Juniper Networks | Engineering Simplicity

vMX

vMX Getting Started Guide for KVM

Published
2021-09-21

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

vMX vMX Getting Started Guide for KVM

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

Use this guide to install the virtual MX router in the KVM environment. This guide also includes basic vMX configuration and management procedures.

After completing the installation and basic configuration procedures covered in this guide, refer to the Junos OS documentation for information about further software configuration on the vMX router.

RELATED DOCUMENTATION

- Junos OS for MX Series Documentation

1

CHAPTER

vMX Overview

vMX Overview

IN THIS SECTION

Read this topic to get an overview about vMX virtual routers.
The vMX router is a virtual version of the MX Series 3D Universal Edge Router. Like the MX Series router, the vMX router runs the Junos operating system (Junos OS) and supports Junos OS packet handling and forwarding modeled after the Trio chipset. Configuration and management of vMX routers are the same as for physical MX Series routers, allowing you to add the vMX router to a network without having to update your operations support systems (OSS).

You install vMX software components on an industry-standard x86 server running a hypervisor, either the kernel-based virtual machine (KVM) hypervisor or the VMware ESXi hypervisor.

For servers running the KVM hypervisor, you also run the Linux operating system and applicable third-party software. vMX software components come in one software package that you install by running an orchestration script included with the package. The orchestration script uses a configuration file that you customize for your vMX deployment. You can install multiple vMX instances on one server.

For servers running the ESXi hypervisor, you run the applicable third-party software.

Some Junos OS software features require a license to activate the feature. To understand more about vMX Licenses, see, vMX Licenses for KVM and VMware. Please refer to the Licensing Guide for general information about License Management. Please refer to the product Data Sheets for further details, or contact your Juniper Account Team or Juniper Partner.

Benefits and Uses of vMX Routers

You can use virtual devices to lower your capital expenditure and operating costs, sometimes through automating network operations. Even without automation, use of the vMX application on standard x86 servers enables you to:

- Quickly introduce new services
- More easily deliver customized and personalized services to customers
- Scale operations to push IP services closer to customers or to manage network growth when growth forecasts are low or uncertain
- Quickly expand service offerings into new sites

A well designed automation strategy decreases costs as well as increasing network efficiency. By automating network tasks with the vMX router, you can:

- Simplify network operations
- Quickly deploy new vMX instances
- Efficiently install a default Junos OS configuration on all or selected vMX instances
- Quickly reconfigure existing vMX routers

You can deploy the vMX router to meet some specific network edge requirements, such as:

- Network simulation
- Terminate broadband subscribers with a virtual broadband network gateway (vBNG)
- Temporary deployment until a physical MX Series router is available

Automation for vMX Routers

Automating network tasks simplifies network configuration, provisioning, and maintenance. Because the vMX software uses the same Junos OS software as MX Series routers and other Juniper Networks routing devices, vMX supports the same automation tools as Junos OS. In addition, you can use standard automation tools to deploy the vMX, as you do other virtualized software.

Architecture of a vMX Instance

The vMX architecture is organized in layers:

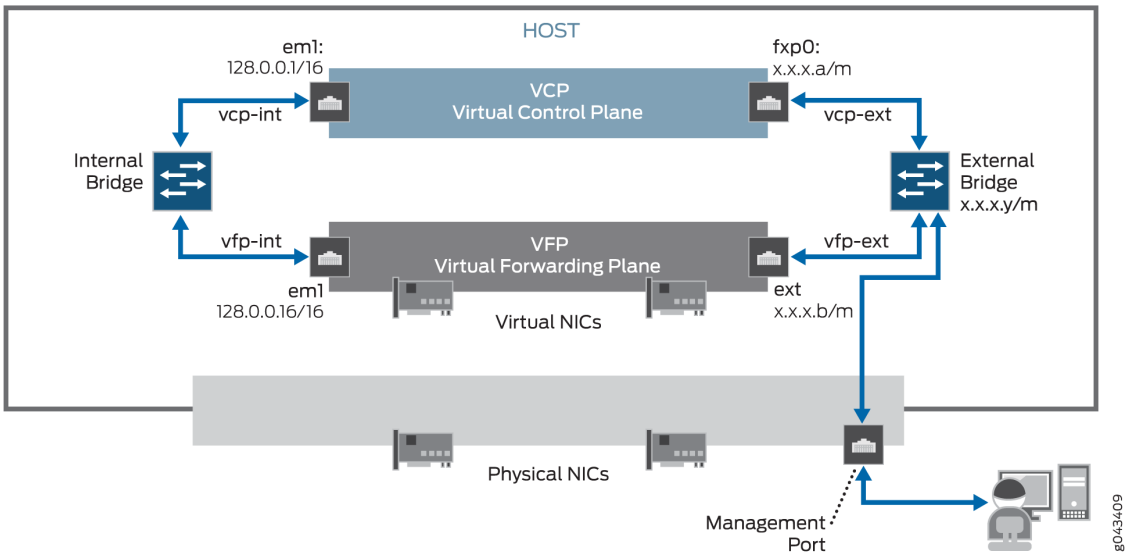
- The vMX router at the top layer
- Third-party software and the hypervisor in the middle layer

Linux, third-party software, and the KVM hypervisor in the middle layer in Junos OS Release 15.1F3 or earlier releases. In Junos OS Release 15.1F3 and earlier releases, the host contains the Linux operating system, applicable third-party software, and the hypervisor.

- The x86 server in the physical layer at the bottom

Figure 1 on page 4 illustrates the architecture of a single vMX instance inside a server. Understanding this architecture can help you plan your vMX configuration.

Figure 1: vMX Instance in a Server



The physical layer of the server contains the physical NICs, *CPU*s, memory, and Ethernet management port. The host contains applicable third-party software and the hypervisor.

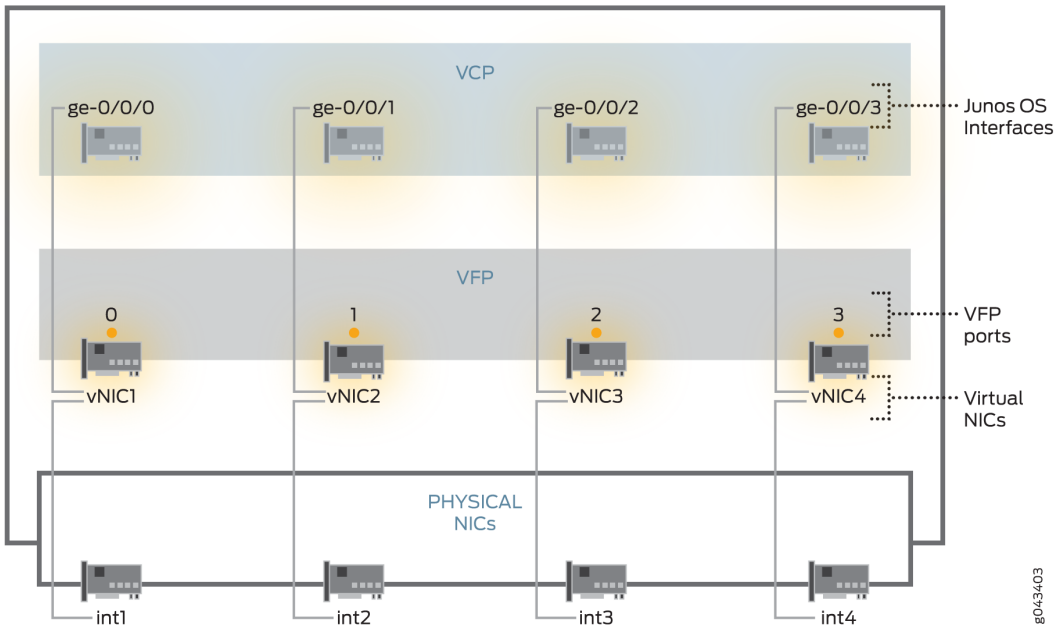
Supported in Junos OS Release 15.1F3 and earlier releases, the host contains the Linux operating system, applicable third-party software, and the hypervisor.

The vMX instance contains two separate virtual machines (VMs), one for the virtual forwarding plane (VFP) and one for the virtual control plane (VCP). The VFP VM runs the virtual Trio forwarding plane software and the VCP VM runs Junos OS.

The hypervisor presents the physical NIC to the VFP VM as a virtual NIC. Each virtual NIC maps to a vMX interface. Figure 2 on page 5 illustrates the mapping.

The orchestration script maps each virtual NIC to a vMX interface that you specify in the configuration file. After you run the orchestration script and the vMX instance is created, you use the Junos OS CLI to configure these vMX interfaces in the VCP (supported in Junos OS Release 15.1F3 or earlier releases).

Figure 2: NIC Mapping



After the vMX instance is created, you use the Junos OS CLI to configure these vMX interfaces in the VCP. The vMX router supports the following types of interface names:

- Gigabit Ethernet (ge)
- 10-Gigabit Ethernet (xe)
- 100-Gigabit Ethernet (et)

NOTE: vMX interfaces configured with the Junos OS CLI and the underlying physical NIC on the server are independent of each other in terms of interface type (for example, ge-0/0/0 can get mapped to a 10-Gigabit NIC).

The VCP VM and VFP VM require Layer 2 connectivity to communicate with each other. An *internal* bridge that is local to the server for each vMX instance enables this communication.

The VCP VM and VFP VM also require Layer 2 connectivity to communicate with the Ethernet management port on the server. You must specify virtual Ethernet interfaces with unique IP addresses

and MAC addresses for both the VFP and VCP to set up an *external* bridge for a vMX instance. Ethernet management traffic for all vMX instances enters the server through the Ethernet management port.

The way network traffic passes from the physical NIC to the virtual NIC depends on the virtualization technique that you configure.

vMX can be configured to run in two modes depending on the use case:

- Lite mode—Needs fewer resources in terms of CPU and memory to run at lower bandwidth.
- Performance mode—Needs higher resources in terms of CPU and memory to run at higher bandwidth.

> **NOTE**: Performance mode is the default mode.

Traffic Flow in a vMX Router

The x86 server architecture consists of multiple sockets and multiple cores within a socket. Each socket also has memory that is used to store packets during I/O transfers from the NIC to the host. To efficiently read packets from memory, guest applications and associated peripherals (such as the NIC) should reside within a single socket. A penalty is associated with spanning CPU sockets for memory accesses, which might result in non-deterministic performance.

The VFP consists of the following functional components:

- Receive thread (RX): RX moves packets from the NIC to the VFP. It performs preclassification to ensure host-bound packets receive priority.
- Worker thread: The Worker performs lookup and tasks associated with packet manipulation and processing. It is the equivalent of the lookup ASIC on the physical MX Series router.
- Transmit thread (TX): TX moves packets from the Worker to the physical NIC.

The RX and TX components are assigned to the same core (I/O core). If there are enough cores available for the VFP, the QoS scheduler can be allocated separate cores. If there are not enough cores available, the QoS scheduler shares the TX core.

TX has a QoS scheduler that can prioritize packets across several queues before they are sent to the NIC (supported in Junos OS Release 16.2).

The RX and TX components can be dedicated to a single core for each 1G or 10G port for the most efficient packet processing. High-bandwidth applications must use SR-IOV. The Worker component utilizes a scale-out distributed architecture that enables multiple Workers to process packets based on

packets-per-second processing needs. Each Worker requires a dedicated core (supported in Junos OS Release 16.2).

RELATED DOCUMENTATION

Virtual Network Interfaces for vMX | 7

https://www.juniper.net/documentation/en_US/release-independent/licensing/topics/topic-map/vmx-licensing.html

Virtual Network Interfaces for vMX

IN THIS SECTION

In a virtual environment, packet input and output capabilities play a significant role in the performance of the packet processing functionality inside the virtual machine, specifically the VFP VM. VFP supports two types of virtual network interfaces:

- Paravirtualized—Paravirtualized network interfaces use network drivers in the guest OS and host OS that interact with the virtual environment and communicate effectively to give higher performance than fully emulated interfaces. In KVM, the supported paravirtualized interface is virtio. For VMware, VMXNET3 is supported.
- *PCI pass-through*—PCI pass-through enables PCI devices such as network interfaces to appear as if they were physically attached to the guest operating system, bypassing the hypervisor and providing a high rate of data transfer. The physical network interfaces support single root I/O virtualization (SR-IOV) capability and can be connected to the VMs using PCI pass-through.

Choose the type based on how you want to use the vMX router. SeeTable 1 on page 8.

Table 1: Considerations for Choosing a Virtualization Technique

Consideration	Paravirtualization Technique	PCI Passthrough Technique
Interfaces	virtio (for KVM), VMXNET3 (for VMware)	SR-IOV
Use Cases	• Network simulation • Low-throughput applications	• Static vMX deployments • High-throughput applications
Host Requirements	No requirements specific to this technique	Physical NIC must support PCI passthrough
VM Mobility (Junos OS Release 15.1F4 or earlier releases)	Moving vMX instance to a new server without reconfiguration.	Creating an identical vMX instance on a new server.

Paravirtualization

Supported in Junos OS Release 15.1F4, in a paravirtualized router, the VM and the host work together to efficiently move packets from the physical NIC to the application in the VM. You implement paravirtualization on the vMX router by configuring virtio, a technique that the KVM hypervisor supports that optimizes network and disk operations for the VM. Both the VFP VM and the host contain virtio drivers that interact to move packets. You implement paravirtualization on the VMware server by configuring VMXNET3 on the ESXi hypervisor. You must provide the following information in the configuration file for each vMX interface:

- Junos OS name
- Unique MAC address

If you want to move the VM from one server to another, you can do so without reconfiguration, provided the names and MAC addresses of each interface remain the same.

PCI Passthrough with SR-IOV

Supported in Junos OS Release 15.1F4, The vMX router supports PCI passthrough in combination with single root I/O virtualization (SR-IOV). In the PCI passthrough technique, you directly assign a NIC's memory space to a VM, enabling packets to bypass the hypervisor. Bypassing the hypervisor increases efficiency and results in high throughput of packets.

With SR-IOV, the hypervisor detects the physical NICs (known as a physical functions) and creates multiple virtual NICs (known as virtual functions) in the VFP VM. In the vMX implementation, the host dedicates a NIC to a single VM.

When you configure PCI passthrough with SR-IOV, you specify the following parameters for each vMX interface:

- Junos OS name
- Unique MAC address
- Name of the physical NIC

Because you create a direct connection between a virtual NIC and a physical NIC, you cannot move a VM from one host to another. If you need to move a VM to another host, you must install a new vMX instance on that host, and delete the vMX instance on the original host.

RELATED DOCUMENTATION

vMX Overview | 2

Licenses for vMX

2

CHAPTER

Installing and Deploying vMX on KVM

Minimum Hardware and Software Requirements

The tables lists the hardware requirements.

Table 2: Minimum Hardware Requirements for vMX

Description	Value
Sample system configuration	For lab simulation and low performance (less than 100 Mbps) use cases, any x86 processor (Intel or AMD) with VT-d capability. For all other use cases, Intel Ivy Bridge processors or later are required. Example of Ivy Bridge processor: Intel Xeon E5-2667 v2 @ 3.30 GHz 25 MB Cache For single root I/O virtualization (SR-IOV) NIC type, use Intel Ivy Bridge CPU (or higher) and Intel x520 NICs using ixgbe driver or X710 NICs with 10G ports and using i40e driver. Any other NIC models are not supported.
	For Junos OS Release 19.1R1-S1 and Junos OS Release 19.2R1 onwards, for single root I/O virtualization (SR-IOV) NIC type, use Intel Ivy Bridge CPU (or higher) and Intel x520 NICs using ixgbe driver, or X710 and XL710 NICs with 10G ports using i40e driver or XL710Q-DA2 NIC with 40G ports using i40e driver. Any other NIC models are not supported. **NOTE**: XL710Q-DA2 with 40G ports is only supported with i40e driver version 2.4.10 or later on Ubuntu 16.04 or RHEL 7.5 When using 40G ports on the vMX instances, quality-of-service (QoS) is not supported.
Number of cores **NOTE**: Performance mode is the default mode and the minimum value is based on one port.	For lite mode with lab simulation use case applications: Minimum of 4 • 1 for VCP • 3 for VFP **NOTE**: If you want to use lite mode when you are running with more than 3 vCPUs for the VFP, you must explicitly configure lite mode.

Table 2: Minimum Hardware Requirements for vMX *(Continued)*

Description	Value
	For performance mode with low-bandwidth (virtio) or high-bandwidth (SR-IOV) applications: Minimum of 9 • 1 for VCP • 8 for VFP The exact number of required vCPUs differs depending on the Junos OS features that are configured and other factors, such as average packet size. You can contact Juniper Networks Technical Assistance Center (JTAC) for validation of your configuration and make sure to test the full configuration under load before use in production. For typical configurations, we recommend the following formula to calculate the minimum vCPUs required by the VFP: • Without QoS—(4 * *number-of-ports*) + 4 • With QoS—(5 * *number-of-ports*) + 4 **NOTE**: All VFP vCPUs must be in the same physical non-uniform memory access (NUMA) node for optimal performance. In addition to vCPUs for the VFP, we recommend 2 x vCPUs for VCP and 2 x vCPUs for Host OS on any server running the vMX.

Table 2: Minimum Hardware Requirements for vMX *(Continued)*

Description	Value
Memory **NOTE**: Performance mode is the default mode.	For lite mode: Minimum of 3 GB • 1 GB for VCP • 2 GB for VFP For performance mode: • Minimum of 5 GB 1 GB for VCP 4 GB for VFP • Recommended of 16 GB 4 GB for VCP 12 GB for VFP Additional 2 GB recommended for host OS
Storage	Local or NAS Each vMX instance requires 44 GB of disk storage Minimum storage requirements: • 40 GB for VCP • 4 GB for VFP
vNICs	SR-IOV **NOTE**: SR-IOV is only supported with Intel Ivy Bridge CPU (or higher) and Intel x520 NICs using ixgbe driver or X710 NICs with 10G ports and using i40e driver. Any other NIC models are not supported. Support for unmodified ixgbe driver and i40e driver is available from Junos OS Release 18.4R1 onwards.

Table 2: Minimum Hardware Requirements for vMX *(Continued)*

Description	Value
Other requirements	Intel VT-d capability Hyperthreading (recommended) AES-NI

Table 3 on page 14 lists the software requirements.

Table 3: Software Requirements for Ubuntu

Description	Value

Operating system
NOTE: Only English localization is supported.
	• For Junos OS 20.1R1 and later releases:
	  • Ubuntu 18.04.3 LTS
	  • Linux 4.15.0-70-generic
	• For Junos OS 18.2 and later releases:
	  • Ubuntu 16.04.5 LTS
	  • Linux 4.4.0-62-generic
	• Prior to Junos OS 18.2 Release
	  • Ubuntu 14.04.1 LTS
	  • Linux 3.19.0-80-generic

Virtualization	• QEMU-KVM 2.11.1(Debian 1:2.11+dfsg-1ubuntu7.20) For Ubuntu 18.04.3 LTS (For Junos OS Release 20.1R1) • QEMU-KVM 2.0.0+dfsg-2ubuntu1.11

Table 3: Software Requirements for Ubuntu *(Continued)*

Description	Value
Required packages **NOTE**: Other additional packages might be required to satisfy all dependencies.	The required packages might change depending upon the supported Ubuntu version. • For Ubuntu 18.04.3 LTS. bridge-utils qemu-kvm libvirt-bin python python-netifaces,vnc4server libyaml-dev python-yaml numactl libparted0-dev libpciaccess-dev libnuma-dev libyajl-dev libxml2-dev libglib2.0-dev libnl-3-dev python-pip python-dev libxslt1-dev • The required packages (Previous releases) bridge-utils qemu-kvm libvirt-bin python python-netifaces vnc4server libyaml-dev python-yaml numactl libparted0-dev libpciaccess-dev libnuma-dev libyajl-dev libxml2-dev libglib2.0-dev libnl-dev python-pip python-dev libxml2-dev libxslt-dev Libvirt versions: • libvirt 1.2.19 • libvirt 1.3.1 (Junos OS 18.2 and later releases) • libvirtd (libvirt) 4.0.0 (Junos OS Release 20.1R1 and later releases)

NOTE: Use the `apt-get install` *`pkg name`* or `sudo apt-get install <pkg-name>` commands to install a package.

