No
Single Root I/O virtualization (SR-IOV) Support	No	No	Yes
Physical OIR Support	No	No	You need to power down the VM, complete the OIR process, and power up the VM.
KVM			
NIC Types Supported	VirtIO, ixgbe/ixgbevf	VirtIO, ixgbe/ixgbevf	Physical NICs (traffic interfaces only):X710 Virtual NICs:e1000, VirtIO
Maximum number of NICs per VM instance	11 (one for management, two are reserved, and eight for traffic)	11 (one for management, two are reserved, and eight for traffic)	11 (one for management, two are reserved, and eight for traffic)
NIC Hot Add/Remove Support	No	No	No
Single Root I/O virtualization (SR-IOV) Support	No	No	No

See the section Installation Requirements for KVM, on page 17 for information on physical NICs supported by the Cisco IOS XRv 9000 Router in KVM environments.

Cisco IOS XRv 9000 Router and Hypervisor Limitations

Cisco IOS XRv 9000 Router limitations are:

- Cisco IOS XRv 9000 Router interface bandwidth defaults to 1GB for all virtualized interfaces, irrespective of the hypervisor's physical NIC bandwidth.
- When a Cisco IOS XRv 9000 Router is using virtualized interfaces or virtual functions (not physical pass-through), and that interface is directly connected to a physical router and the physical router's connecting interface goes down, the change is not reflected on Cisco IOS XRv 9000 Router. This is because the Cisco IOS XRv 9000 Router is actually connected to the hypervisor's vSwitch and the vSwitch uplink port is connected to the physical interface of the router. This is expected behavior.
- Cisco IOS XRv 9000 Router provides an MTU range of 64-9216 bytes but, Cisco advises you to use minimum MTU value of 68 bytes. However, VMWare ESXi vSwitches support maximum frame size of 9000 bytes.

Benefits of Virtualization Using Cisco IOS XRv 9000 Router

Cisco IOS XRv 9000 Router provides these virtualization benefits in a cloud environment.

Table 29: Virtualization Benefits

Benefits	Description
Hardware independence	Because Cisco IOS XRv 9000 Router runs on a virtual machine, it can be supported on any x86 hardware that the virtualization platform supports.
Resources sharing	The resources used by Cisco IOS XRv 9000 Router are managed by the hypervisor, and resources can be shared among VMs. The amount of hardware resources that the VM server allocates to a specific VM can be reallocated to another VM on the server.
Flexibility in deployment	You can easily move a VM from one server to another. Thus, you can move Cisco IOS XRv 9000 Router from a server in one physical location to a server in another physical location without moving any hardware resources.

Cisco IOS XRv 9000 Router Architecture-Differences from Hardware Platforms

Unlike traditional Cisco hardware router platforms, Cisco IOS XRv 9000 Router is a virtual router that runs independently on an x86 machine. As a result, Cisco IOS XRv 9000 Router architecture has unique attributes that differentiate it from hardware-based router platforms.

This table compares some key areas where Cisco IOS XRv 9000 Router differs from the Cisco ASR 9000 Series Router.

Table 30: Cisco IOS XRv 9000 Series Router architecture differences with Cisco ASR 9000 Series Router

Feature	Cisco ASR 9000 Series	Cisco IOS XRv 9000 Series
Distributed routing	Distributed routing system which consists of RP and LCs. LCs are inter-connected through fabric.	Centralized routing system which consists of a combination of RP and LC. Because it is a virtualized platform there are no LCs and no fabric.
Control plane and data plane separation	Control plane and data plane located in the same chassis	Architecturally supports control plane and data plane separation. Supports data plane OIR.
Interface naming	The line interface is hosted on LC. The name of line interface indicates the location of the interface in the chassis. For example, Tenge 0/0/0/0 is the first port of LC slot 0.	The line interface is hosted on RP. The name of the line interface represents the instance of certain type. For example, Tenge 0/0/0/0 is the first instance of the Tenge interface.
Cluster	Supports cluster of ASR9000 routers as one logical router.	Not supported
Satellite interface	Supports satellite interface.	Not supported
Control plane redundancy	Supports active and standby RP	Not supported
Dynamic resource allocation	The resources are fixed	Memory and CPU can be dynamically allocated during installation.
Physical resources	Managed by architecture of the hardware platform	Memory and CPU can be assigned during VM provisioning, but requires a reboot for changes to take effect.
Console types supported	Physical serial port	• VGA console • Serial port [default]
ROMMON	Supported	The Cisco IOS XRv 9000 does not include ROMMON, but uses GRUB to provide similar but more limited functionality.
ISSU	Supported	Not supported
Interface module	Supports installation of pluggable interface module	Not supported
Dynamic addition/deletion of ports	Supported	Supported, but requires VM reload. **Note** Power down the VM before adding or removing interfaces in VMware ESXi and KVM environment.

Support Information for Platforms and Cisco Software Images

Cisco software is packaged in feature sets consisting of software images that support specific platforms. The feature sets available for a specific platform depend on which Cisco software images are included in a release. To identify the set of software images available in a specific release or to find out if a feature is available in a given Cisco IOS XR software image, you can use Cisco Feature Navigator, the Software advisor, or the software release notes.

Cisco Feature Navigator

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS XR software images support a specific software release, feature set, or platform. An account on Cisco.com is not required to access Cisco Feature Navigator.

Software Advisor

To see if a feature is supported by a Cisco IOS XR release, to locate the software document for that feature, or to check the minimum Cisco IOS XR software requirements with your router, Cisco maintains the Software Advisor tool on Cisco.com. You must be a registered user on Cisco.com to access this tool.

Software Release Notes

Cisco IOS XR software release notes provide the following information:

- Platform support
- Memory recommendations
- New features
- Open and resolved severity 1 and 2 caveats

Release notes are intended to be release-specific for the most current release, and the information provided in these documents may not be cumulative in providing information about features that first appeared in previous releases. See Cisco Feature Navigator for cumulative feature information.

For more information, see the Cisco IOS XRv 9000 Router Release Notes page.

VMware ESXi Support Information

Cisco IOS XRv 9000 Router runs on the VMware ESXi hypervisor. You can use the single VMware ESXi hypervisor to run several VMs. Use the VMware vSphere client GUI to create and manage VMs.

The VMware vSphere Client is an application for creating, configuring, and managing VMs on the VMware vCenter Server. Cisco IOS XRv 9000 Router can boot from a virtual disk located on the data store. You can perform basic administration tasks such as start and shutdown Cisco IOS XRv 9000 Router using the VMware vSphere client.

VMware vCenter Server manages the vSphere environment and provides unified management of all the hosts and VMs in the data center from a single console.

This table lists the VMware virtual machine vendor tools supported for the Cisco IOS XRv 9000 Router.

Table 31: VMware Virtual Machine Requirements

Cisco IOS XRv 9000	Supported Tools and Requirements	Supported vSwitch
Release 7.3.1	PC running VMware vSphere Client 5.5, 6.0 Server running VMware ESXi 6.7 and newer version VMware vCenter installation tool	VMware standard and distributed switch
Release 25.2.1	PC running VMware vSphere Client 5.5, 6.0 Server running VMware ESXi 8 VMware vCenter installation tool	VMware standard and distributed switch

Supported VMware Features and Operations

VMware supports various features and operations that allow you to manage your virtual applications and perform operations such as cloning, migration, shutdown and resume.

Some of these operations cause the runtime state of the VM to be saved and then restored upon restarting. If the runtime state includes traffic-related state, then on resumption or replaying the runtime state, additional errors, statistics, or messages maybe displayed on the user console. If the saved state is just configuration driven, you can use these features and operations without a problem.

This table lists the VMware features and operations that are supported on Cisco IOS XRv 9000 Router in the latest Cisco IOS XR Software Releases. For more information about VMware features and operations, see the VMware Documentation.

Table 32: Supported VMware Features and Operations: Storage Options (for Both vCenter Server and vSphere Client)

Entities	Status	Description
Local Storage	Supported	Local storage is in the internal hard disks located inside your ESXi host. Local storage devices do not support sharing across multiple hosts. A datastore on a local storage device can be accessed by only one host.
External Storage Target	Supported	You can deploy Cisco IOS XRv 9000 Router on external storage, that is; a Storage Area Network (SAN).

Entities	Status	Description
Mount or Pass Through of USB Storage	Not supported	You can connect USB sticks to Cisco IOS XRv 9000 Router and use them as storage devices. In VMware ESXi, you need to add a USB controller and then assign the disk devices to Cisco IOS XRv 9000 Router.

The following table lists features that are supported or not-supported in the latest Cisco IOS XR Software Releases.

Table 33: Supported VMware Features and Operations: General Features (for vCenter Server Only)

Entities	Status	Description
Cloning	Supported	Enables cloning a virtual machine or template, or cloning a virtual machine to a template.
Migrating	Not supported	The entire state of the virtual machine as well as its configuration file, if necessary, is moved to the new host even while the data storage remains in the same location on shared storage.
vMotion	Not supported	Enables moving the VM from one physical server to another while the VM remains active.
Template	Supported	Uses templates to create new virtual machines by cloning the template as a virtual machine.

This table lists supported VMware features and operations for both vCenter Server and vSphere Client in the latest Cisco IOS XR Software Releases.

Table 34: Supported VMware Features and Operations: Operations (for Both vCenter Server and vSphere Client)

Entities	Status	Description
Power On	Supported	Powers on the virtual machine and boots the guest operating system if the guest operating system is installed.
Power Off	Supported	Stops the virtual machine until it is powered back. The power off option performs a “hard” power off, which is analogous to pulling the power cable on a physical machine and always works.
Shut Down	Not supported	Shut Down, or “soft” power off, leverages VMware Tools to perform a graceful shutdown of a guest operating system. In certain situations, such as when VMware Tools is not installed or the guest operating system is hung, shut down might not succeed and using the Power off option is necessary.
Suspend	Not supported	Suspends the virtual machine.