Table 4 on page 16 lists the software requirements for Red Hat Enterprise Linux.

Table 4: Software Requirements for Red Hat Enterprise Linux

Description	Value
Operating system **NOTE**: Only English localization is supported.	• Junos OS Release 20.3R1 Red Hat Enterprise Linux Server 7.7 Kernel: 3.10.0-1062.4.1.el7.x86_64 • Junos OS Release 19.4R1 Red Hat Enterprise Linux Server 7.6 Kernel: 3.10.0-862.el7.x86_64 • Junos OS Release 19.1R1-S1 and Junos OS Release 19.2R1 Red Hat Enterprise Linux Server 7.5 (Maipo) Kernel: 3.10.0-862.el7.x86_64 • Junos OS Release 17.4R1 Red Hat Enterprise Linux 7.2 Kernel: 3.10.0-327.4.5 • Junos OS Release 17.3R1 Red Hat Enterprise Linux 7.3 Kernel: 3.10.0-514.6.2
Virtualization	QEMU-KVM 1.5.3
Required packages **NOTE**: SR-IOV requires these packages: kernel-devel gcc	python27-python-pip python27-python-devel numactl-libs libpciaccess-devel parted-devel yajl-devel libxml2-devel glib2-devel libnl-devel libxslt-devel libyaml-devel numactl-devel redhat-lsb kmod-ixgbe libvirt-daemon-kvm numactl telnet net-tools **NOTE**: libvirt 1.2.17 or later

NOTE: Use the `yum install` *pkg name* command to install a package.

Table 5 on page 17 lists the software requirements for CentOS.

Table 5: Software Requirements for CentOS

Description	Value
Operating system **NOTE**: Only English localization is supported.	CentOS 7.2 Linux 3.10.0-327.22.2
Virtualization	QEMU-KVM 1.5.3
Required packages	python27-python-pip python27-python-devel numactl-libs libpciaccess-devel parted-devel yajl-devel libxml2-devel glib2-devel libnl-devel libxslt-devel libyaml-devel numactl-devel redhat-lsb kmod-ixgbe libvirt-daemon-kvm numactl telnet net-tools **NOTE**: libvirt 1.2.19 To avoid any conflicts, install libvirt 1.2.19 instead of updating from libvirt 1.2.17.

NOTE: Use the `yum install` *pkg name* command to install a package.

RELATED DOCUMENTATION

- *Preparing the Ubuntu Host to Install vMX*
- *Preparing the Red Hat Enterprise Linux Host to Install vMX*
- *Preparing the CentOS Host to Install vMX*
- *Installing vMX for Different Use Cases*

vMX Package Contents

Table 6 on page 18 lists the contents of the vMX package.

Table 6: vMX Package Contents

Filename	Description
vmx.sh	Main orchestration script. Note: Only English locale is supported for using the vmx.sh script.
vmx_release.txt	vMX release information details
config/	Startup configuration file: • config/vmx.conf—Configuration file for defining vMX parameters. • config/vmx-junosdev.conf—Configuration file for binding devices (for virtio NICs). See "Specifying vMX Configuration File Parameters" on page 52 for more information.
drivers/	Source files for modified ixgbe and i40e drivers.
env/	OS environment settings.
images/	Software image files. • images/junos-vmx-x86-64-*.qcow2—Software image files for VCP. • images/vmxhdd.img—Software image file for VCP file storage. • images/vFPC_*.img—Software image file for VFP.
openstack	Scripts and xml files for open stack deployment.

Table 6: vMX Package Contents *(Continued)*

Filename	Description
scripts	Juniper Networks orchestration scripts.

The vMX package consists of the following components: (in Junos OS Release 15.1F4 and earlier releases)

```
build
config
– vmx.conf
– vmx-junosdev.conf
docs
drivers
– ixgbe-3.19.1
env
images
– jinstall64-vmx-15.1F4.15-domestic.img
– jinstall64-vmx-15.1F4.15-domestic-signed.img
– vmxhdd.img
– vFPC_20151203.img
scripts
– common
– junosdev-bind
– kvm
– templates
vmx.sh
```

NOTE: Modified IXGBE drivers are included in the package. Multicast promiscuous mode for Virtual Functions is needed to receive control traffic that comes with broadcast MAC addresses. The reference driver does not come with this mode set, so the IXGBE drivers in this package contain certain modifications to overcome this limitation.

RELATED DOCUMENTATION

Minimum Hardware and Software Requirements | 11

Installing vMX on KVM

IN THIS SECTION

Read this topic to understand how to install the virtual MX router in the KVM environment.

Preparing the Ubuntu Host to Install vMX

To prepare the Ubuntu host system for installing vMX (Starting in Junos OS Release 15.1F6):

1. Meet the minimum software and OS requirements described in "Minimum Hardware and Software Requirements" on page 11.
2. See "Upgrading Kernel " on page 22and "Upgrading to libvirt 1.2.19" on page 22 sections below.
3. If you are using Intel XL710 PCI-Express family cards, make sure you update the drivers. See "Updating Drivers for the X710 NIC" on page 24.
4. Enable Intel VT-d in BIOS. (We recommend that you verify the process with the vendor because different systems have different methods to enable VT-d.)

   Refer to the procedure to enable VT-d available on the Intel Website.

5. Disable KSM by setting `KSM_ENABLED=0` in **/etc/default/qemu-kvm**.

6. Disable APIC virtualization by editing the **/etc/modprobe.d/qemu-system-x86.conf** file and adding `enable_apicv=0` to the line containing `options kvm_intel`.

   ```
   options kvm_intel nested=1 enable_apicv=0
   ```

7. Restart the host to disable KSM and APIC virtualization.

8. If you are using SR-IOV, you must perform this step.

   NOTE: You must remove any previous installation with an external bridge in **/etc/network/interfaces** and revert to using the original management interface. Make sure that the `ifconfig -a` command does not show external bridges before you proceed with the installation.

   To determine whether an external bridge is displayed, use the `ifconfig` command to see the management interface. To confirm that this interface is used for an external bridge group, use the `brctl show` command to see whether the management interface is listed as an external bridge.

   Enable SR-IOV capability by turning on `intel_iommu=on` in the **/etc/default/grub** directory.

   ```
   GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on"
   ```

   Append the `intel_iommu=on` string to any existing text for the `GRUB_CMDLINE_LINUX_DEFAULT` parameter.

   Run the `update-grub` command followed by the `reboot` command.

9. For optimal performance, we recommend you configure the size of Huge Pages to be 1G on the host and make sure the NUMA node for the VFP has at least 16 1G Huge Pages. To configure the size of Huge Pages, add the following line in **/etc/default/grub**:

   ```
   GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=number-of-huge-pages"
   ```

   The number of Huge Pages must be at least (16G * *number-of-numa-sockets*).

10. Run the `modprobe kvm-intel` command before you install vMX.

NOTE: Starting in Junos OS 18.2 and later releases, Ubuntu 16.04.5 LTS and Linux 4.4.0-62-generic are supported.

To meet the minimum software and OS requirements, you might need to perform these tasks:

Upgrading the Kernel

NOTE: Upgrading Linux kernel in Ubuntu 16.04 version is not required.

NOTE: If you are using Ubuntu 14.04.1 LTS, which comes with 3.19.0-80-generic, you can skip this step. Ubuntu 14.04 comes with a lower version of kernel (Linux 3.13.0-24-generic) than the recommended version (Linux 3.19.0-80-generic).

To upgrade the kernel:

1. Determine your version of the kernel.

```
uname -a
Linux rbu-node-33 3.19.0-80-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64
x86_64 GNU/Linux
```

2. If your version differs from the version shown in step 1, run the following commands:

```
apt-get install linux-firmware
apt-get install linux-image-3.19.0-80-generic
apt-get install linux-image-extra-3.19.0-80-generic
apt-get install linux-headers-3.19.0-80-generic
```

3. Restart the system.

Upgrading to libvirt 1.2.19

NOTE: Ubuntu 16.04.5 supports Libvirt version is 1.3.1. Upgrading libvirt in Ubuntu 16.04 is not required.

Ubuntu 14.04 supports libvirt 1.2.2 (which works for VFP lite mode). If you are using the VFP performance mode or deploying multiple vMX instances using the VFP lite mode, you must upgrade to libvirt 1.2.19.

To upgrade libvirt:

1. Make sure that you install all the packages listed in "Minimum Hardware and Software Requirements" on page 11.
2. Navigate to the **/tmp** directory using the `cd /tmp` command.
3. Get the `libvirt-1.2.19` source code by using the command `wget http://libvirt.org/sources/libvirt-1.2.19.tar.gz`.
4. Uncompress and untar the file using the `tar xzvf libvirt-1.2.19.tar.gz` command.
5. Navigate to the **libvirt-1.2.19** directory using the `cd libvirt-1.2.19` command.
6. Stop libvirtd with the `service libvirt-bin stop` command.
7. Run the `./configure --prefix=/usr --localstatedir=/ --with-numactl` command.
8. Run the `make` command.
9. Run the `make install` command.
10. Make sure that the libvirtd daemon is running. (Use the `service libvirt-bin start` command to start it again. If it does not start, use the `/usr/sbin/libvirtd -d` command.)

```
root@vmx-server:~# ps aux | grep libvirtd
root      1509  0.0  0.0 372564 16452 ?        Sl   10:25   0:00 /usr/sbin/libvirtd -d
```

11. Verify that the versions of libvirtd and virsh are 1.2.19.

```
root@vmx-server:~# /usr/sbin/libvirtd --version
libvirtd (libvirt) 1.2.19
root@vmx-server:~#  /usr/bin/virsh --version
1.2.19
root@vmx-server:~#
```

The system displays the code compilation log.

NOTE: If you cannot deploy vMX after upgrading libvirt, bring down the virbr0 bridge with the `ifconfig virbr0 down` command and delete the bridge with the `brctl delbr virbr0` command.

Updating Drivers for the X710 NIC

If you are using Intel XL710 PCI-Express family NICs, make sure you update the drivers before you install vMX.

To update the drivers:

1. Download the vMX software package as root and uncompress the package.

```
tar xzvf package-name
```

2. Install the i40e driver from the installation directory.

```
cd drivers/i40e-1.3.46/src
make install
```

3. Install the latest i40evf driver from Intel.

   For example, the following commands download and install Version 1.4.15:

```
cd /tmp
wget https://downloadmirror.intel.com/26003/eng/i40evf-1.4.15.tar.gz
tar zxvf i40evf-1.4.15.tar.gz
cd i40evf-1.4.15/src
make install
```

4. Update initrd with the drivers.

```
update-initramfs -u -k 'uname -r'
```

5. Activate the new driver.

```
rmmod i40e
modprobe i40e
```

Install the Other Required Packages

Use the following commands to install python-netifaces package on Ubuntu.

```
apt-get install python-pip
apt-get install python-netifaces
pip install pyyaml
```

Preparing the Red Hat Enterprise Linux Host to Install vMX

IN THIS SECTION

To prepare the host system running Red Hat Enterprise Linux for installing vMX, perform the task for your version:

Preparing the Red Hat Enterprise Linux 7.3 Host to Install vMX

To prepare the host system running Red Hat Enterprise Linux 7.3 for installing vMX:

1. Meet the minimum software and OS requirements described in "Minimum Hardware and Software Requirements" on page 11.
2. Enable hyperthreading and VT-d in BIOS.

   If you are using SR-IOV, enable SR-IOV in BIOS.

   We recommend that you verify the process with the vendor because different systems have different methods to access and change BIOS settings.
3. During the OS installation, select the **Virtualization Host** and **Virtualization Platform** software collections.

   If you did not select these software collections during the GUI installation, use the following commands to install them:

   ```
   yum groupinstall "virtualization host"
   yum groupinstall "virtualization platform"
   ```

4. Register your host using your Red Hat account credentials. Enable the appropriate repositories.

```
subscription-manager register --username username --password password --auto-attach
subscription-manager repos --enable rhel-7-fast-datapath-htb-rpms
subscription-manager repos --enable rhel-7-fast-datapath-rpms
subscription-manager repos --enable rhel-7-server-extras-rpms
subscription-manager repos --enable rhel-7-server-nfv-rpms
subscription-manager repos --enable rhel-7-server-optional-rpms
subscription-manager repos --enable rhel-7-server-rh-common-rpms
subscription-manager repos --enable rhel-7-server-rhn-tools-beta-rpms
subscription-manager repos --enable rhel-7-server-rpms
subscription-manager repos --enable rhel-ha-for-rhel-7-server-rpms
subscription-manager repos --enable rhel-server-rhscl-7-rpms
```

   To install the Extra Packages for Enterprise Linux 7 (epel) repository:

```
yum -y install wget
cd /tmp/
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum -y install epel-release-latest-7.noarch.rpm
```

5. Update currently installed packages.

```
yum upgrade
```

6. For optimal performance, we recommend you configure the size of Huge Pages to be 1G on the host and make sure that the NUMA node for the VFP has at least sixteen 1G Huge Pages. To configure the size of Huge Pages, use the following step:

   For Red Hat: Add the Huge Pages configuration.

```
grubby --update-kernel=ALL --args="default_hugepagesz=huge-pages-size  hugepagesz=huge-
pages-size hugepages=number-of-huge-pages"
grub2-install /dev/boot-device-name
reboot
```

   Use the mount | grep boot command to determine the boot device name.

   The number of Huge Pages must be at least (16G * *number-of-numa-sockets*).

7. Install the required packages.

```
yum install python27-python-pip python27-python-devel numactl-libs libpciaccess-devel
parted-devel yajl-devel libxml2-devel glib2-devel libnl-devel libxslt-devel libyaml-devel
numactl-devel redhat-lsb kmod-ixgbe libvirt-daemon-kvm numactl telnet net-tools dosfstools
```

8. (Optional) If you are using SR-IOV, you must install these packages and enable SR-IOV capability.

```
yum install kernel-devel gcc
grubby --args="intel_iommu=on" --update-kernel=ALL
```

   Reboot and log in again.

9. Link the **qemu-kvm** binary to the **qemu-system-x86_64** file.

```
ln -s /usr/libexec/qemu-kvm /usr/bin/qemu-system-x86_64
```

10. Set up the path for the correct Python release and install the PyYAML library.

```
PATH=/opt/rh/python27/root/usr/bin:$PATH
export PATH
pip install netifaces pyyaml
```

11. If you have installed any Red Hat OpenStack libraries, you must change `script/templates/red_{vPFE, vRE}-ref.xml` to use `<type arch='x86_64' machine='pc-0.13'>hvm</type>` as the machine type.

12. Disable KSM.

```
systemctl disable ksm
systemctl disable ksmtuned
```

    To verify that KSM is disabled run the following command.

```
cat /sys/kernel/mm/ksm/run 0
```

    The value 0 in the output indicates that KSM is disabled.

13. Disable APIC virtualization by editing the **/etc/modprobe.d/kvm.conf** file and adding `enable_apicv=n` to the line containing options kvm_intel.

```
modprobe -r kvm_intel
```

```
vi /etc/modprobe.d/kvm.conf to add the following lines
options kvm-intel enable_apicv=n
```

You can use `enable_apicv=0` also.

```
modprobe kvm-intel
```

Restart the host to disable KSM and APIC virtualization.

14. Stop and disable Network Manager.

```
systemctl disable NetworkManager
systemctl stop NetworkManager
```

If you cannot stop Network Manager, you can prevent **resolv.conf** from being overwritten with the `chattr +I /etc/resolv.conf` command.

15. Ensure that the build directory is readable by the QEMU user.

```
chmod -R o+r,o+x build-directory-pathname
```

As an alternative, you can configure QEMU to run as the root user by setting the **/etc/libvirt/qemu.conf** file to `user="root"`.

You can now install vMX.

NOTE: When you install vMX with the `sh vmx.sh -lv --install` command, you might see a kernel version mismatch warning. You can ignore this warning.

Preparing the Red Hat Enterprise Linux 7.2 Host to Install vMX

To prepare the host system running Red Hat Enterprise Linux 7.2 for installing vMX:

1. Meet the minimum software and OS requirements described in "Minimum Hardware and Software Requirements" on page 11.
2. Enable hyperthreading and VT-d in BIOS.

   If you are using SR-IOV, enable SR-IOV in BIOS.

   We recommend that you verify the process with the vendor because different systems have different methods to access and change BIOS settings.
3. During the OS installation, select the **Virtualization Host** and **Virtualization Platform** software collections.

   If you did not select these software collections during the GUI installation, use the following commands to install them:

   ```
   yum groupinstall "virtualization host"
   yum groupinstall "virtualization platform"
   ```
4. Register your host using your Red Hat account credentials. Enable the appropriate repositories.

   ```
   subscription-manager register --username username --password password --auto-attach
   subscription-manager repos --enable rhel-server-rhscl-7-rpms
   subscription-manager repos --enable rhel-7-server-extras-rpms
   subscription-manager repos --enable rhel-7-server-rhn-tools-beta-rpms
   subscription-manager repos --enable rhel-7-server-optional-rpms
   ```
5. Update currently installed packages.

   ```
   yum upgrade
   ```
6. Install the required packages.

   ```
   yum install python27-python-pip python27-python-devel numactl-libs libpciaccess-devel
   parted-devel yajl-devel libxml2-devel glib2-devel libnl-devel libxslt-devel libyaml-devel
   numactl-devel redhat-lsb kmod-ixgbe libvirt-daemon-kvm numactl telnet net-tools dosfstools
   ```
7. For optimal performance, we recommend you configure the size of Huge Pages to be 1G on the host and make sure that the NUMA node for the VFP has at least sixteen 1G Huge Pages. To configure the size of Huge Pages, use the following step:

For Red Hat: Add the Huge Pages configuration.

```
grubby --update-kernel=ALL --args="default_hugepagesz=huge-pages-size  hugepagesz=huge-
pages-size hugepages=number-of-huge-pages"
grub2-install /dev/boot-device-name
 reboot
```

Use the mount | grep boot command to determine the boot device name.

The number of Huge Pages must be at least (16G * *number-of-numa-sockets*).

8. (Optional) If you are using SR-IOV, you must install these packages and enable SR-IOV capability.

```
yum install kernel-devel gcc
grubby --args="intel_iommu=on" --update-kernel=ALL
```

Reboot and log in again.

9. Link the **qemu-kvm** binary to the **qemu-system-x86_64** file.

```
ln -s /usr/libexec/qemu-kvm /usr/bin/qemu-system-x86_64
```

10. Set up the path for the correct Python release and install the PyYAML library.

```
PATH=/opt/rh/python27/root/usr/bin:$PATH
export PATH
pip install netifaces pyyaml
```

11. If you have installed any Red Hat OpenStack libraries, you must change `script/templates/red_{vPFE, vRE}-ref.xml` to use `<type arch='x86_64' machine='pc-0.13'>hvm</type>` as the machine type.

12. Disable KSM.

```
systemctl disable ksm
systemctl disable ksmtuned
```

To verify that KSM is disabled run the following command.

```
cat /sys/kernel/mm/ksm/run 0
```

The value 0 in the output indicates that KSM is disabled.

13. Disable APIC virtualization by editing the **/etc/modprobe.d/kvm.conf** file and adding `enable_apicv=n` to the line containing options kvm_intel.

```
modprobe -r kvm_intel
```

```
vi /etc/modprobe.d/kvm.conf to add the following lines
options kvm-intel enable_apicv=n
```

You can use `enable_apicv=0` also.

```
modprobe kvm-intel
```

Restart the host to disable KSM and APIC virtualization.

14. Stop and disable Network Manager.

```
systemctl disable NetworkManager
systemctl stop NetworkManager
```

If you cannot stop Network Manager, you can prevent **resolv.conf** from being overwritten with the `chattr +I /etc/resolv.conf` command.

15. Ensure that the build directory is readable by the QEMU user.

```
chmod -R o+r,o+x build-directory-pathname
```

As an alternative, you can configure QEMU to run as the root user by setting the **/etc/libvirt/qemu.conf** file to `user="root"`.

You can now install vMX.

NOTE: When you install vMX with the `sh vmx.sh -lv --install` command, you might see a kernel version mismatch warning. You can ignore this warning.

Preparing the CentOS Host to Install vMX

To prepare the host system running CentOS for installing vMX:

1. Meet the minimum software and OS requirements described in "Minimum Hardware and Software Requirements" on page 11.
2. Enable hyperthreading and VT-d in BIOS.