Entities	Status	Description
Reset/Restart	Supported	Stops the virtual machine and restarts (reboots) it.
OVF Creation	Supported	An OVF package captures the state of a virtual machine into a self-contained package. You can create the OVF file by exporting it to your local computer.
OVA Creation	Supported	Single file (OVA) to package the OVF template into a single .ova file. This enables distributing the OVF package as a single file, if it needs to be explicitly downloaded from a website or moved around using a USB key.

This table lists supported VMware features and operations: Networking Features in the latest Cisco IOS XR Software Releases

Table 35: Supported VMware Features and Operations: Networking Features

Entities	Status	Description
Custom MAC address	Supported	From both vCenter Server and vSphere Client. Allows you to set up the MAC address manually for a virtual network adapter.
Distributed vSwitch	Supported	From vCenter Server only. A vSphere distributed switch on a vCenter Server data center can handle networking traffic for all associated hosts on the data center.
Distributed Resources Scheduler	Not supported	Provides automatic load balancing across hosts.
NIC Load Balancing	Not supported	From both vCenter Server and vSphere Client. Load balancing and failover policies allow you to determine how network traffic is distributed between adapters and how to reroute traffic if an adapter fails.

Entities	Status	Description
NIC Teaming	Not supported	From both vCenter Server and vSphere Client. Allows you to set up an environment where each virtual switch connects to two uplink adapters that form a NIC team. The NIC teams can then either share the load of traffic between physical and virtual networks among some or all of its members, or provide passive failover in the event of a hardware failure or a network outage. **Note** NIC Teaming can cause a large number of ARP packets to flood Cisco IOS XRv 9000 Router and overload the CPU. To avoid this situation, reduce the number of ARP packets and implement NIC Teaming as Active-Standby rather than Active-Active.
vSwitch	Supported	From both vCenter Server and vSphere Client. A vSwitch is a virtualized version of a Layer 2 physical switch. A vSwitch can route traffic internally between virtual machines and link to external networks. You can use vSwitches to combine the bandwidth of multiple network adapters and balance communications traffic among them. You can also configure a vSwitch to handle a physical NIC failover.

This table lists not-supported VMware features and Operations: High Availability, in the latest Cisco IOS XR Software Releases.

Table 36: Not-supported VMware Features and Operations: High Availability

Entities	Status	Description
VM-Level High Availability	Not supported	To monitor operating system failures, VM-Level High Availability monitors heartbeat information in the VMware High Availability cluster. Failures are detected when no heartbeat is received from a given virtual machine within a user-specified time interval. VM-Level High Availability is enabled by creating a resource pool of VMs using VMware vCenter Server.
Host-Level High Availability	Not supported	To monitor physical servers, an agent on each server maintains a heartbeat with the other servers in the resource pool such that a loss of heartbeat automatically initiates the restart of all affected virtual machines on other servers in the resource pool. Host-Level High Availability is enabled by creating a resource pool of servers or hosts, and enabling high availability in vSphere.

Entities	Status	Description
Fault Tolerance	Not supported	Using high availability, fault tolerance is enabled on the ESXi host. When you enable fault tolerance on the VM running Cisco IOS XRv 9000 Router, a secondary VM on another host in the cluster is created. If the primary host goes down, then the VM on the secondary host will take over as the primary VM for Cisco IOS XRv 9000 Router.

Note The Cisco IOS XRv 9000 Router does not support Active/Standby control plane redundancy.

KVM Support on OpenStack

Cisco IOS XRv 9000 router supports installation of a KVM in the OpenStack environment. The OpenStack support requires the qcow2 installation file available on the Cisco.com download page.

For information on supported OpenStack and Red Hat Enterprise Linux versions, see latest Release Notes for Cisco IOS XRv 9000 Router for Cisco IOS XR Software .

Certificate management for M7

Cisco IOS XRv 9000 router supports these commands to facilitate certificate management for M7 setup.

- **show platform security variable customer dba location** *location*
- **show platform security variable customer dbx location** *location*
- **platform security variable customer [append|update] dba** *file_location* **location** *location*
- **platform security variable customer [append|update] dbx** *file_location* **location** *location*

Use the command **show platform security variable customer [dba|dbx] location** *location* to display the details of customer-provided certificates stored in the security database. You can replace *location* with either a specific node location to narrow down your query or use *all* to retrieve certificate information from all nodes.

Use the command **platform security variable customer [append|update] [dba|dbx]** *file_location* **location** *location* to append or update a customer-provided security variable or certificate from the specified file (*file_location*) to the security database.