   If you are using SR-IOV, enable SR-IOV in BIOS.

   We recommend that you verify the process with the vendor because different systems have different methods to access and change BIOS settings.
3. During the OS installation, select the **Virtualization Host** and **Virtualization Platform** software collections.

   If you did not select these software collections during the GUI installation, use the following commands to install them:

   ```
   yum groupinstall "virtualization host"
   yum groupinstall "virtualization platform"
   ```
4. Enable the appropriate repositories.

   ```
   yum install -y "http://elrepo.org/elrepo-release-7.0-4.el7.elrepo.noarch.rpm"
   yum install centos-release-scl
   ```
5. Update currently installed packages.

   ```
   yum upgrade
   ```
6. Install the required packages.

   ```
   yum install python27-python-pip python27-python-devel numactl-libs libpciaccess-devel
   parted-devel yajl-devel libxml2-devel glib2-devel libnl-devel libxslt-devel libyaml-devel
   numactl-devel redhat-lsb kmod-ixgbe libvirt-daemon-kvm numactl telnet net-tools
   ```
7. (Optional) If you are using SR-IOV, you must install these packages and enable SR-IOV capability.

   ```
   yum install kernel-devel gcc
   grubby --args="intel_iommu=on" --update-kernel=ALL
   ```

Reboot and log in again.

8. Link the **qemu-kvm** binary to the **qemu-system-x86_64** file.

```
ln -s /usr/libexec/qemu-kvm /usr/bin/qemu-system-x86_64
```

9. Set up the path for the correct Python release and install the PyYAML library.

```
PATH=/opt/rh/python27/root/usr/bin:$PATH
export PATH
pip install netifaces pyyaml
```

NOTE: In case of error with installation, use the following workaround:

```
# yum install python27-python-pip
# scl enable python27 bash
 # source scl_source enable python27
# export LD_LIBRARY_PATH=/opt/rh/python27/root/usr/lib64
# pip install -upgrade pip
# pip install netifaces pyyaml
```

10. Disable KSM.

```
systemctl disable ksm
systemctl disable ksmtuned
```

To verify that KSM is disabled run the following command.

```
cat /sys/kernel/mm/ksm/run 0
```

The value 0 in the output indicates that KSM is disabled.

11. Disable APIC virtualization by editing the **/etc/modprobe.d/kvm.conf** file and adding `enable_apicv=0` to the line containing options kvm_intel.

- ```
  modprobe -r kvm_intel
  ```
  ```

- ```
  vi /etc/modprobe.d/kvm.conf to add the following lines
  options kvm-intel enable_apicv=0
  ```

- ```
  modprobe kvm-intel
  ```

Restart the host to disable KSM and APIC virtualization.

12. Stop and disable Network Manager.

    ```
    systemctl disable NetworkManager
    systemctl stop NetworkManager
    ```

    If you cannot stop Network Manager, you can prevent **resolv.conf** from being overwritten with the `chattr +I /etc/resolv.conf` command.

13. Ensure that the build directory is readable by the QEMU user.

    ```
    chmod -R o+r,o+x build-directory-pathname
    ```

    As an alternative, you can configure QEMU to run as the root user by setting the **/etc/libvirt/qemu.conf** file to `user=root`.

14. Add this line to the end of the **/etc/profile** file.

    ```
    export PATH=/opt/rh/python27/root/usr/bin:$PATH
    ```

You can now install vMX.

NOTE: When you install vMX with the `sh vmx.sh -lv --install` command, you might see a kernel version mismatch warning. You can ignore this warning.

Installing vMX for Different Use Cases

IN THIS SECTION

Installing vMX is different for specific use cases. Table lists the sample configuration requirements for some vMX use cases.

Table 7: Sample Configurations for Use Cases (supported in Junos OS Release 18.3 to 18.4)

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Lab simulation Up to 100 Mbps performance	4: 1 for VCP 3 for VFP	5 GB: 1 GB for VCP 4 GB for VFP	virtio
Low-bandwidth applications Up to 3 Gbps performance	10: 1 for VCP 9 for VFP	20 GB: 4 GB for VCP 16 GB for VFP	virtio
High-bandwidth applications or performance testing For 3 Gbps and beyond performance	10: 1 for VCP 9 for VFP	20 GB 4 GB for VCP 16 GB for VFP	SR-IOV

Table 7: Sample Configurations for Use Cases (supported in Junos OS Release 18.3 to 18.4) *(Continued)*

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Dual virtual Routing Engines **NOTE**: When deploying on separate hosts, you must set up a connection between the hosts for the VCPs to communicate with each other.	Double the number of VCP resources for your particular use case is consumed when deploying both VCP instances.	Double the number of VCP resources for your particular use case is consumed when deploying both VCP instances.	virtio or SR-IOV

Table 8: Sample Configurations for Use Cases (supported in Junos OS Release 18.1 to 18.2)

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Lab simulation Up to 100 Mbps performance	4: 1 for VCP 3 for VFP	5 GB: 1 GB for VCP 4 GB for VFP	virtio
Low-bandwidth applications Up to 3 Gbps performance	8: 1 for VCP 7 for VFP	16 GB: 4 GB for VCP 12 GB for VFP	virtio
High-bandwidth applications or performance testing For 3 Gbps and beyond performance	8: 1 for VCP 7 for VFP	16 GB 4 GB for VCP 12 GB for VFP	SR-IOV

Table 8: Sample Configurations for Use Cases (supported in Junos OS Release 18.1 to 18.2) *(Continued)*

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Dual virtual Routing Engines **NOTE**: When deploying on separate hosts, you must set up a connection between the hosts for the VCPs to communicate with each other.	Double the number of VCP resources for your particular use case is consumed when deploying both VCP instances.	Double the number of VCP resources for your particular use case is consumed when deploying both VCP instances.	virtio or SR-IOV

Table 9: Sample Configurations for Use Cases (supported in Junos OS Release 17.4)

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Lab simulation Up to 100 Mbps performance	4: 1 for VCP 3 for VFP	5 GB: 1 GB for VCP 4 GB for VFP	virtio
Low-bandwidth applications Up to 3 Gbps performance	8: 1 for VCP 7 for VFP	16 GB: 4 GB for VCP 12 GB for VFP	virtio
High-bandwidth applications or performance testing For 3 Gbps and beyond performance	8: 1 for VCP 7 for VFP	16 GB 4 GB for VCP 12 GB for VFP	SR-IOV

Table 10: Sample Configurations for Use Cases (supported in Junos OS Release 15.1F6 to 17.3)

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Lab simulation Up to 100 Mbps performance	4: 1 for VCP 3 for VFP	5 GB: 1 GB for VCP 4 GB for VFP	virtio
Low-bandwidth applications Up to 3 Gbps performance	8: 1 for VCP 7 for VFP	16 GB: 4 GB for VCP 12 GB for VFP	virtio
High-bandwidth applications or performance testing For 3 Gbps and beyond performance	8: 1 for VCP 7 for VFP	16 GB 4 GB for VCP 12 GB for VFP	SR-IOV

Table 11: Sample Configurations for Use Cases (supported in Junos OS Release 15.1F4 to 15.1F3)

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Lab simulation Up to 100 Mbps performance	4: 1 for VCP 3 for VFP	10 GB: 2 GB for VCP 8 GB for VFP	virtio
Low-bandwidth applications Up to 3 Gbps performance	4: 1 for VCP 3 for VFP	10 GB: 2 GB for VCP 8 GB for VFP	virtio or SR-IOV

Table 11: Sample Configurations for Use Cases (supported in Junos OS Release 15.1F4 to 15.1F3) *(Continued)*

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
High-bandwidth applications or performance testing For 3 Gbps and beyond performance (with minimum of two 10Gb Ethernet ports) Up to 80 Gbps of raw performance	8: 1 for VCP 7 for VFP	16 GB 4 GB for VCP 12 GB for VFP	SR-IOV

Table 12: Sample Configurations for Use Cases (supported in Junos OS Release 14.1)

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
Lab simulation Up to 100 Mbps performance	4: 1 for VCP 3 for VFP	8 GB: 2 GB for VCP 6 GB for VFP	virtio
Low-bandwidth applications Up to 3 Gbps performance	4: 1 for VCP 3 for VFP	8 GB: 2 GB for VCP 6 GB for VFP	virtio or SR-IOV

Table 12: Sample Configurations for Use Cases (supported in Junos OS Release 14.1) *(Continued)*

Use Case	Minimum vCPUs	Minimum Memory	NIC Device Type
High-bandwidth applications or performance testing For 3 Gbps and beyond performance (with minimum of two 10Gb Ethernet ports) Up to 80 Gbps of raw performance	5: 1 for VCP 4 for VFP	8 GB 2 GB for VCP 6 GB for VFP	SR-IOV

NOTE: From Junos OS Release 18.4R1 (Ubuntu host) and Junos OS Release 19.1R1 (RedHat host), you can set the `use_native_drivers` value to true in the vMX configuration file to use the latest unmodified drivers for your network interface cards for vMX installations

To install vMX for a particular use case, perform one of the following tasks:

Installing vMX for Lab Simulation

Starting in Junos OS Release 14.1, the use case for lab simulation uses the virtio NIC.

To install vMX for the lab simulation (less than 100 Mbps) application use case:

1. Download the vMX software package as root and uncompress the package.

   `tar xzvf` *`package-name`*

2. Change directory to the location of the uncompressed vMX package.

   `cd` *`package-location`*

3. Edit the **config/vmx.conf** text file with a text editor to configure a single vMX instance.

   Ensure the following parameter is set properly in the vMX configuration file:

   ```
   device-type : virtio
   ```

   See "Specifying vMX Configuration File Parameters" on page 52.

4. Run the `./vmx.sh -lv --install` script to deploy the vMX instance specified by the **config/vmx.conf** startup configuration file and provide verbose-level logging to a file. See "Deploying and Managing vMX" on page 52.

5. From the VCP, enable lite mode for the VFP.

```
user@vmx# set chassis fpc 0 lite-mode
```

Here is a sample vMX startup configuration file using the virtio device type for lab simulation:

```
---
#Configuration on the host side - management interface, VM images etc.
HOST:
    identifier                : vmx1   # Maximum 4 characters
    host-management-interface : eth0
    routing-engine-image      : "/home/vmx/vmxlite/images/junos-vmx-x86-64.qcow2"
    routing-engine-hdd        : "/home/vmx/vmxlite/images/vmxhdd.img"
    forwarding-engine-image   : "/home/vmx/vmxlite/images/vFPC.img"

---
#External bridge configuration
BRIDGES:
    - type  : external
      name  : br-ext                 # Max 10 characters

---
#vRE VM parameters
CONTROL_PLANE:
    vcpus       : 1
    memory-mb   : 1024
    console_port: 8601

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.94
        macaddr   : "0A:00:DD:C0:DE:0E"

---
#vPFE VM parameters
FORWARDING_PLANE:
    memory-mb   : 4096
    vcpus       : 3
    console_port: 8602
    device-type : virtio
```

```
    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.98
        macaddr   : "0A:00:DD:C0:DE:10"

---
#Interfaces
JUNOS_DEVICES:
   - interface            : ge-0/0/0
     mac-address          : "02:06:0A:0E:FF:F0"
     description          : "ge-0/0/0 interface"

   - interface            : ge-0/0/1
     mac-address          : "02:06:0A:0E:FF:F1"
     description          : "ge-0/0/1 interface"
```

Installing vMX for Low-Bandwidth Applications

Starting in Junos OS Release 14.1, the use case for low-bandwidth applications uses virtio or SR-IOV NICs.

To install vMX for the low-bandwidth (up to 3 Gbps) application use case:

1. Download the vMX software package as root and uncompress the package.

   `tar xzvf` *`package-name`*

2. Change directory to the location of the uncompressed vMX package.

   `cd` *`package-location`*

3. Edit the **config/vmx.conf** text file with a text editor to configure a single vMX instance.

   Ensure the following parameter is set properly in the vMX configuration file:

   `device-type: virtio` or `device-type: sriov`

   See "Specifying vMX Configuration File Parameters" on page 52.

4. Run the `./vmx.sh -lv --install` script to deploy the vMX instance specified by the **config/vmx.conf** startup configuration file and provide verbose-level logging to a file. See "Deploying and Managing vMX" on page 52.

5. From the VCP, enable performance mode for the VFP.

```
user@vmx# set chassis fpc 0 performance-mode
```

Here is a sample vMX startup configuration file using the virtio device type for low-bandwidth applications:

```
---
#Configuration on the host side - management interface, VM images etc.
HOST:
    identifier                : vmx1   # Maximum 4 characters
    host-management-interface : eth0
    routing-engine-image      : "/home/vmx/vmx/images/junos-vmx-x86-64.qcow2"
    routing-engine-hdd        : "/home/vmx/vmx/images/vmxhdd.img"
    forwarding-engine-image   : "/home/vmx/vmx/images/vFPC.img"

---
#External bridge configuration
BRIDGES:
    - type  : external
      name  : br-ext                  # Max 10 characters

---
#vRE VM parameters
CONTROL_PLANE:
    vcpus       : 1
    memory-mb   : 4096
    console_port: 8601

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.94
        macaddr   : "0A:00:DD:C0:DE:0E"

---
#vPFE VM parameters
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 9
    console_port: 8602
    device-type : virtio

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.98
        macaddr   : "0A:00:DD:C0:DE:10"
```

```
---
#Interfaces
JUNOS_DEVICES:
   - interface           : ge-0/0/0
     mac-address         : "02:06:0A:0E:FF:F0"
     description         : "ge-0/0/0 interface"

   - interface           : ge-0/0/1
     mac-address         : "02:06:0A:0E:FF:F1"
     description         : "ge-0/0/1 interface"
```

Installing vMX for High-Bandwidth Applications

Starting in Junos OS Release 14.1, the use case for high-bandwidth applications uses the SR-IOV NICs.

To install vMX for the high-bandwidth (above 3 Gbps) application use case:

1. Download the vMX software package as root and uncompress the package.

   `tar xzvf` *`package-name`*

2. Change directory to the location of the uncompressed vMX package.

   `cd` *`package-location`*

3. Edit the **config/vmx.conf** text file with a text editor to configure a single vMX instance.

   Ensure the following parameter is set properly in the vMX configuration file:

   ```
   device-type: sriov
   ```

   See "Specifying vMX Configuration File Parameters" on page 52.

4. Run the `./vmx.sh -lv --install` script to deploy the vMX instance specified by the **config/vmx.conf** startup configuration file and provide verbose-level logging to a file. See "Deploying and Managing vMX" on page 52.

5. From the VCP, enable performance mode for the VFP.

   ```
   user@vmx# set chassis fpc 0 performance-mode
   ```

Here is a sample vMX startup configuration file using the SR-IOV device type:

```
---
#Configuration on the host side - management interface, VM images etc.
HOST:
    identifier               : vmx1   # Maximum 4 characters
```

```
    host-management-interface : eth0
    routing-engine-image      : "/home/vmx/images/junos-vmx-x86-64.qcow2"
    routing-engine-hdd        : "/home/vmx/images/vmxhdd.img"
    forwarding-engine-image   : "/home/vmx/images/vFPC.img"

---
#External bridge configuration
BRIDGES:
    - type  : external
      name  : br-ext                  # Max 10 characters

---
#VCP VM parameters
CONTROL_PLANE:
    vcpus       : 1
    memory-mb   : 4096
    console_port: 8601

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.94
        macaddr   : "0A:00:DD:C0:DE:0E"

---
#VFP VM parameters
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 9
    console_port: 8602
    device-type : sriov

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.98
        macaddr   : "0A:00:DD:C0:DE:10"

---
#Interfaces
JUNOS_DEVICES:
   - interface            : ge-0/0/0
     port-speed-mbps      : 10000
     nic                  : eth1
     mtu                  : 2000
```

```
    virtual-function     : 0
    mac-address          : "02:06:0A:0E:FF:F0"
    description          : "ge-0/0/0 connects to eth1"

  - interface            : ge-0/0/1
    port-speed-mbps      : 10000
    nic                  : eth2
    mtu                  : 2000
    virtual-function     : 0
    mac-address          : "02:06:0A:0E:FF:F1"
    description          : "ge-0/0/1 connects to eth2"
```

For more information see, "Example: Enabling SR-IOV on vMX Instances on KVM" on page 83.

Installing vMX with Dual Routing Engines

You can set up redundant Routing Engines on the vMX server by creating the primary Routing Engine (re0) and backup Routing Engine (re1) in the CONTROL_PLANE section of the vMX startup configuration file (default file is **config/vmx.conf**).

> **NOTE**: When deploying the Routing Engines on separate hosts, you must set up a connection between the hosts for the VCPs to communicate with each other.

Starting in Junos OS Release 18.1 to install vMX for the dual Routing Engines use case:

1. Download the vMX software package as root and uncompress the package.

   `tar xzvf` *`package-name`*

2. Change directory to the location of the uncompressed vMX package.

   `cd` *`package-location`*

3. Edit the **config/vmx.conf** text file with a text editor to configure the vMX instance.

   The default `CONTROL_PLANE` section resembles the following with one interface entry:

```
CONTROL_PLANE:
    vcpus       : 1
    memory-mb   : 2048
    console_port: 8896

    interfaces  :
      - type      : static
```

```
        ipaddr    : 10.216.48.117
        macaddr   : "0A:01:03:A1:A1:02"
```

To set up the redundant Routing Engines:

a. Navigate to `CONTROL_PLANE` and specify the proper number of vCPUs (`vcpus`) and amount of memory (`memory-mb`).

b. Starting with Junos OS Release 18.1R1, add the `deploy` parameter to designate the Routing Engine instance deployed on this host. If you do not specify this parameter, all instances (0,1) are deployed on the host.

   When deploying the Routing Engines on separate hosts, you must set up a connection between the hosts for the VCPs to communicate with each other.

c. Modify the interfaces entry to add `instance : 0` after the `type` parameter to set up `re0`.

   Specify the `ipaddr` and `macaddr` parameters. This address is the management IP address for the VCP VM (fxp0).

d. Add another entry, but specify `instance : 1` to set up `re1` and specify the `console_port` parameter for `re1` after the `instance : 1` parameter.

   Specify the `ipaddr` and `macaddr` parameters. This address is the management IP address for the VCP VM (fxp0).

The revised `CONTROL_PLANE` section that deploys re0 on the host resembles the following example with two interface entries:

```
CONTROL_PLANE:
    vcpus        : 1
    memory-mb    : 4096
    console_port : 8896
    deploy       : 0

    interfaces :
      - type     : static
        instance : 0
        ipaddr   : 10.216.48.117
        macaddr  : "0A:01:03:A1:A1:02"

      - type         : static
        instance     : 1
        console_port : 8897
```

```
        ipaddr       : 10.216.48.118
        macaddr      : "0A:01:03:A1:A1:06"
```

See "Specifying vMX Configuration File Parameters" on page 52.

4. Run the `./vmx.sh -lv --install` script to deploy the vMX instance specified by the **config/vmx.conf** startup configuration file and provide verbose-level logging to a file. See "Deploying and Managing vMX" on page 52.
5. From the VCP, enable performance mode for the VFP.

```
user@vmx# set chassis fpc 0 performance-mode
```

6. When deploying the Routing Engines on separate hosts, you must set up a connection between the hosts for the VCPs to communicate with each other.

   For example, to set up a connection (such as br-int-vmx1) between the two hosts over an interface (such as eth1), run the following command on both hosts:

```
ifconfig eth1 up && brctl addif br-int-vmx1 eth1
```

Here is a sample vMX startup configuration file that is deploying the first Routing Engine instance on this host:

```
---
#Configuration on the host side - management interface, VM images etc.
HOST:
    identifier                : vmx1   # Maximum 4 characters
    host-management-interface : eth0
    routing-engine-image      : "/home/vmx/images/junos-vmx-x86-64.qcow2"
    routing-engine-hdd        : "/home/vmx/images/vmxhdd.img"
    forwarding-engine-image   : "/home/vmx/images/vFPC.img"

---
#External bridge configuration
BRIDGES:
    - type  : external
      name  : br-ext                 # Max 10 characters

---
#VCP VM parameters
CONTROL_PLANE:
    vcpus        : 1
```

```
    memory-mb    : 4096
    console_port : 8601
    deploy       : 0

    interfaces  :
      - type      : static
        instance  : 0
        ipaddr    : 10.102.144.94
        macaddr   : "0A:00:DD:C0:DE:0E"

      - type         : static
        instance     : 1
        console_port : 8612
        ipaddr       : 10.102.144.95
        macaddr      : "0A:00:DD:C0:DE:0F"

---
#VFP VM parameters
FORWARDING_PLANE:
    memory-mb   : 12288
    vcpus       : 10
    console_port: 8602
    device-type : sriov

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.98
        macaddr   : "0A:00:DD:C0:DE:10"

---
#Interfaces
JUNOS_DEVICES:
   - interface            : ge-0/0/0
     port-speed-mbps      : 10000
     nic                  : eth1
     mtu                  : 2000
     virtual-function     : 0
     mac-address          : "02:06:0A:0E:FF:F0"
     description          : "ge-0/0/0 connects to eth1"

   - interface            : ge-0/0/1
     port-speed-mbps      : 10000
     nic                  : eth2
```

```
    mtu                 : 2000
    virtual-function    : 0
    mac-address         : "02:06:0A:0E:FF:F1"
    description         : "ge-0/0/1 connects to eth2"
```

Installing vMX with Mixed WAN Interfaces

Starting in Junos OS Release 17.2, the use case for mixed WAN interfaces uses the virtio and SR-IOV interfaces. Sample configuration requirements are the same as for using SR-IOV device type.

To install vMX with mixed interfaces:

1. Download the vMX software package as root and uncompress the package.

   `tar xzvf` *`package-name`*

2. Change directory to the location of the uncompressed vMX package.

   `cd` *`package-location`*

3. Edit the **config/vmx.conf** text file with a text editor to configure a single vMX instance.

   Ensure the following parameter is set properly in the vMX configuration file:

   ```
   device-type: mixed
   ```

   When configuring the interfaces, make sure the virtio interfaces are specified before the SR-IOV interfaces. The `type` parameter specifies the interface type.

   See "Specifying vMX Configuration File Parameters" on page 52.

4. Run the `./vmx.sh -lv --install` script to deploy the vMX instance specified by the **config/vmx.conf** startup configuration file and provide verbose-level logging to a file. See "Deploying and Managing vMX" on page 52.

5. From the VCP, enable performance mode for the VFP.

   ```
   user@vmx# set chassis fpc 0 performance-mode
   ```

Here is a sample vMX startup configuration file using mixed interfaces:

```
---
#Configuration on the host side - management interface, VM images etc.
HOST:
    identifier                : vmx1   # Maximum 4 characters
    host-management-interface : eth0
    routing-engine-image      : "/home/vmx/images/junos-vmx-x86-64.qcow2"
    routing-engine-hdd        : "/home/vmx/images/vmxhdd.img"
```

```
    forwarding-engine-image   : "/home/vmx/images/vFPC.img"

---
#External bridge configuration
BRIDGES:
    - type  : external
      name  : br-ext                 # Max 10 characters

---
#VCP VM parameters
CONTROL_PLANE:
    vcpus       : 1
    memory-mb   : 4096
    console_port: 8601

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.94
        macaddr   : "0A:00:DD:C0:DE:0E"

---
#VFP VM parameters
FORWARDING_PLANE:
    memory-mb   : 12288
    vcpus       : 10
    console_port: 8602
    device-type : mixed

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.98
        macaddr   : "0A:00:DD:C0:DE:10"

---
#Interfaces
JUNOS_DEVICES:
   - interface            : ge-0/0/0
     type                 : virtio
     mac-address          : "02:06:0A:0E:FF:F0"
     description          : "ge-0/0/0 interface"

   - interface            : ge-0/0/1
     type                 : sriov
```

```
port-speed-mbps      : 10000
nic                  : eth2
mtu                  : 2000
virtual-function     : 0
mac-address          : "02:06:0A:0E:FF:F1"
description          : "ge-0/0/1 connects to eth2"
```

RELATED DOCUMENTATION

- Minimum Hardware and Software Requirements | 11
- vMX Package Contents | 18
- Deploying and Managing vMX | 52

Deploying and Managing vMX

IN THIS SECTION

Read this topic to understand the procedures required to manage vMX instance after you install it.

Specifying vMX Configuration File Parameters

IN THIS SECTION

The parameters required to configure vMX are defined in the startup configuration file. The configuration file is in YAML format. The default file is **config/vmx.conf**. We recommend you to rename the configuration file to a different name so that you can use the same configuration file every time you create different instances.

NOTE: You must set up these three interfaces to launch the VFP.

- Management access
- Bridge for internal communication between the VCP and VFP
- WAN interface (minimum of one)

Starting in Junos OS Release 18.1, to configure the vMX instance, download and modify the startup configuration file (**vmx.conf**).

1. Download the vMX software package as root and uncompress the package.

   ```
   tar xzvf package-name
   ```

2. Change directory to the location of the uncompressed vMX package.

   ```
   cd package-location
   ```

3. Edit the **config/vmx.conf** text file with a text editor to configure a single vMX instance and save the file.

To customize the configuration, perform these tasks:

Configuring the Host

To configure the host environment, you must change the identifier for each vMX instance and you must provide the correct path for the images.

To configure the host, navigate to **Host** and specify the following parameters:

Parameter	Description
`identifier`	Name of the vMX instance, maximum of four alphanumeric characters.
`host-management-interface`	Name of the physical NIC on the host device that is used for management access (eth0). **NOTE**: The interfaces for HOST:host-management-interface, CONTROL_PLANE, and FORWARDING_PLANE must be on the same subnet.
`routing-engine-image`	Absolute path to the **junos-vmx-x86-64-*.qcow2** file for launching VCP.
`routing-engine-hdd`	Absolute path to the **vmxhdd.img** file for VCP storage.
`forwarding-engine-image`	Absolute path to the **vFPC-*.img** file for launching VFP.
`make-local-copy-of-images`	(Optional) Makes a local copy of the VCP and VFP images and uses the local copy to launch vMX. Default value is `yes`. **NOTE**: Copy the image file from its default location to ensure that the scripts do not try to use the same image file concurrently.
`make-local-copy-of-vmxhdd`	(Optional) Makes a local copy of the VCP storage image and uses the local copy to launch vMX. Default value is `yes`. **NOTE**: Copy the image file from its default location to ensure that the scripts do not try to use the same image file concurrently.

Configuring the VCP VM

To configure the VCP VM, you must change the IP address and you must make sure the console port is not being used by another vMX instance or another server.

To configure the VCP VM, navigate to **CONTROL_PLANE** and specify the following parameters:

NOTE:

Parameter	Description
`vcpus`	Number of vCPUs for the VCP, default is 1. Starting in Junos OS Release 18.1, If you are deploying dual VCP instances, you must double the number of vCPUs.
`memory-mb`	Amount of memory for the VCP, default is 2 GB. In Junos OS Release 15.1F6, amount of memory for the VCP; minimum is 4 GB (performance mode) and 1 GB (lite mode).
`console_port`	KVM TCP-based console port. It must be a unique number.
`deploy`	(Optional) VCP instance to deploy on this host. Specify the number of the instance; first instance is 0, second instance is 1, and multiple instances are separated by a comma. If you do not specify this parameter, both instances (`0,1`) are deployed on this host. If `none` is set, no VCP instance will be deployed on this host. **NOTE**: When deploying on separate hosts, you must set up a connection between the hosts for the VCPs to communicate. Starting in Junos OS Release 18.1 If you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you want to disable VCP for the Control Plane on the server, you have the option to specify `none`.
`console_listen`	(Optional) IP address for the interface from which the console can be accessed; default is 127.0.0.1, which only allows access from within the host. To allow access from any interfaces, specify 0.0.0.0.

(Continued)

Parameter	Description

`instance` (starting in Junos OS Release 18.1)	VCP instance. Navigate to **interfaces > type (static)** and include this parameter below it.
	
	(Optional) Create the second instance below the first instance and include the `console_port` parameter for the second instance. The parameters for specifying both VCP instances might resemble the following:
	
	`interfaces :`
	`- type : static`
	`instance : 0`
	`ipaddr : 10.102.144.94`
	`macaddr : "0A:00:DD:C0:DE:0E"`
	
	`- type : static`
	`instance : 1`
	`console_port: 8612`
	`ipaddr : 10.102.144.95`
	`macaddr : "0A:00:DD:C0:DE:0F"`

`ipaddr`	Management IP address for the VCP VM (fxp0). Navigate to **interfaces > type (static) > ipaddr** to modify this parameter. **NOTE**: The interfaces for HOST:host-management-interface, CONTROL_PLANE, and FORWARDING_PLANE must be on the same subnet.

Configuring the VFP VM

Before you configure the VFP VM, consider the following:

- You must make sure the console port is not being used by another vMX instance or another server.
- To disable network access to the VFP console, do not configure an IP address.
- Based on your requirements, you might want to change the memory, number of vCPUs, and the device type. See "Installing vMX for Different Use Cases" on page 35 for some sample configuration requirements.

NOTE: Starting in Junos OS Release 18.1 if you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you need to disable VFP for the Forwarding Plane on the server, you have the option to specify `none`.

To configure the VFP VM, navigate to **FORWARDING_PLANE** and specify the following parameters:

Parameter	Description
`memory-mb`	Amount of memory for the VFP, default is 6 GB.
`vcpus`	Number of vCPUs for the VFP, default is 3.
`console_port`	KVM TCP-based console port. It must be a unique number.
`deploy`	(Optional) VFP instance to deploy on this host. Specify the number of the instance; first instance is 0, second instance is 1, and multiple instances are separated by a comma. If you do not specify this parameter, both instances (`0,1`) are deployed on this host. If `none` is set, no VFP instance will be deployed on this host. **NOTE**: When deploying on separate hosts, you must set up a connection between the hosts for the VFPs to communicate.
`console_listen`	(Optional) IP address for the interface from which the console can be accessed; default is 127.0.0.1, which only allows access from within the host. To allow access from any interfaces, specify 0.0.0.0.
`device-type`	NIC interface type, either `sriov` or `virtio`. If you are configuring both virtio and SR-IOV interfaces, specify `mixed`.
`ipaddr`	Management IP address for the VFP VM (eth0). Navigate to **interfaces > type (static) > ipaddr** to modify this parameter. **NOTE**: The interfaces for HOST:host-management-interface, CONTROL_PLANE, and FORWARDING_PLANE must be on the same subnet.

(Continued)

Parameter	Description
use_native_drivers	Set to true to allow using the host's driver. **NOTE**: From Junos OS Release 18.4R1 (Ubuntu host) and Junos OS Release 19.1R1 (Red Hat host), you can set the use_native_drivers value to true to use the latest unmodified drivers for your network interface cards for vMX installations.

To configure the VFP VM, navigate to **FORWARDING_PLANE** and specify the following parameters (supported in Junos OS Release 15.1F6):

Parameter	Description
memory-mb	Amount of memory for the VFP; minimum is 12 GB (performance mode) and 4 GB (lite mode).
vcpus	Number of vCPUs for the VFP; minimum is 7 (performance mode) and 3 (lite mode).
console_port	KVM TCP-based console port. It must be a unique number.
console_listen	(Optional) IP address for the interface from which the console can be accessed; default is 127.0.0.1, which only allows access from within the host. To allow access from any interfaces, specify 0.0.0.0.
device-type	NIC interface type, either sriov or virtio.
ipaddr	Management IP address for the VFP VM (eth0). Navigate to **interfaces > type (static) > ipaddr** to modify this parameter. **NOTE**: The interfaces for HOST:host-management-interface, CONTROL_PLANE, and FORWARDING_PLANE must be on the same subnet.

Configuring Interfaces

The JUNOS_DEVICES interface names correspond to the Linux physical NIC names on the host. Bring up the Linux physical NIC ports that are defined in this section before proceeding. For example, use the `ifconfig eth9 up` command to bring up the NIC ports on the eth9 interface.

To configure interfaces for virtio device types, you must specify the interface and the MAC address. You can bind virtio devices to connect virtio NICs in the vMX to physical NICs or virtio NICs in another vMX (see "Binding virtio Devices" on page 66).

To configure interfaces for SR-IOV device types, you must specify the interface, the NIC, and the MAC address.

To configure the routed interfaces, navigate to **JUNOS_DEVICES** and specify the following parameters:

Parameter	Description
`interface`	Name of the interface on the vMX. **NOTE**: The interface names that are defined in the **vmx.conf** file must be contiguous starting from ge-0/0/0. The total number of interfaces supported is 23 for configurations running in performance mode. If you are running virtio interfaces in lite mode, you can use up to 96 interfaces.
`type` (supported in Junos OS Release 17.2 onwards)	NIC interface type, either `sriov` or `virtio`. **NOTE**: If you are configuring both interface types, you must specify the virtio interfaces before the SR-IOV interfaces.
`port-speed-mbps`	(SR-IOV only) Port speed for the physical NIC, default is 10000 Mbps.
`nic`	(SR-IOV only) Name of the physical NIC. **NOTE**: Depending on the version of udev, you can rename the classic Linux standard ethXX names. See Predictable Network Interface Names for more information.
`mtu`	(SR-IOV only) MTU value, default is 2000 and maximum is 9500. To change the MTU configuration for virtio device types, modify the `mtu` parameter in the device binding file (**vmx-junosdev.conf**).

(Continued)

Parameter	Description
`virtual-function`	(SR-IOV only) Child unit of the physical NIC, default is 0. (SR-IOV only) Virtual function number of the physical NIC; default is 0 (supported in Junos OS Release 15.1F5 and earlier releases).
`mac-address`	Unicast MAC address for the physical NIC.
`description`	Description of the mapping.

Release History Table

Release	Description
18.1	Starting in Junos OS Release 18.1 If you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you want to disable VCP for the Control Plane on the server, you have the option to specify none.
18.1	Starting in Junos OS Release 18.1 if you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you need to disable VFP for the Forwarding Plane on the server, you have the option to specify none.

Connecting to VMs

IN THIS SECTION

Perform these tasks to connect to the virtual machines for first-time configuration, to enable access by other means (like Telnet or SSH):

Logging In to VCP

You can access the serial console using the `./vmx.sh --console vcp` *`vmx-id`* command, where *`vmx-id`* is the vMX identifier specified in the startup configuration file, and log in with the username `root` and no password.

To disconnect from the console, log out of the session and press Ctrl +]. At the `telnet>` prompt, type `close` and press Enter.

Logging In to VFP

You can access the serial console using the `./vmx.sh --console vfp vmx-id` command, where `vmx-id` is the vMX identifier specified in the startup configuration file, and log in with the username root and password root.

To disconnect from the console, log out of the session and press Ctrl +]. At the `telnet>` prompt, type `close` and press `Enter`.

To SSH into the virtual forwarding plane (VFP), use the IP address defined under FORWARDING_PLANE in the vmx.conf file. For security reasons, you cannot connect to VFP using the Telnet protocol.

Also for security reasons you cannot connect to the VFP instance using the SSH protocol with the root user. You must first access the VFP with console, login as root user, and create a user that you can then use to SSH in with.

For example:

Access the VFP with the console:

```
root@ubuntu:~/19.2/vmx# ./vmx.sh --console vfp vmx1
```

```
root@qemux86-64:/home/pfe/riot# ./vfp_util.sh -create_user
Enter Username:pfe
Enter Password:
Re-enter Password:
Not copying any file from skel directory into it.
User pfe created, HOME:/var/pfe
Restarting OpenBSD Secure Shell server: sshd.
```

Now when using SSH to access the VFP as the PFE user you can login as super user to access to the root directory.

```
pfe@qemux86-64:~$ su
root@qemux86-64:/var/pfe# id
 uid=0(root) gid=0(root) groups=0(root)
 root@qemux86-64:/var/pfe#
```

Managing vMX

IN THIS SECTION

NOTE: Only English locale is supported for using the vmx.sh script.

After you install and deploy vMX, you can use the **vmx.sh** script with different options to perform these tasks:

Deploying vMX

NOTE: You must be logged in as root to use the control options.

Using the `--install` option also launches the VCP and VFP VMs.

We recommend you deploy the vMX by running the `./vmx.sh -lv --install` script to provide verbose-level logging to a file for the deployment of the vMX instance.

NOTE: Only English locale is supported for using the vmx.sh script.

NOTE: If you cannot deploy vMX after upgrading libvirt, bring down the virbr0 bridge with the `ifconfig virbr0 down` command and delete the bridge with the `brctl delbr virbr0` command.

NOTE: Before you reboot the host server, you must shut down the vMX instance using the `request system halt` command.

To deploy vMX, use these options with the **vmx.sh** script:

`--cfg` *`file`* Use the specified vMX startup configuration file. The default file is **config/vmx.conf**.

`--install` Start vMX by setting up the environment, driver dependencies, and memory requirements and deploying the vMX. If you do not specify a startup configuration file with the `--cfg` option, the default file is used.

NOTE: If you cannot deploy vMX after upgrading libvirt, bring down the virbr0 bridge with the `ifconfig virbr0 down` command and delete the bridge with the `brctl delbr virbr0` command.

This example deploys a new vMX instance specified by the **my-vmx.cfg** configuration file and provides verbose-level logging to a file:

```
./vmx.sh -lv --install --cfg config/my-vmx.cfg
```

Managing vMX Deployments

NOTE: You must be logged in as root to use the control options.

Use these options with the **vmx.sh** script to stop, start, restart, verify, and clean up an existing vMX:

--cfg *file* Use the specified vMX startup configuration file. The default file is **config/vmx.conf**.

--cleanup Stop vMX and clean up relevant information about the vMX instance. It also tears down the Linux bridges and other dependencies. If you do not specify a startup configuration file with the `--cfg` option, the default file is used.

--restart Stop and start a running vMX. This option is useful for redeploying a vMX that has parameter changes in the startup configuration file. If you do not specify a startup configuration file with the `--cfg` option, the default file is used.

--start Start the vMX that was running and stopped. If you do not specify a startup configuration file with the `--cfg` option, the default file is used.

--status Verify the status of a deployed vMX. If you do not specify a startup configuration file with the `--cfg` option, the default file is used.

--stop Stop vMX without cleaning up build files so that the vMX can be started quickly without setup performed by the `--install` option.

This example tears down an existing vMX instance specified by the **my-vmx.cfg** configuration file:

```
./vmx.sh --cleanup --cfg config/my-vmx.cfg
```

Starting in Junos OS release 19.1 onwards, if you are deploying the vMX image with i40e driver-based NIC cards and want to redeploy the vMX that has parameter changes in the startup configuration file, we recommend not using the options such as --restart or --start/--stop. You must use the following options:

1. Use the `./vmx.sh --cleanup` command to clean up an existing vMX.
2. Run the `./vmx.sh -lv --install` script to re-deploy vMX.

The vMX instance starts with the updated configuration.

Specifying the Temporary File Directory

NOTE: You must be logged in as root to use the control options.

To specify the directory used for temporary files, run the `./vmx.sh build` *directory* script. The default directory is **build/*vmx-id***, where ***vmx-id*** is the vMX identifier specified in the startup configuration file.

By default, copies of the VCP and VFP images are copied to this directory. We recommend that you do not change the `make-local-copy-of-images` and `make-local-copy-of-vmxhdd` parameters when specifying startup configuration file parameters for the host.

Specifying the Environment File

NOTE: You must be logged in as root to use the control options.

To specify the environment file (**.env**), run the `./vmx.sh env` *file* script. The default file is **env/ubuntu_sriov.env**.

Configuring Logging Options for vMX

You can enable logging options. It is especially useful when used with the control options, such as `--install`.

Use these options to configure logging:

`-l` Enable logging to a file in the specified build directory. The default directory is **build/*vmx-id*/logs**, where ***vmx-id*** is the vMX identifier specified in the startup configuration file. By default, logging is disabled.

`-lv` Enable logging with verbose details.

`-lvf` Enable logging with verbose details to the foreground (standard output).

This example deploys a new vMX instance specified by the **my-vmx.cfg** configuration file and provides verbose-level logging to a file:

```
./vmx.sh -lv --install --cfg config/my-vmx.cfg
```

Connecting to Console Port for the VMs

Use these options with the **vmx.sh** script to connect to the console of the VCP or VFP of the specified vMX:

`--console vcp [`*`vmx-id`*`]` Connect to the console of the VCP for the specified vMX. The vMX identifier is specified in the startup configuration file.

`--console vfp [`*`vmx-id`*`]` Connect to the console of the VFP for the specified vMX. The vMX identifier is specified in the startup configuration file.

This example connects to the console of the VCP for the vMX instance specified by the vmx1 identifier:

```
./vmx.sh --console vcp vmx1
```

Getting Help for the Script Options

To obtain on-line help for the **vmx.sh** script options, run the `./vmx.sh --help` script.

Binding virtio Devices

IN THIS SECTION

For configurations using virtio device types, you can bind multiple vMX instances together on the same system if the host has enough CPU and memory to support the vMX instances. You configure each vMX instance with a different startup configuration file.

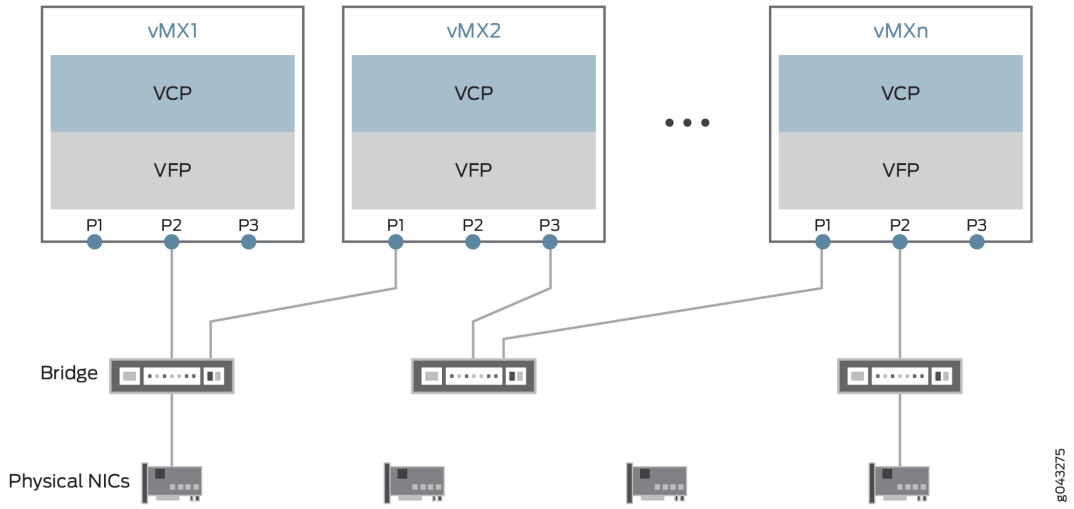
If you are deploying multiple vMX instances, make sure:

- The VM identifiers are unique across all instances.
- The console ports of the VCP and the VFP are unique across all instances.
- The external static IP address of the VCP and the VFP are unique across all instances.
- The MAC addresses of the VCP and the VFP are unique across all instances, whenever specified.

NOTE: All VMs share the same management domain. The physical management interface (for example, eth0) is also part of this global external bridge.

You can connect virtio NICs in the vMX to physical NICs or virtio NICs in another vMX by binding these devices as shown in Figure 3 on page 67.

Figure 3: Binding Devices



To manage device bindings, perform these tasks:

Setting Up the Device Bindings

The parameters required to configure vMX to bind devices are defined in the device-binding file. The device-binding file is in YAML format. The default file is **config/vmx-junosdev.conf**.

The device-binding file defines the endpoints of each link originating from the VFP of a vMX. One endpoint must be a device using virtio NICs. The other endpoint can be a physical NIC, a virtio NIC in another vMX instance, or a Linux bridge.

To bind the vMX instances together:

1. Edit the **config/vmx-junosdev.conf** file to set up the communication between the vMX instances.
2. Modify the `link_name` to the name of the Linux bridge (as shown by the `brctl show` command). The link name can be 15 characters long. It must be unique for each bridge. If more than two interfaces (virtual or physical) are connected by a Linux bridge, then the bridge name is derived from the `dev_name` of the common endpoint for the connected devices.
3. Specify the `mtu` to change the MTU value for virtio device types from the default of 1500. The maximum value is 9500.

   To change the MTU configuration for SR-IOV device types, modify the `mtu` parameter in the startup configuration file (**vmx.conf**).
4. Specify the endpoints for vMX devices (`junos_dev` type) by customizing these parameters:
   - `type`—Type of device is `junos_dev`.

- vm-name—Name of the vMX identifier specified in the startup configuration file for that vMX instance.
- dev-name—Name of the interface on vMX as specified in the startup configuration file.

5. Specify the endpoints for physical NICs (host_dev type) by customizing these parameters:
   - type—Type of device is host_dev.
   - dev-name—Name of the physical NIC on the host.
6. Specify the endpoints for bridges (bridge_dev type) by customizing these parameters:
   - type—Type of device is bridge_dev.
   - dev-name—Name of the Linux bridge.
7. If you have multiple device-binding files, save them with different names.

Here is a sample vMX device-binding file:

```
interfaces :

     - link_name  : link_host
       mtu        : 1500
       endpoint_1 :
         - type        : junos_dev
           vm_name     : vmx1
           dev_name    : ge-0/0/0
       endpoint_2 :
         - type        : host_dev
           dev_name    : int2

     - link_name  : link_vmx_12
       mtu        : 1500
       endpoint_1 :
         - type        : junos_dev
           vm_name     : vmx1
           dev_name    : ge-0/0/1
       endpoint_2 :
         - type        : junos_dev
           vm_name     : vmx2
           dev_name    : ge-0/0/0

     - link_name  : bridge_vmx_123
       endpoint_1 :
         - type        : junos_dev
```

```
            vm_name     : vmx1
            dev_name    : ge-0/0/2
        endpoint_2 :
          - type        : bridge_dev
            dev_name    : bridge1

      - link_name   : bridge_vmx_123
        endpoint_1 :
          - type        : junos_dev
            vm_name     : vmx2
            dev_name    : ge-0/0/1
        endpoint_2 :
          - type        : bridge_dev
            dev_name    : bridge1

      - link_name   : bridge_vmx_123
        endpoint_1 :
          - type        : junos_dev
            vm_name     : vmx3
            dev_name    : ge-0/0/0
        endpoint_2 :
          - type        : bridge_dev
            dev_name    : bridge1
```

The `link_host` entry shows how to connect the ge-0/0/0 interface on vmx1 to the physical NIC. The `link_vmx_12` entry shows how to connect two interfaces on vmx1 and vmx2 to each other. The `bridge_vmx_123` entries show how to connect the interfaces on vmx1, vmx2, and vmx3 with a bridge.

Creating Device Bindings

NOTE: You must be logged in as root to bind devices.

To bind devices with virtio NICs to other devices, define your devices in the vMX device-binding file and run the `./vmx.sh --bind-dev --cfg` *`device-binding-file`* script to create the device binding. If you do not specify a file, the default file is **config/vmx-junosdev.conf**.

This example creates device bindings with the specified device-binding file:

```
./vmx.sh --bind-dev --cfg config/vmx1-junosdev.conf
```

Deleting Device Bindings

NOTE: You must be logged in as root to unbind devices.

To unbind devices, run the `./vmx.sh --unbind-dev --cfg` *`device-binding-file`* script to delete the device bindings created with the `--bind-dev` option. If you do not specify a file, the default file is **config/vmx-junosdev.conf**.

This example deletes device bindings with the specified device-binding file:

```
./vmx.sh --unbind-dev --cfg config/vmx1-junosdev.conf
```

Verifying Device Bindings

NOTE: You must be logged in as root to bind devices.

To verify the status of device bindings created with the `--bind-dev` option, run the `./vmx.sh --bind-check --cfg` *`device-binding-file`* script. If you do not specify a file, the default file is **config/vmx-junosdev.conf**.

This example verifies the status of the device bindings for the specified device-binding file:

```
./vmx.sh --bind-check --cfg config/vmx1-junosdev.conf
```

Release History Table

Release	Description
18.1	Starting in Junos OS Release 18.1 If you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you want to disable VCP for the Control Plane on the server, you have the option to specify none.
18.1	Starting in Junos OS Release 18.1 if you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you need to disable VFP for the Forwarding Plane on the server, you have the option to specify none.
18.1	Starting in Junos OS Release 18.1 If you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you want to disable VCP for the Control Plane on the server, you have the option to specify none.

18.1	Starting in Junos OS Release 18.1 if you are deploying across multiple servers (for example, one server as the RE and one server as the PFE), and you need to disable VFP for the Forwarding Plane on the server, you have the option to specify none.

RELATED DOCUMENTATION

- Minimum Hardware and Software Requirements | 11
- vMX Package Contents | 18
- Installing vMX on KVM | 20
- Installing Nested vMX VMs | 71

Installing Nested vMX VMs

IN THIS SECTION

A nested virtual machine is a virtual machine contained within another VM. Read this topic to understand how to launch the nested vMX VM on KVM.

Overview of the Nested VM Model

IN THIS SECTION

The nested vMX virtual machine (VM) model has the virtual control plane (VCP) running as a VM within the virtual forwarding plane (VFP) VM. The VFP VM runs the virtual Trio forwarding plane software and the VCP VM runs Junos OS. The VCP VM and VFP VM require Layer 2 connectivity to communicate with each other. An *internal* bridge that is local to the server for each vMX instance enables this communication. The VCP VM and VFP VM also require Layer 2 connectivity to communicate with the Ethernet management port on the server. You must specify virtual Ethernet interfaces with unique IP addresses and MAC addresses for both the VFP and VCP to set up an *external* bridge for a vMX instance. Ethernet management traffic for all vMX instances enters the server through the Ethernet management port.

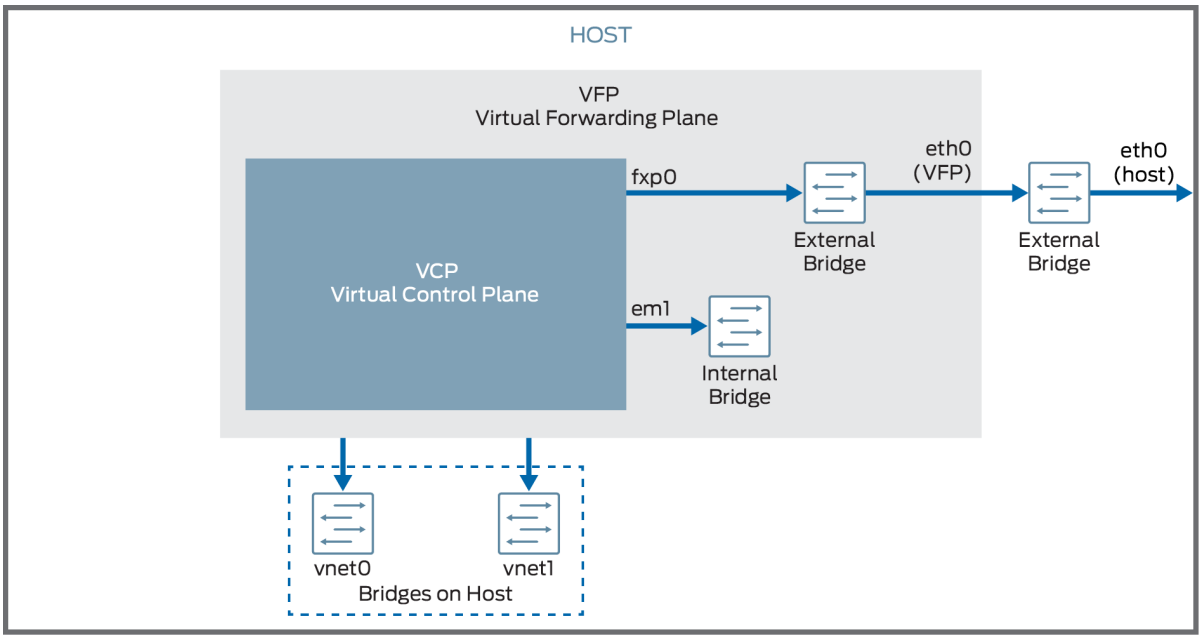
The nested vMX VM supports virtio and SR-IOV interfaces for forwarding ports. The first interface is used for management and must be a virtio interface connected to the br-ext bridge (external bridge). Subsequent interfaces are WAN interfaces and can be virtio or SR-IOV interfaces. You must create the bridges for all the virtio interfaces. You must have at least one WAN interface for forwarding.

Nested VM with Virtio Interfaces

In virtio mode, the server interfaces must not be configured with the VFs. You can remove or reset the interfaces (`eth1`) using the `rmmod ixgbe` command and you can add the IXGBE driver with default interface to the server interface using the `modprobe ixgbe` command.

Figure 4 on page 73 illustrates the nested vMX VM model with virtio interfaces.

Figure 4: Nested VM with virtio Interfaces



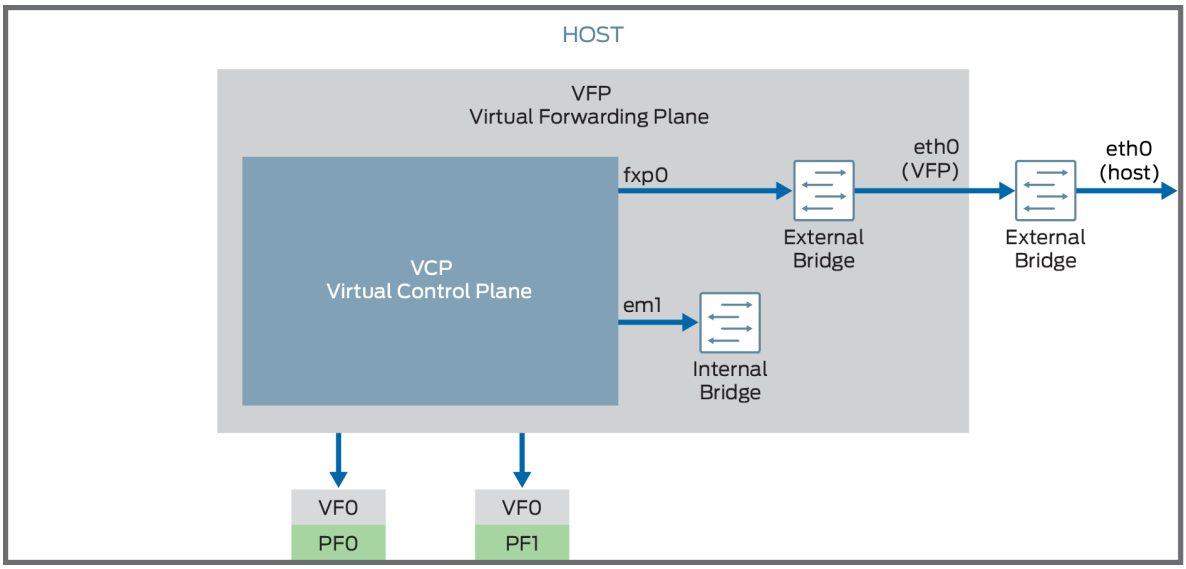
Nested VM with SR-IOV Interfaces

In SR-IOV mode, the vMX interfaces are associated with the server interfaces. For example, the `ge-0/0/0` interface is associated with `eth1` . `eth1` is defined in the `.conf file- interface: ge-0/0/0 ,nic: eth1`.

The VF is added to the IXGBE driver of the server interface `eth1` which associated with the VF and can be checked using the `ip link show eth1` command while running in the SR-IOV mode.

Figure 5 on page 74 illustrates the nested vMX VM model with SR-IOV interfaces.

Figure 5: Nested VM with SR-IOV Interfaces



For SR-IOV interfaces, you must load the modified IXGBE driver before launching the nested vMX VM.

The way network traffic passes from the physical NIC to the virtual NIC depends on the virtualization technique that you configure.

System Requirements for Nested VM Model

vMX can be configured to run in two modes depending on the use case:

- Lite mode—Needs fewer resources in terms of CPU and memory to run at lower bandwidth.
- Performance mode—Needs higher resources in terms of CPU and memory to run at higher bandwidth.

> **NOTE**: Performance mode is the default mode.

vMX Limitations with the Nested VM Model

vMX does not support the following features with the nested VM model:

- Attachment or detachment of interfaces while a vMX instance is running
- Upgrade of Junos OS release

Hardware and Software Requirements for Nested vMX VMs

Table 13 on page 75 lists the hardware requirements.

Table 13: Minimum Hardware Requirements for the Nested vMX VM

Description	Value
Sample system configuration	For virtio: Any x86 processor (Intel or AMD) with VT-d capability. For SR-IOV: Intel 82599-based PCI-Express cards (10 Gbps) and Ivy Bridge processors.
Number of cores **NOTE**: Performance mode is the default mode and the minimum value is based on one port.	• For lite mode: Minimum of 4 vCPUs **NOTE**: If you want to use lite mode when you are running with more than 4 vCPUs for the VFP, you must explicitly configure lite mode. • For performance mode: Minimum of 8 vCPUs **NOTE**: To calculate the recommended number of vCPUs needed by VFP for performance mode: (3 * *number-of-forwarding-ports*) + 4

Memory	• For lite mode: Minimum of 3 GB
	• For performance mode:
	  • Minimum of 5 GB
	  • Recommended of 16 GB

Table 14 on page 75 lists the software requirements.

Table 14: Software Requirements for Ubuntu

Description	Value
Operating system	Ubuntu 14.04.1 LTS Linux 3.19.0-80-generic

Table 14: Software Requirements for Ubuntu *(Continued)*

Description	Value
Virtualization	QEMU-KVM 2.0.0+dfsg-2ubuntu1.11
Required packages **NOTE**: Other additional packages might be required to satisfy all dependencies.	bridge-utils qemu-kvm libvirt-bin virtinst **NOTE**: libvirt 1.2.19

Installing and Launching the Nested vMX VM on KVM

IN THIS SECTION

To launch the nested vMX VM on KVM, perform these tasks.

Preparing the Ubuntu Host to Install the Nested vMX VM

To prepare the Ubuntu host system for installing vMX:

1. Meet the software and OS requirements described in "Hardware and Software Requirements for Nested vMX VMs" on page 75.
2. Enable Intel VT-d in BIOS. (We recommend that you verify the process with the vendor because different systems have different methods to enable VT-d.)

   Refer to the procedure to enable VT-d available on the Intel Website.
3. Disable KSM by setting `KSM_ENABLED=0` in **/etc/default/qemu-kvm**.

4. Disable APIC virtualization by editing the **/etc/modprobe.d/qemu-system-x86.conf** file and adding `enable_apicv=0` to the line containing `options kvm_intel`.

   ```
   options kvm_intel nested=1 enable_apicv=0
   ```

5. Restart the host to disable KSM and APIC virtualization.
6. If you are using SR-IOV, you must perform this step.

   > **NOTE**: You must remove any previous installation with an external bridge in **/etc/network/interfaces** and revert to using the original management interface. Make sure that the `ifconfig -a` command does not show external bridges before you proceed with the installation.
   >
   > To determine whether an external bridge is displayed, use the `ifconfig` command to see the management interface. To confirm that this interface is used for an external bridge group, use the `brctl show` command to see whether the management interface is listed as an external bridge.

   Enable SR-IOV capability by turning on `intel_iommu=on` in the **/etc/default/grub** directory.

   ```
   GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on"
   ```

   Append the `intel_iommu=on` string to any existing text for the `GRUB_CMDLINE_LINUX_DEFAULT` parameter.
7. For optimal performance, we recommend you configure the size of Huge Pages to be 1G on the host and make sure the NUMA node for the VFP has at least 16 1G Huge Pages. To configure the size of Huge Pages, add the following line in **/etc/default/grub**:

   ```
   GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=number-of-huge-pages"
   ```

   The number of Huge Pages must be at least (16G * *number-of-numa-sockets*).
8. Run the `update-grub` command followed by the `reboot` command.
9. Run the `modprobe kvm-intel` command before you install vMX.

Loading the Modified IXGBE Driver

If you are using SR-IOV interfaces, you must load the modified IXGBE driver before launching the nested vMX VM. To load the modified IXGBE driver:

1. Download the vMX KVM software package and uncompress the package.

   ```
   tar xvf package-name
   ```

2. Before compiling the driver, make sure gcc and make are installed.

```
sudo apt-get update
sudo apt-get install make gcc
```

3. Unload the default IXGBE driver, compile the modified Juniper Networks driver, and load the modified IXGBE driver.

```
cd package-location/drivers/ixgbe-3.19.1/src
make
sudo rmmod ixgbe
sudo insmod ./ixgbe.ko max_vfs=1,1
sudo make install
```

4. Verify the driver version (3.19.1) on the SR-IOV interfaces.

Launching a Nested vMX Instance

To launch the nested vMX instance:

1. Download the vMX Nested software package.
2. Convert the `vmdk` image to `qcow2` format.

```
qemu-img convert -f vmdk -O qcow2 vmdk-filename qcow2-filename
```

3. Create the bridges for the virtio interfaces.

```
brctl addbr  bridge-name
```

NOTE: When you create a bridge using the `brctl addbr <bridge-name>` command, the server might lose the connection. Alternatively, you can spawn the vMX in unnested mode (either in SRIOV or virtio mode) and use the `virsh destroy vcp` *vcp-name* and `virsh destroy vfp` *vfp-name* commands to create and retain the bridge.

NOTE: You must create the bridges for the virtio interfaces before you launch the nested vMX instance.

4. Launch the nested vMX VM instance with the `virt-install` command. For example:

```
sudo virt-install --hvm --vcpus=number-vcpus -r memory \
    --serial tcp,host=:console-port,mode=bind,protocol=telnet \
    --nographics --import --noautoconsole \
    --cpu \
SandyBridge,+erms,+smep,+fsgsbase,+pdpe1gb,+rdrand,+f16c,+osxsave,+dca,+pcid,+pdcm,+x
tpr,+tm2,+est,+smx,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme \
    -w bridge=br-ext,model=virtio \
    -w bridge=bridge-name,model=virtio \
    --host-device=pci-id \
    -n name --disk disk-image,format=qcow2
```

where:

- `--vcpus`—Specifies the number of vCPUs.

  For lite mode, minimum of 4 vCPUs. For performance mode, minimum of [(4 * *number-of-forwarding-ports*) + 4] vCPUs.

- `-r`—Specifies the amount of memory the VM uses in MB. Minimum of 16 GB.

- `--serial`—Specifies the serial port for the VFP.

- `-w`—Specifies the virtio interface. The first interface is used for management and is connected to the br-ext bridge. Subsequent interfaces are WAN interfaces and are connected to the bridges on the host.

- `--host-device`—Specifies the SR-IOV interface as the PCI ID of the virtual function (VF0).

  To determine the PCI ID:

  a. Use the `ip link` command to obtain the interface names for which you create VFs that are bound to the vMX instance.

  b. Use the `ethtool -i` *`interface-name`* utility to determine the PCI bus information.

```
driver: ixgbe
version: 3.19.1
firmware-version: 0x61bd0001
bus-info: 0000:81:00.0
supports-statistics: yes
```

```
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

c. Use the `virsh nodedev-list` command to obtain the VF PCI ID.

```
pci_0000_81_00_0
pci_0000_81_00_1
pci_0000_81_10_0
pci_0000_81_10_1
```

- `-n`—Specifies the name of the vMX VM.
- `--disk`—Specifies the path to the `qcow2` file (**vmx-nested-*release*.qcow2**).

For example, this command launches a vMX instance in performance mode with two virtio interfaces connected to the vnet0 and vnet1 bridges:

```
sudo virt-install --hvm --vcpus=12 -r 16384 \
    --serial tcp,host=:4001,mode=bind,protocol=telnet \
    --nographics --import --noautoconsole \
    --cpu \
SandyBridge,+erms,+smep,+fsgsbase,+pdpe1gb,+rdrand,+f16c,+osxsave,+dca,+pcid,+pdcm,+x
tpr,+tm2,+est,+smx,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme \
    -w bridge=br-ext,model=virtio \
    -w bridge=vnet0,model=virtio \
    -w bridge=vnet1,model=virtio \
    -n vmx1 --disk vmx-nested-17.2R1.13-4.qcow2,format=qcow2
```

For example, this command launches a vMX instance in performance mode with two SR-IOV interfaces:

```
sudo virt-install --hvm --vcpus=12 -r 16384 \
    --serial tcp,host=:4001,mode=bind,protocol=telnet \
    --nographics --import --noautoconsole \
    --cpu \
SandyBridge,+erms,+smep,+fsgsbase,+pdpe1gb,+rdrand,+f16c,+osxsave,+dca,+pcid,+pdcm,+x
tpr,+tm2,+est,+smx,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme \
    -w bridge=br-ext,model=virtio \
    --host-device=pci_0000_81_10_0 \
```

```
    --host-device=pci_0000_81_10_1 \
    -n vmx2 --disk vmx-nested-17.2R1.13-4.qcow2,format=qcow2
```

Connecting to the VFP Console Port

After launching the vMX instance with the `virt-install` command, you can connect to the console port of the VFP from the host with the `telnet localhost` *`serial-port`* command, where *`serial-port`* is the port you specified as `host` with the `-serial` parameter.

For example:

```
$ telnet localhost 4001
```

Log in with the default username `jnpr` and password `jnpr123`. Become root using the `sudo -i` command.

The br-ext interface tries to fetch an IP address using DHCP. Use the `ifconfig br-ext` command to display the assigned IP address. If DHCP is unavailable or if you prefer a static IP address, assign an IP address to `br-ext`. You can now connect to the VFP using the SSH protocol and this assigned IP address.

Connecting to the VCP

When the VCP VM is launched, you can connect to the VCP console port at TCP port 8601 from the VFP VM using this command:

```
$ telnet localhost 8601
```

From the console port, you can log in with username `root` and no password.

At a minimum, you must perform these initial Junos OS configuration tasks after logging in to the VCP:

1. Start the CLI.

```
root@% cli
root@>
```

2. Enter configuration mode.

```
root@> configure
```

```
[edit]
root@#
```

3. Configure the root password.

```
[edit]
root@# set system root-authentication plain-text-password
New password: password
Retype new password: password
```

4. Configure the IP address and prefix length for the router's management Ethernet interface.

```
[edit]
root@# set interfaces fxp0 unit 0 family inet address address/prefix-length
```

5. Commit the configuration.

```
[edit]
root@# commit
```

RELATED DOCUMENTATION

Installing vMX on KVM | 20

Deploying and Managing vMX | 52

Example: Enabling SR-IOV on vMX Instances on KVM

IN THIS SECTION

vMX on KVM supports single-root I/O virtualization (SR-IOV) interface types. Single root I/O virtualization (SR-IOV) allows a physical function to appear as multiple, separate vNICs. SR-IOV allows a device, such as a network adapter to have separate access to its resources among various hardware functions. If you have a physical NIC that supports SR-IOV, you can attach SR-IOV-enabled vNICs or virtual functions (VFs) to the vMX instance to improve performance.

System requirements:

- Junos OS Release 18.4 or later.
- SR-IOV on the VMX for KVM requires one of the following Intel NIC drivers:
  - Intel X520 or X540 using 10G ports and ixgbe driver
  - Intel X710 or XL710 using 10G ports and i40e driver

  Starting in Junos OS Release 19.1R1-S1 and in Junos OS Release 19.2R1, support for 40G ports with Intel XL710-QDA2 NICs are available for VMX instances. When using 40G ports, the vMX autodetects the port speed and assigns two I/O vCPUs.

To enable SR-IOV on VMX instances, you must complete the following tasks:

- Prepare a NIC to use SR-IOV in system (/sys/) mode.

- Install driver from Intel, you must compile the driver, uninstall old driver, and install new compiled driver

> **NOTE:** The vMX installer provides a modified intel-driver as well. You can either use the native drivers from Intel, or use vMX modified driver.

- Prepare vmx.conf file
- Use Junos CLI to configure native driver
- BIOS requirement to enable SR-IOV- Ensure that Intel VT-d or AMD IOMMU are enabled in the system's BIOS settings.

Procedure for Identifying PCI-Addresses and Kernel Name for the NIC

1. To find the PCI address, use the following command:

```
lab@ubuntu2:/etc/modprobe.d$ ethtool -i ens8f1 | grep bus
```

```
bus-info: 0000:85:00.1
```

2. To find the kernel name using PCI, use the following command:

```
lab@ubuntu2:~$ cd /sys/bus/pci/devices
lab@ubuntu2:/sys/bus/pci/devices$ ls 0000\:85\:00.1/net/
```

```
ens8f1
```

3. To find out the driver in use for the NIC, use the following command:

```
lab@ubuntu2:~$ ethtool -i ens8f1 | grep ^driver
```

```
driver: ixgbe
```

Download and Install the Latest Driver Software from Intel

You can download the latest driver software from Intel and replace existing driver software provided by Ubuntu.

In this example, download the software from Intel® Network Adapter Driver for PCIe* Intel® 10 Gigabit Ethernet Network Connections Under Linux and save it into any directory of your choice and follow the README instructions to proceed next.

To install driver software from Intel:

1. Install the driver software.

```
cd ~/intel_ixgbe/ixgbe-5.5.3/src
sudo make install
```

2. Uninstall the old driver and load the updated driver by using the `rmmod/modprobe` command.

```
sudo rmmod ixgbe
sudo modprobe ixgbe
```

WARNING: The command `rmmod` uninstalls the 10GE driver. If this is the only interface you are connected to, then access to the host will be lost.

3. Verify if the new driver is installed correctly.

```
lab@ubuntu2:~/intel_ixgbe/ixgbe-5.5.3/src$ modinfo ixgbe | grep -i version
version:        5.5.3
```

Prepare NIC to Use SR-IOV in System Mode

The host needs to be informed for each dedicated NIC by setting the *sriov_numvfs* value, how many VFs are going to use SR-IOV for the given NIC. The vmx.sh script have no information of how many VFs will use the shared NIC. Because of this, you must configure the *sriov_numvfs* accordingly.

This value can be set as a boot-option to be persistent after a reboot and can be changed on-the-fly which would not be persistent after a reboot.

The procedure given in this example is temporary solution for configuring *sriov_numvfs* using **/sys** Any setting to /sys/class/net/*interface-name*/device/*sriov_numvfs* is non-permanent, hence the configuration does not survive a reboot.

To prepare NIC to use SR-IOV, complete the following steps:

Create a virtual function (VF) using the following command:

```
# echo num_of_vf > /sys/class/net/interface-name/device/sriov_numvfs
```

Below command allows 4 VNFs to use shared NIC ens8f1 for SR-IOV. You must either use sudo or need login as root user.

```
As sudo user:
root@ubuntu2:~# echo 4 | sudo tee -a  /sys/class/net/ens8f1/device/sriov_numvfs
```

```
As root user
root@ubuntu2:~# echo 4 > /sys/class/net/ens8f1/device/sriov_numvfs
```

NOTE: The `sriov_numvfs` option only accepts values 0-n, where n is the maximum number of VFs that are supported by the SR-IOV.

Setting SR-IOV at Boot-Time

The following procedures provide some alternate methods for configuring SR-IOV where the configuration persists a reboot of the host.

Following options are available to set the value during the boot-process of the host:

- Using rc.local
- Setting modprobe options
- Setting kernel-paramater using grub

Below example shows a method to configure the *sriov_numvfs* value by using grub kernel command

- You must set "intel_iommu=on" and ixgbe.max_vfs= *value*

For more information on hugepages, see Preparing the Ubuntu Host to Install vMX.

Edit the file “/etc/default/grub”:

```
lab@ubuntu2:~$ cat /etc/default/grub | grep -i cmd
GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on"
GRUB_CMDLINE_LINUX="isolcpus=34-41,48-55 default_hugepagesz=1G
hugepagesz=1G hugepages=120 ixgbe.max_vfs=8"
```

After editing, update the following:

```
sudo update-grub
```

Write new boot-loader to make changes active upon next reboot.

```
sudo grub-install /dev/sda
```

Reboot the host to make settings active.

```
sudo reboot
```

Verify sriov_numvfs Settings

IN THIS SECTION

Purpose

To verify the *sriov_numvfs* configuration using the CLI. In this example, the required NIC to use with SR-IOV is ens8f1 at PCI-address 85:00.0. Please note the “Virtual Function” in the output.

Action

```
lab@ubuntu2:~$ lspci | grep 85
```

```
85:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network
Connection (rev 01)
85:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network
Connection (rev 01)
85:10.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.1 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.5 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.6 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:10.7 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.1 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.5 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.6 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
85:11.7 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
```

```
Function (rev 01)
```

The /sys/class/net/ens8f1/device/sriov_numvfs also contains the desired value of sriov_numvfs.

```
lab@ubuntu2:~$ cat /sys/class/net/ens8f1/device/sriov_numvfs
```

```
8
```

For testing, the sriov_numvfs can be changed quickly by directly writing into /sys/class/net/interface-name/device/sriov_numvfs.

Changing the Number of sriov_numvfs

We recommend creating sriov_numvfs in advance (example: by using grub command at boot-time), because, changing the VF's number is not allowed after deploying 1st vMX instance which uses given NIC with SR-IOV. If you must change the VF's number, then you must stop the running vMX. We recommend to set the sriov_numvfs option to a higher value to avoid changing the sriov_numvfs number afterwards.

To change an already configured value of VFs, you must first change it's value as 0, and then change it to required integer value.

If there are no VFs assigned, the number of VFs can be changed to any valid value (0 - n, where n is the maximum number of VFs that are supported by the SR-IOV)

You must perform the following steps to modify the number of VFs:

1. Stop running VNFs using the shared SR-IOV NIC.
2. Disable the SR-IOV network adapter by setting the number of Virtual Functors (VFs) to 0.

```
As root:
root@ubuntu2:~# echo 0 > /sys/class/net/ens8f1/device/sriov_numvfs
```

```
As sudo:
lab@ubuntu2:~$ echo 0  | sudo tee -a /sys/class/net/ens8f1/device/sriov_numvfs
```

3. Change the required number of VF (you are using six in this example)

```
root@ubuntu2:~# echo 6 > /sys/class/net/ens8f1/device/sriov_numvfs
```

NOTE: If you see the following error message, then first set the value to zero as described in step 2 before performing step 3

```
root@ubuntu2:~# echo 6 > /sys/class/net/ens8f1/device/sriov_numvfs bash: echo: write
error: Device or resource busy
```

4. Verify your configuration by using the following command:

```
root@ubuntu2:~# cat /sys/class/net/ens8f1/device/sriov_numvfs
```

Before restarting the vMX, adopt the vmx.conf file for SR-IOV usage.

Updating the VMX Configuration File (vmx.conf) Parameters

The parameters required to configure vMX are defined in the startup configuration file. The configuration file is in YAML format. The default file is config/vmx.conf. You can save your configuration file to a different name for different instances.

To configure interfaces for SR-IOV device types, you must specify the interface, the NIC, and the MAC address. Table 15 on page 90 provides the details of the configuration parameters that we are using to change vmx.conf file.

Table 15: VMX Configuration File Parameters

Components	VM Parameters	Description
vPFE	`device-type`	Use `sriov` for all interfaces using the SR-IOV or use `mixed` to allow mixing of SR-IOV and non-SR-IOV-based interfaces.

Table 15: VMX Configuration File Parameters *(Continued)*

Components	VM Parameters	Description
	`use_native_drivers`	Set to `true` to allow using the host's Intel ixgbe driver (which was downloaded and complied in above steps)
Interfaces	`type`	If type is set to `sriov`, then `port-speed-mbps` and `nic` must be set.
	`port-speed-mbps`	Set it to `10000` for 10GE NIC.
	`nic`	The kernel-name for the interface to use.
	`virtual-function`	Set to `0` for first vMX instance using this NIC. Ensure to set to `1` for 2nd vMX using this shared NIC (and so on)
	`mac-address`	Ensure that each VF instance using the shared NIC is using a unique or different MAC address.

A sample vmx.conf file:

```
lab@ubuntu2:~/vmx/config$ cat vmx.conf.sriov
```

```
##############################################################
#
#  vmx.conf
#  Config file for vmx on the hypervisor.
#  Uses YAML syntax.
#  Leave a space after ":" to specify the parameter value.
#
##############################################################

---
```

```
#Configuration on the host side - management interface, VM images etc.
HOST:
    identifier               : vmx1   # Maximum 6 characters
    host-management-interface : ens4f0
    routing-engine-image      : "/home/lab/vmx/images/junos-vmx-x86-64-18.1R3-S2.5.qcow2"
    routing-engine-hdd        : "/home/lab/vmx/images/vmxhdd.img"
    forwarding-engine-image   : "/home/lab/vmx/images/vFPC-20181023.img"

---
#External bridge configuration
BRIDGES:
    - type  : external
      name  : br-ext                 # Max 10 characters

---
#vRE VM parameters
CONTROL_PLANE:
    vcpus       : 1
    memory-mb   : 1024
    console_port: 8601

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.94
        macaddr   : "0A:00:DD:C0:DE:0E"

---
#vPFE VM parameters
FORWARDING_PLANE:
    memory-mb   : 8192
    vcpus       : 4
    console_port: 8602
    device-type : mixed                     <<<< sriov or mixed
    use_native_drivers : true               <<<< use drivers as provided by the host

    interfaces  :
      - type      : static
        ipaddr    : 10.102.144.98
        macaddr   : "0A:00:DD:C0:DE:10"

---
#Interfaces
JUNOS_DEVICES:
```

```
   - interface            : ge-0/0/0
     type                 : virtio           <<< required
     mac-address          : "02:06:0A:0E:FF:F0"
     description          : "ge-0/0/0 interface"

   - interface            : ge-0/0/1
     type                 : sriov          <<< required
     mtu                  : 9192           <<< required
     port-speed-mbps      : 10000          <<< required
     nic                  : ens8f1         <<< required
     virtual-function     : 0              <<< use consecutive / unique numbers for each vMX
instance
     mac-address          : "02:06:0A:0E:FF:F1"  <<< make sure that each vNF is using a
DIFFERENT MAC-address
     description          : "ge-0/0/1 interface"
```

Start vmx-install

```
lab@ubuntu2:~/vmx$ sudo ./vmx.sh --install --cfg ./config/vmx.conf.sriov
```

Changes Required for Using Intel ixgbe Driver

When you try to move an existing deployment from modified IXGBE driver to unmodified IXGBE driver, enter edit mode in Junos CLI and use the following command when using "native" drivers.

```
user@host# set interfaces vlan-offload
```

For more information, see "Modified and Unmodified IXGBE Driver" on page 100.

RELATED DOCUMENTATION

Modified and Unmodified IXGBE Driver | 100

Minimum Hardware and Software Requirements | 11

3
CHAPTER

Configuring Modified and Unmodified Drivers

Modified and Unmodified i40e Driver

IN THIS SECTION

Read this topic to understand modified and unmodified i40e driver support for vMX instances.

Understanding the Differences between Modified and Unmodified i40e Driver

The single root I/O virtualization (SR-IOV) functionality consists of a physical function (PF) driver and a virtual function (VF) driver. The PF driver of an SR-IOV device is used to manage the physical function of an SR-IOV capable device. A VF driver of an SR-IOV device shares one or more physical resources with the physical function and other virtual functions that are associated with the same physical function.

In the modified i40e driver the physical function sets the port to the MAC promiscuous and VLAN promiscuous mode. In this case, all the frames associated with the port is passed to the single VF which is associated with the vMX. A single VF instance might be supported on a PF and the total number of VLANs per IFD is limited to 64 if the `vlan-offload` option is configured.

In the unmodified i40e driver, the vMX configures the device through the PF driver with the VLAN ID that the PF driver receive. When an Ethernet frame is received, the outer VLAN is compared with configured VLAN ID and frame, and then forwarded to the VF associated with the vMX. In another vMX instance, using a different VF on the same physical port, you must configure a different set of VLAN IDs to the device to receive the Ethernet frames. As a result, multiple vMX instances can share the same physical port only if the VLAN ID is unique. The IFL configuration determines the VLAN ID of the Ethernet frame that needs to be sent to the vMX through the VF.

NOTE: When using the modified driver, you can only create a single VF per PF. Unmodified driver supports multiple VFs per PF.

NOTE: By default LLDP is consumed by i40e physical function (PF) driver. To disable the LLDP packet consumption at PF level, use following command:

`#echo lldp stop > /sys/kernel/debug/i40e/PCI-bus-info/`

You can retrieve PCI bus information from the output of `ethtool -i interface-name| grep bus-info` command.

NOTE: If you notice that i40e driver link is not stable, you can renegotiate the link speed by using the following command:

`# ethtool -r ethX/interface-name`

Deploying vMX with Unmodified i40e Driver

Before installing a vMX instance, you must choose to load the unmodified i40e driver. To load the unmodified i40e driver:

NOTE: Starting in Junos OS Release 18.4R1, vMX instances can be deployed with an unmodified i40e driver on Ubuntu version 16.04. XL710 NIC recommended if unmodified i40e driver version is 2.4.10 and firmware version 6.01. Unmodified 2.4.10 driver is qualified for XL710.

NOTE: To use the unmodified driver, you must set the value of the `use_native_drivers` command to `true` in the vMX configuration file.

1. Upgrade the host OS to Ubuntu 16.04 version or later, and ensure that the IP route package value is `iproute2-4.9.0`.

2. Remove the existing driver module.

```
rmmod i40e
```

3. Install the required version of the unmodified driver on the host. If host is running an older version of the driver, upgrade the host to the required version. For example:

```
insmod i40e.ko
```

4. Use the ethtool -i *interface-name* utility to determine the driver information.

```
[root@host ~]# ethtool -i eth8

driver: i40e
version: 2.4.10
firmware-version: 6.01 0x80003484 1.1747.0
```

NOTE: The firmware version must be compatible with the driver version that you are installing.

5. Create a virtual function (VF) using either of the following commands.

```
echo num_of_vf > /sys/class/net/<interface-name>/device/sriov_numvfs
```

For example, if you want to create two VFs, use the following command:

```
echo 2 > /sys/class/net/eth16/device/sriov_numvfs
```

If you want to modify the number of VFs, use the following command:

```
echo 0 > /sys/class/net/<interface-name>/device/sriov_numvfs
echo num_of_vf > /sys/class/net/<interface-name>/device/sriov_numvfs
```

NOTE: On some PCI devices, when you change the number of VFs, you might receive the error message : `Device or resource busy`. In such cases, you first set sriov_numvfs to 0, and then set it to your new value.

If the value of sriov_numvfs > 0, then you have to set it to 0 first and then change it to numeric value.

6. Configure the vMX configuration file (vmx.conf) to skip the installation of the modified driver. For example:

```
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 12
    console_port: 8602
    device-type : sriov
    use_native_drivers : true
```

7. Install vMX.

```
./vmx.sh --install --cfg ../vmx.conf
```

The vMX programs the PF driver with VLAN information. The PF driver compares the outer VLAN of the VLAN tag information of the packets against the programmed VLAN and forwards to corresponding VF.

1. Enter the CLI configuration mode after logging in to the vMX and set the per interface configuration knob for the respective interface.

```
set interfaces <interface-name>  vlan-offload
```

Moving from Modified i40e Driver to Unmodified i40e Driver

When you try to move an existing deployment from modified i40e driver to unmodified i40e driver, perform the following steps:

NOTE: Use the `set interface <interface-name> vlan-offload` command to offload the VLAN filtering to unmodified PF driver.

NOTE: Support for modified drivers for i40e is not available starting in Junos OS Release 19.1 and later releases.

1. Install the required version of the unmodified driver on the host. If host is running an older version of the driver, upgrade the host to the required version. For example:

```
insmod ./i40e.ko <installing the driver>
ethtool -i eth8
driver: i40e
version: 2.4.10
firmware-version: 6.01 0x80003484 1.1747.0
```

NOTE: The firmware version must be compatible with the driver version you are installing.

2. Configure the vMX configuration file (vmx.conf) to skip the installation of the modified driver. For example:

```
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 12
    console_port: 8602
    device-type : sriov
    use_native_drivers : true
```

3. Install vMX.

```
./vmx.sh --install --cfg ../vmx.conf
```

4. Login to vMX and set the per IFD configuration knob for the respective IFDs.

```
set interfaces <interface-name>  vlan-offload
```

Moving from Unmodified i40e Driver to Modified i40e Driver

When you try to move an existing deployment to from unmodified i40e driver to modified i40e driver, perform the following steps:

1. Clear the relevant knob from vMX configuration file.

```
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 12
    console_port: 8602
    device-type : sriov
```

2. Clean the vMX.

```
./vmx.sh --cleanup --cfg ../vmx.conf
```

3. Reinstall vMX on your device.

```
./vmx.sh --install --cfg ../vmx.conf
```

RELATED DOCUMENTATION

Example: Enabling SR-IOV on vMX Instances on KVM | 83

Modified and Unmodified IXGBE Driver | 100

Understanding the Features Supported on Modified and Unmodified Drivers | 106

Modified and Unmodified IXGBE Driver

IN THIS SECTION

Read this topic to understand the modified and unmodified IXGBE driver support for vMX instances.

Understanding the Differences between Modified and Unmodified IXGBE Driver

The single root I/O virtualization (SR-IOV) functionality consists of a physical function (PF) driver and a virtual function (VF) driver. The PF driver of an SR-IOV device is used to manage the physical function of an SR-IOV capable device. A VF driver of an SR-IOV device shares one or more physical resources with the physical function and other virtual functions that are associated with the same physical function.

In the modified IXGBE driver, the PF driver is in VLAN promiscuous mode and the modified driver accepts and transfers all the packets to the virtual Forwarding Plane (vFP) irrespective of the VLAN tag. The vFP does the filtering of packets based on the VLAN and rejects the packets if the VLAN is not programmed. The knowledge of VLAN stays within the vFP.

In the unmodified IXGBE driver, the vMX configures the device using the PF driver with the VLAN ID the driver receives. When an Ethernet frame is received, the outer VLAN is compared with the configured VLAN ID and frame, and then forwarded to the appropriate VF associated with the vMX instance. When another vMX instance is using a different VF on the same physical port, you can configure a different set of VLAN IDs to the device to receive the Ethernet frames. As a result, multiple vMX instances can share the same physical port only if the VLAN ID is unique (multiple VFs are supported on a port).

The IFL configuration determines the VLAN ID of the Ethernet frames that can be sent to the vMX through the VF. In the case of unmodified IXGBE driver, the MAC cannot be set to promiscuous mode resulting in the layer 2 forwarding functionality not being supported on the vMX with the unmodified driver.

NOTE: On a vMX instance, you can create multiple VFs on the same PF, but only one VF from the PF must be assigned to one vMX instance. You can assign other VFs from the same PF to other vMX instances.

Deploying vMX with Unmodified IXGBE Driver

Before installing a vMX instance, you must choose to load the unmodified IXGBE driver. To load the unmodified IXGBE driver:

> **NOTE**: Starting in Junos OS Release 18.4R1, vMX instances can be deployed with an unmodified IXGBE driver on Ubuntu version 16.04. IXGBE based NIC recommended if IXGBE driver version is 5.3.6 and compatible firmware version is 0x61bd0001.

> **NOTE**: To use the unmodified driver, you must set the value of the `use_native_drivers` command to `true` in the vMX configuration file.

1. Upgrade the host OS to Ubuntu 16.04 version or later, and ensure that the IP route package value is `iproute2-4.9.0`.
2. Remove the existing driver module.

```
rmmod ixgbe
```

3. Install the required version of the unmodified driver on the host. If host is running an older version of the driver, upgrade the host to the required version. For example:

```
insmod ixgbe.ko
```

4. Use the ethtool -i *interface-name* utility to determine the driver information.

```
[root@host ~]# ethtool -i eth6

driver: ixgbe
version: 5.3.6
firmware-version: 0x61bd0001
```

> **NOTE**: The firmware version must be compatible with the driver version that you are installing.

5. Create a virtual function (VF) using either of the following commands.

```
echo num_of_vf > /sys/class/net/<interface-name>/device/sriov_numvfs
```

For example, if you want to create two VFs, use the following command:

```
echo 2 > /sys/class/net/eth16/device/sriov_numvfs
```

If you want to modify the number of VFs, use the following command:

```
echo 0 > /sys/class/net/<interface-name>/device/sriov_numvfs
echo num_of_vf > /sys/class/net/<interface-name>/device/sriov_numvfs
```

> **NOTE**: On some PCI devices, when you change the number of VFs, you might receive the error message : `Device or resource busy`. In such cases, you first set sriov_numvfs to 0, and then set it to your new value.
>
> If the value of sriov_numvfs > 0, then you have to set it to 0 first and then change it to numeric value.

6. Configure the vMX configuration file (vmx.conf) to skip the installation of the modified driver. For example:

```
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 12
    console_port: 8602
    device-type : sriov
    use_native_drivers : true
```

7. Install vMX.

```
./vmx.sh --install --cfg ../vmx.conf
```

The vMX programs the PF driver with VLAN information. The PF driver compares the outer VLAN of the VLAN tag information of the packets against the programmed VLAN and forwards to corresponding VF.

1. Enter the CLI configuration mode after logging in to the vMX and set the per interface configuration knob for the respective interface.

```
set interfaces <interface-name>  vlan-offload
```

Moving from Modified IXGBE Driver to Unmodified IXGBE Driver

When you try to move an existing deployment from modified IXGBE driver to unmodified IXGBE driver, perform the following steps:

NOTE: Use the `set interface <interface-name> new-vlan-offload-knob` command to offload the VLAN filtering to unmodified PF driver.

1. Install the required version of the unmodified driver on the host. If host is running an older version of the driver, upgrade the host to the required version. For example:

```
insmod ./ixgbe.ko <installing the driver>
ethtool -i eth8
driver: ixgbe
version: 5.3.6
firmware-version: 0x61bd0001
```

NOTE: The vMX with the modified driver is the default choice at the time of spawning vMX. You can choose the unmodified PF driver through the configuration. This selection must be made before installing vMX and cannot be modified during run time.

2. Configure the vMX configuration file (vmx.conf) to skip the installation of the modified driver. For example:

```
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 12
    console_port: 8602
```

```
    device-type : sriov
    use_native_drivers : true
```

3. Install vMX.

```
./vmx.sh --install --cfg ../vmx.conf
```

4. Login to vMX and set the VLAN offload option.

```
set interfaces <interface-name>  vlan-offload
```

A single VF instance might be supported on a PF and the total number of VLANs per interface is limited to 64 if the vlan-offload option is configured.

Moving from Unmodified IXGBE Driver to Modified IXGBE Driver

When you try to move an existing deployment from unmodified IXGBE driver to modified IXGBE driver, perform the following steps:

1. Clear the relevant knob from vMX configuration file.

```
FORWARDING_PLANE:
    memory-mb   : 16384
    vcpus       : 12
    console_port: 8602
    device-type : sriov
```

2. Cleanup the vMX, delete existing configuration and VLAN IDs.

```
./vmx.sh --cleanup --cfg ../vmx.conf
```

3. Reinstall vMX on your device.

```
./vmx.sh --install --cfg ../vmx.conf
```

RELATED DOCUMENTATION

Example: Enabling SR-IOV on vMX Instances on KVM | 83

Modified and Unmodified i40e Driver | 95

Understanding the Features Supported on Modified and Unmodified Drivers | 106

Understanding the Features Supported on Modified and Unmodified Drivers

Table 16 on page 106 lists the features that are supported on modified and unmodified drivers.

Table 16: Features Supported on Modified and Unmodified Drivers

Feature	IXGBE Driver				i40e Driver			
	vlan-offload=on		vlan-offload=off		vlan-offload=on		vlan-offload=off	
	trust=on	trust=off	trust=on	trust=off	trust=on	trust=off	trust=on	trust=off
Untagged (IPv4 or IPv6)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Single tag (IPv4 or IPv6)	Yes	Yes	No	No	Yes	Yes	Yes	Yes
Q-in-Q or double tag	Yes	Yes	No	No	Yes	Yes	Yes	Yes
MTU (64 - 9000)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
MPLS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 16: Features Supported on Modified and Unmodified Drivers *(Continued)*

Feature	IXGBE Driver				i40e Driver			
BGP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OSPF version 2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OSPF version 3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AE or LACP	Yes	No	Yes	No	Yes	Yes	Yes	Yes
LLDP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
STP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BFD or Micro-BFD	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CFM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
LFM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ISIS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
VRRP	No	No	No	No	Yes	Yes	Yes	Yes
Multicast (PIM or IGMP or IGMP traffic)	Yes	No	Yes	No	Yes	No	Yes	No

Table 16: Features Supported on Modified and Unmodified Drivers *(Continued)*

Feature	IXGBE Driver				i40e Driver			
Layer 2 bridging	No	No	No	No	Yes	No	Yes	No
Multiple VFs per PF with VLAN to VF mapping	Yes	Yes	No	No	Yes	Yes	No	No

NOTE: By default, the `ethtool priv-flag vf-true-promisc-support` option is set to false. It means that the promiscuous mode for the virtual function (VF) 'will be set to limited mode.

To set the promiscuous mode for the VF to true promiscuous and allow the VF to see all ingress traffic, use the following command:

```
#ethtool set-priv-flags vf-true-promisc-support on
```

NOTE: Support for modified drivers for i40e is not available starting in Junos OS Release 19.1 and later releases.

NOTE: The `use_native_drivers` option does not support Layer 2 promiscuous mode and other such features.

RELATED DOCUMENTATION

Modified and Unmodified i40e Driver | **95**

Modified and Unmodified IXGBE Driver | **100**

CHAPTER 4

Configuring vMX Chassis-Level Features

Configuring the Number of Active Ports on vMX

You can specify the number of active ports for vMX. The default number of ports is 10, but you can specify any value in the range of 1 through 23. You can change this number if you want to limit the number of Ethernet interfaces in the VCP VM to match the number of NICs added to the VFP VM.

NOTE: If you are running virtio interfaces in lite mode, you can use up to 96 ports.
Other configurations running in performance mode support up to 23 ports.

To specify the number of active ports, configure the number of ports at the `[edit chassis fpc 0 pic 0]` hierarchy level.

```
[edit]
user@vmx# set chassis fpc 0 pic 0 number-of-ports
```

RELATED DOCUMENTATION

- Naming the Interfaces | 110
- Configuring the Media MTU | 111
- Enabling Performance Mode or Lite Mode | 112
- Tuning Performance Mode | 114

Naming the Interfaces

vMX supports the following interface types:

- Gigabit Ethernet (ge)
- 10-Gigabit Ethernet (xe)
- 100-Gigabit Ethernet (et)

By default, the interfaces come up as ge interfaces with 1 Gbps bandwidth in the Junos OS configuration. The default port speed values for the interface types are 1 Gbps (ge), 10 Gbps (xe), and 100 Gbps (et). If you do not enable schedulers, the speed is only for display purposes and is not

enforced. If you enable schedulers, the transmit rate of the port is limited to the speed unless it is overridden by the shaping rate in the CoS configuration.

To specify the interface types, configure the interface type at the `[edit chassis fpc 0 pic 0]` hierarchy level.

```
[edit]
user@vmx# set chassis fpc 0 pic 0 interface-type (ge | xe | et)
```

RELATED DOCUMENTATION

- Configuring the Number of Active Ports on vMX | 110
- Configuring the Media MTU | 111
- Enabling Performance Mode or Lite Mode | 112
- Tuning Performance Mode | 114

Configuring the Media MTU

For vMX, you can configure the media MTU in the range 256 through 9500.

NOTE: For VMware, the maximum value is 9000. For AWS, the maximum value is 1514.

You configure the MTU by including the `mtu` statement at the `[edit interface interface-name]` hierarchy level.

```
[edit]
user@vmx# set interface ge-0/0/0 mtu bytes
```

RELATED DOCUMENTATION

- Configuring the Number of Active Ports on vMX | 110
- Naming the Interfaces | 110

Enabling Performance Mode or Lite Mode

vMX can be configured to run in two modes depending on the use case.

- Lite mode—Needs fewer resources in terms of CPU and memory to run at lower bandwidth.
- Performance mode—Needs higher resources in terms of CPU and memory to run at higher bandwidth.

> **NOTE**: Starting in Junos OS Release 15.1F6 and later releases performance mode is enabled implicitly by default.
>
> When you enable performance mode, make sure you have configured the proper number of vCPUs (four or more VPCUs) and memory for your VMs based on your use case.

You can explicitly enable lite-mode. If you are using paravirtualized network interfaces such as virtio (for KVM) or VMXNET3 (for VMware) for lab simulation use cases, you can disable performance mode by including the `lite-mode` statement at the [`edit chassis fpc 0`] hierarchy level.

```
[edit]
user@vmx# set chassis fpc 0 lite-mode
```

You can explicitly enable performance mode by including the `performance-mode` statement at the [`edit chassis fpc 0`] hierarchy level.

```
[edit]
user@vmx# set chassis fpc 0 performance-mode
```

NOTE: We recommend that you enable hyperthreading in BIOS. We recommend that you verify the process with the vendor because different systems have different methods to enable hyperthreading.

Starting with Junos OS Release 17.3R1, the `show chassis hardware` command displays the mode in which vMX is running in the part number field for the FPC. RIOT-PERF indicates performance mode and RIOT-LITE indicates lite mode. For example, this output indicates that vMX is running in lite mode.

```
user@vmx> show chassis hardware
```

```
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                VM54599D128A      VMX
Midplane
Routing Engine 0                                         RE-VMX
CB 0                                                     VMX SCB
CB 1                                                     VMX SCB
FPC 0                                                    Virtual FPC
  CPU            Rev. 1.0 RIOT-LITE    BUILTIN
  MIC 0                                                  Virtual
    PIC 0                BUILTIN      BUILTIN           Virtual
```

Table 17 on page 113 highlights some of the challenging features which are supported in the Fast Path and some which are not supported. Features which are not supported in the Fast Path still work but they get less than 100K PPS per worker vCPU.

Table 17: Features Support in Fast Path

Features	Support in Fast Path
Pseudowire Headend Termination (PWHT) (Layer 2 VPN)	Not Supported
L2 circuit	Not Supported
Ethernet VPN (EVPN)	Not Supported

Table 17: Features Support in Fast Path *(Continued)*

Features	Support in Fast Path
Virtual Extensible LAN protocol (VXLAN)	Not Supported
MPLS-over-UDP (MPLSoUDP)	Not Supported
Inline J-flow	Supported
Pseudowire Headend Termination (PWHT) (Layer 3 VPN and IP)	Supported
GRE	Supported
logical tunnel interfaces (lt)	Supported

Release History Table

Release	Description
15.1F6	Starting in Junos OS Release 15.1F6 and later releases performance mode is enabled implicitly by default.

RELATED DOCUMENTATION

Tuning Performance Mode | 114

lite-mode | 115

performance-mode | 117

Tuning Performance Mode

To tune performance mode for the traffic, you can specify the number of Workers dedicated to processing multicast and control traffic. You can specify any value in the range of 0 through 15. The default of 0 specifies that all available Workers are used to process all traffic.

The number of dedicated Workers specified in relation to the number of available Workers results in the following behavior:

- If the number of dedicated Workers is greater than or equal to the number of available Workers, then all available Workers are used to process all traffic.
- If the number of dedicated Workers is less than the number of available Workers, then the first set of available Workers (equal to the specified number of dedicated Workers) is used to process multicast and control traffic while the remaining available Workers are used to process flow cache traffic.

To specify the number of dedicated Workers for processing multicast and control traffic, configure the number of Workers at the [`edit chassis fpc 0 performance-mode`] hierarchy level.

```
[edit]
user@vmx# set chassis fpc 0 performance-mode number-of-ucode-workers number-workers
```

NOTE: Changing the number of Workers reboots the FPC.

RELATED DOCUMENTATION

Enabling Performance Mode or Lite Mode | 112

performance-mode | 117

lite-mode

IN THIS SECTION

Syntax

```
lite-mode;
```

Hierarchy Level

```
[edit chassis fpc 0]
```

Release Information

Statement introduced in Junos OS Release 15.1F4 and 16.1R1.

Description

(vMX routers only) Enables vMX to run in lite mode and disables performance mode. Lite mode needs fewer vCPUs and memory to run at lower bandwidth. If you are using paravirtualized network interfaces such as virtio (for KVM) or VMXNET3 (for VMware) for lab simulation use cases, you can enable lite mode.

NOTE: Make sure you have configured the proper number of vCPUs and memory for your VMs based on your use case. If you have not configured enough vCPUs for performance mode, vMX runs in lite mode.

Starting with Junos OS Release 15.1F6, performance mode is enabled by default for vMX.

NOTE: The FPC reboots if you change this configuration.

Options

lite-mode Enables lite mode.

To disable lite mode, enable performance mode by including the `performance-mode` statement at the [`edit chassis fpc 0`] hierarchy level.

Required Privilege Level

interface—To view this statement in the configuration.

interface-control—To add this statement to the configuration.

RELATED DOCUMENTATION

performance-mode | 117

performance-mode

IN THIS SECTION

Syntax

```
performance-mode {
    number-of-ucode-workers number-of-ucode-workers;
}
```

Hierarchy Level

```
[edit chassis fpc 0]
```

Release Information

Statement introduced in Junos OS Release 15.1F4 and 16.1R1.

`number-of-ucode-workers` option introduced in Junos OS Release 15.1F6 and 16.2R1 for vMX routers.

Description

(vMX routers only) Enables vMX to run in performance mode. Performance mode needs more vCPUs and memory to run at higher bandwidth.

NOTE: When you enable performance mode, make sure you have configured the proper number of vCPUs and memory for your VMs based on your use case. If you have not configured enough vCPUs, vMX runs in lite mode.

Starting with Junos OS Release 15.1F6, performance mode is enabled by default for vMX.

NOTE: The FPC reboots if you change this configuration.

You can tune performance mode for unicast traffic by changing the number of Workers dedicated to processing multicast and control traffic. Starting with Junos OS Release 17.2R1, you do not need to

specify dedicated Workers for processing multicast traffic. The default specifies that all available Workers are used to process all traffic.

The number of dedicated Workers specified in relation to the number of available Workers results in the following behavior:

- If the number of dedicated Workers is greater than or equal to the number of available Workers, then all available Workers are used to process all traffic.
- If the number of dedicated Workers is less than the number of available Workers, then the first set of available Workers (equal to the specified number of dedicated Workers) is used to process multicast and control traffic while the remaining available Workers are used to process flow cache traffic.

Options

performance-mode

Enables performance mode.

To disable performance mode, enable lite mode by including the `lite-mode` statement at the [`edit chassis fpc 0`] hierarchy level.

number-of-ucode-workers ***number-workers***

Specifies the number of dedicated Workers for processing multicast and control traffic.

- **Range:** 0 through 15
- **Default:** 0 specifies that all available Workers are used to process all traffic.

Required Privilege Level

`interface`—To view this statement in the configuration.

`interface-control`—To add this statement to the configuration.

RELATED DOCUMENTATION

lite-mode | 115

5

CHAPTER

Class of Service for vMX

CoS on vMX Overview

vMX supports two-level hierarchical scheduling (per-unit scheduler or hierarchical scheduler) with VLAN queuing. Each VLAN (logical interface) uses three traffic classes and eight queues.

Starting with Junos OS Release 17.3R1, vMX supports four-level hierarchical scheduling for up to 16 level 2 CoS scheduler nodes. The level 2 node maps to the interface set or VLAN (logical interface).

vMX supports shaping at the traffic class level, not at the queue level. A traffic class is a bundle of queues with fixed priority. The next level in the hierarchy is the VLAN (logical interface), which is a bundle of traffic classes.

vMX has the following fixed priorities and queues for these traffic classes:

- Traffic Class 1: High (strict priority)

  Queue 0

  Queue 6

- Traffic Class 2: Medium (strict priority)

  Queue 1

  Queue 7

- Traffic Class 3: Low

  Queue 2

  Queue 3

  Queue 4

  Queue 5

> **NOTE**: Both Traffic Class 1 and Traffic Class 2 follow strict priority, so all excess traffic is discarded as tail drops. However, Traffic Class 3 does not follow strict priority, so the shaping rate is set to the shaping rate of the VLAN.
>
> All queues in the same traffic class have equal priority, so the scheduler pulls packets from each queue in the traffic class based on weighted round robin (WRR) for the VLAN.

All configured forwarding classes must be mapped to one of the queues.

The following features are not supported::

- Weighted random early detection (WRED)
- Queue buffer size configuration

NOTE: No commit errors are displayed for unsupported features.

Starting in Junos OS Release 18.4R1, the quality of service (QoS) configuration is enhanced such that, when a port is oversubscribed and congested, a subscriber with higher priority gets more weight than a subscriber with a lower priority. For example, when a subscriber on a port has 100 MB service and another subscriber has 10 MB service then the subscriber with 100 MB service gets more priority than the subscriber with 10 MB service. You must ensure that the priority is followed at level 1 and level 2 nodes, regardless of the weight. The WRR provides the ability handle the oversubscription so that the scheduled traffic reflects a ratio of the shaping rate configured for the individual VLANs.

Use the following commands to configure a maximum number of 16384 subscribers per port on a level 2 node and a maximum number of 32768 subscribers per port on a level 3 node:

```
set interfaces <interface-name> hierarchical-scheduler maximum-hierarchy 3 max-l2-nodes 16384
set interfaces <interface-name> hierarchical-scheduler maximum-hierarchy 3 max-l3-nodes 32768
```

NOTE: The default number of subscribers that are configured per level 2 node is 4000.

Use the following command to disable the WRR feature:

```
subport_oversubscription_disable=1 in the /etc/riot/runtime.conf of the vFP
```

The following list describes the limitations for WRR:

- The delay-buffer rate must be configured for WRR to work appropriately.
- A discrepancy in the delay-buffer rate values, among the VLANs belonging to the same level 2 scheduler node can cause the WRR to work incorrectly.
- The WRR works incorrectly when the ratio of shaping rate is greater than 100 among all the subscribers.
- The number of level 2 scheduler nodes and the number of subscribers per level 2 scheduler node must be equal to 32,000.
- Any modification to the level 2 scheduler node configuration would require a FPC reset.

RELATED DOCUMENTATION

CoS Features and Limitations on vMX

IN THIS SECTION

vMX has the following limitations for CoS support:

- Schedulers support only the `transmit-rate` and `excess-rate` statements. Only weights are supported at the queue level, so transmission rate and excess rate are used for calculating queue weights.
  - If `transmit-rate percent` is configured at the queue level, then configure guaranteed rate at the VLAN level.

    NOTE: Guaranteed rate is not supported, but it is used to calculate queue weights.

  - If you only configure transmit rate, queue weights are calculated based on the transmission rate.
  - If you only configure excess rate, queue weights are calculated based on the excess rate.
  - If you configure both transmit rate and excess rate, queue weights are calculated based on the excess rate.
  - If you configure the excess rate for one queue, the excess rate is expected for all the queues to compute the weights. If the excess rate is not configured, the default weight of 1 is used.

    NOTE: To get the expected behavior, you must configure the excess rate for all queues.

- Traffic control profiles support only the `shaping-rate` and `scheduler-map` statements.

  If a traffic control profile has a default scheduler map, you must configure the guaranteed rate.

- For high- and medium-priority traffic classes, the transmission rate is the shaping rate.
- For low-priority queues, the shaping rate for the VLAN is used for the queue. As a result, the low-priority queues can burst up to the configured shaping rate for the VLAN. The transmission rate is used as the WRR weight when there is more than one queue configured for a given priority.

Some considerations for the high- and medium-priority traffic classes:

- All excess traffic from the traffic classes for high- and medium-priority queues are discarded as tail drops.
- For high- and medium-priority traffic classes, the transmission rate is the shaping rate.

  If the transmission rate is not configured and the shaping rate is configured, then the queue weight is calculated based upon the configured shaping rate.

  If you configure the transmission rate for both queues of the same traffic class, the shaping rate of the traffic class is the sum of the individual transmission rates of the queues for that traffic class.
- If a queue is not configured, its transmission rate is set to zero.

  If no queues are configured, the shaping rate of the VLAN is applied to the traffic class as the transmission rate.
- If any of the queues in the traffic class is configured, the shaping rate of the VLAN is set to the guaranteed rate of the configured queue. If a queue is not configured, the guaranteed rate is set to zero by default.
- If the sum of the rates of the individual queues in a traffic class exceeds the shaping rate of the VLAN, the shaping rate of the VLAN is used as the shaping rate of the traffic class.

Weighted Round-Robin of Subscriber Traffic on a Port Limitations

The following list describes the limitations for WRR:

- A discrepancy in the delay-buffer rate values among the VLANs belonging to the same level 2 scheduler node can cause the WRR to work incorrectly.
- WRR does not work correctly if the ratio of the shaping rate is greater than 100 among all the subscribers.
- The number of level 2 scheduler nodes and the number of subscribers per level 2 scheduler node must be equal to 32,000 for it to work correctly.
- Any modification to the level 2 scheduler node configuration requires an FPC reset.

RELATED DOCUMENTATION

Configuring Hierarchical CoS on vMX | 131

CoS on vMX Overview | 121

Configuring Four-Level Hierarchical Scheduling on vMX

Starting with Junos OS Release 17.3R1, four-level hierarchical scheduling for up to 16 level 2 CoS scheduler nodes is supported on vMX routers. The level 2 node maps to the interface set or VLAN (logical interface). Two of the level 2 nodes are used for control traffic. If you configure less than four nodes, no commit errors are displayed but there are not enough nodes for other applications to use. Different interfaces can have a different number of level 2 nodes. The interface can be an inline service interface.

To configure four-level hierarchical scheduling:

1. Hierarchical CoS is disabled by default. Configure flexible queuing to enable CoS.

```
[edit]
user@vmx# set chassis fpc 0 flexible-queuing-mode
```

NOTE: The FPC reboots if you enable flexible queuing.

2. Enable hierarchical scheduling.

```
[edit]
user@vmx# set interfaces interface-name implicit-hierarchy
```

3. Set the maximum number of hierarchical scheduling levels for node scaling to 3. If the `maximum-hierarchy-levels` option is not configured, it is automatically set to 2.

```
[edit]
user@vmx# set interfaces interface-name hierarchical-scheduler maximum-hierarchy-levels 3
```

4. Specify the maximum number of level 2 scheduler nodes; only 1, 2, 4, 8, and 16 are valid values. The default value is 4. We recommend that you do not configure less than four nodes because two of the nodes are used for control traffic.

```
[edit]
user@vmx# set interfaces interface-name hierarchical-scheduler maximum-l2-nodes number-of-nodes
```

For example:

```
[edit]
user@vmx# set interfaces ge-0/0/0 hierarchical-scheduler maximum-l2-nodes 4
```

NOTE: This configuration must be present before you reboot the FPC.

RELATED DOCUMENTATION

- CoS on vMX Overview | 121
- CoS Features and Limitations on vMX | 123
- Configuring Hierarchical CoS on vMX | 131

Packet Loss Priority and Drop Profiles on vMX

IN THIS SECTION

vMX handles packet priorities within a queue by assigning a threshold to each loss priority within a queue and dropping new packets of that loss priority level when the queue depth exceeds the threshold. When the queue becomes oversubscribed, packets of lower priority are dropped to ensure that there is room in the queue for packets of higher priority.

Packet loss priority has four loss priority levels:

- low
- medium-low
- medium-high
- high

vMX supports three thresholds so the medium-low and medium-high loss priority levels are grouped together. vMX maps the packet loss priority to tricolor marking as follows:

Packet Loss Priority	Color
low	green
medium-low	yellow
medium-high	yellow
high	red

vMX drop profiles define the threshold within a queue for a given loss priority as the fill level value associated with the drop probability of 100 percent. If you do not specify a drop probability of 100 percent in the drop profile, the threshold defaults to 100 percent. All other fill level values are ignored. These drop profiles can be referenced by the scheduler to evaluate packets with different loss priority settings.

You can set packet loss priority for packets using behavior aggregate (BA) classifiers, firewall filters, or firewall policers.

Limitations

vMX has the following limitations for supporting drop profiles and packet loss priority:

- If you do not apply drop profiles to the queue, then packets are tail dropped.

- The `show interface queue` command does not display separate drop rates for the medium-high PLP and medium-low PLP because they both map to yellow. All yellow drop rates appear as medium-high drops.

RELATED DOCUMENTATION

Managing Congestion Using Drop Profiles and Packet Loss Priorities on vMX | 128

CoS on vMX Overview | 121

CoS Features and Limitations on vMX | 123

Managing Congestion Using Drop Profiles and Packet Loss Priorities on vMX

IN THIS SECTION

When you are configuring CoS, you can manage congestion by configuring drop profiles to specify the thresholds for packet loss priority. You reference the drop profiles in the scheduler configuration to assign a drop profile to the loss priority setting.

To configure how packet loss priority is handled for queues, perform these tasks:

Configuring Drop Profiles

Drop profiles specify the threshold for a given loss priority.

NOTE: The threshold for the loss priority assigned this drop profile is the `fill-level` value associated with the `drop-probability` of 100. If you do not specify a drop probability of 100 percent in the drop profile, the fill level defaults to 100 percent. All other fill levels are ignored.

To specify the drop profile, include the `drop-profiles` statement at the `[edit class-of-service]` hierarchy level.

```
[edit]
user@vmx# set class-of-service drop-profiles profile-name
```

To specify the threshold for the loss priority, include the `fill-level` and `drop-probability` statements at the `[edit class-of-service drop-profiles profile-name]` hierarchy level.

```
[edit class-of-service drop-profiles profile-name]
user@vmx# set fill-level percentage drop-probability percentage
```

For example, the `dpLow` drop profile specifies a threshold of 100 percent, the `dpMed` drop profile specifies a threshold of 75 percent, and the `dpHigh` drop profile specifies a threshold of 50 percent.

```
[edit]
user@vmx# set class-of-service drop-profiles dpLow fill-level 100 drop-probability 100
user@vmx# set class-of-service drop-profiles dpMed fill-level 75 drop-probability 100
user@vmx# set class-of-service drop-profiles dpHigh fill-level 50 drop-probability 100
```

Configuring Schedulers with Drop Profiles

The drop profile map contains the mapping of loss priority and protocol type to configured drop profiles. You can associate multiple drop profile maps with a scheduler.

NOTE: If you do not apply drop profiles to the queue, then packets are tail dropped.

To specify the drop profile map, include the `drop-profile-map` statement at the `[edit class-of-service schedulers` *`scheduler-name`*`]` hierarchy level.

```
[edit class-of-service schedulers scheduler-name]
user@vmx# set drop-profile-map loss-priority (any | low | medium-low | medium-high | high)
protocol any drop-profile profile-name
```

For example, the `sched-be` scheduler applies the `dpLow` drop profile to packets with low loss priority for any protocol type, applies the `dpMed` drop profile to packets with medium-high loss priority for any protocol type, and applies the `dpHigh` drop profile to packets with high loss priority for any protocol type.

```
[edit class-of-service schedulers sched-be]
user@vmx# set drop-profile-map loss-priority low protocol any drop-profile dpLow
user@vmx# set drop-profile-map loss-priority medium-high protocol any drop-profile dpMed
user@vmx# set drop-profile-map loss-priority high protocol any drop-profile dpHigh
```

RELATED DOCUMENTATION

- Packet Loss Priority and Drop Profiles on vMX | 126
- CoS on vMX Overview | 121
- CoS Features and Limitations on vMX | 123

Configuring Hierarchical CoS on vMX

IN THIS SECTION

To configure hierarchical CoS, perform these tasks:

Enabling Flexible Queuing

Hierarchical CoS is disabled by default. To enable hierarchical CoS, include the `flexible-queuing-mode` statement at the `[edit chassis fpc 0]` hierarchy level and restart the FPC.

```
[edit]
user@vmx# set chassis fpc 0 flexible-queuing-mode
```

Mapping Forwarding Classes to Queues on vMX

You must map all configured forwarding classes to one of the queues.

```
[edit]
user@vmx# set class-of-service forwarding-classes class class-name queue-num queue-number
```

Configuring Traffic Control Profiles for vMX

Traffic control profiles support only the `shaping-rate` and `scheduler-map` statements for vMX.

To specify the shaping rate, include the `shaping-rate` statement at the `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level.

```
[edit]
user@vmx# set class-of-service traffic-control-profiles profile-name shaping-rate rate
```

To specify the scheduler map, include the `scheduler-map` statement at the `[edit class-of-service traffic-control-profiles profile-name]` hierarchy level.

```
[edit]
user@vmx# set class-of-service traffic-control-profiles profile-name scheduler-map map-name
```

Configuring Schedulers on vMX

The scheduler map contains the mapping of forwarding classes to their schedulers. The scheduler defines the properties for the queue.

Schedulers support only the `transmit-rate` and `excess-rate proportion` statements for vMX.

To specify the transmission rate, include the `transmit-rate` statement at the `[edit class-of-service schedulers scheduler-name]` hierarchy level.

```
[edit]
user@vmx# set class-of-service schedulers scheduler-name transmit-rate rate
```

BEST PRACTICE: Guaranteed rate is not supported, so there is no reserved bandwidth for the VLAN. To get the expected behavior, we recommend that you configure the transmit rate to be the guaranteed rate.

To specify the proportion of the excess bandwidth to share, include the `excess-rate proportion` statement at the `[edit class-of-service schedulers` *`scheduler-name`*`]` hierarchy level. The value is in the range of 0 through 1000.

```
[edit]
user@vmx# set class-of-service schedulers scheduler-name excess-rate proportion value
```

If you configure the excess rate for one queue, the excess rate is expected for all the queues to compute the weights. If the excess rate is not configured, the default weight of 1 is used.

NOTE: To get the expected behavior, you must configure the excess rate for all queues.
For example, if you configure excess rate for the low-priority queues, configure the same excess rate for the high- and medium-priority queues.

RELATED DOCUMENTATION

- Example: Configuring Hierarchical CoS on vMX | 133
- CoS on vMX Overview | 121
- CoS Features and Limitations on vMX | 123

Example: Configuring Hierarchical CoS on vMX

IN THIS SECTION

This example describes how to configure hierarchical CoS on vMX with eight queues.

Requirements

This example uses the following hardware and software components:

- Junos OS Release 16.2
- vMX Release 16.2

Overview

This example configures two-level hierarchical schedulers with specified transmission rates.

Configuration

IN THIS SECTION

Configuring the Chassis

CLI Quick Configuration

```
[edit]
set chassis fpc 0 flexible-queuing-mode
```

Step-by-Step Procedure

To enable hierarchical CoS on the chassis:

1. Enable flexible queuing mode on the chassis.

```
[edit]
user@vmx# set chassis fpc 0 flexible-queuing-mode
```

Once you commit the configuration, the FPC is restarted.

Applying Shaping and Scheduling to VLANs

CLI Quick Configuration

```
[edit]
set class-of-service forwarding-classes class voice1 queue-num 0
set class-of-service forwarding-classes class video1 queue-num 1
set class-of-service forwarding-classes class data1 queue-num 2
set class-of-service forwarding-classes class data2 queue-num 3
set class-of-service forwarding-classes class data3 queue-num 4
set class-of-service forwarding-classes class data4 queue-num 5
set class-of-service forwarding-classes class voice2 queue-num 6
set class-of-service forwarding-classes class video2 queue-num 7
set interfaces ge-0/0/0 hierarchical-scheduler maximum-hierarchy-levels 2
set interfaces ge-0/0/0 vlan-tagging
set interfaces ge-0/0/0 unit 100 vlan-id 100
set interfaces ge-0/0/0 unit 100 family inet address 10.2.2.1/24
set interfaces ge-0/0/1 hierarchical-scheduler maximum-hierarchy-levels 2
set interfaces ge-0/0/1 vlan-tagging
set interfaces ge-0/0/1 unit 100 vlan-id 100
set interfaces ge-0/0/1 unit 100 family inet address 10.1.1.1/24
```

```
set class-of-service classifiers inet-precedence vlan_tos forwarding-class voice1 loss-priority
low code-points 000
set class-of-service classifiers inet-precedence vlan_tos forwarding-class video1 loss-priority
low code-points 001
set class-of-service classifiers inet-precedence vlan_tos forwarding-class data1 loss-priority
low code-points 010
set class-of-service classifiers inet-precedence vlan_tos forwarding-class data2 loss-priority
low code-points 011
set class-of-service classifiers inet-precedence vlan_tos forwarding-class data3 loss-priority
low code-points 100
set class-of-service classifiers inet-precedence vlan_tos forwarding-class data4 loss-priority
low code-points 101
set class-of-service classifiers inet-precedence vlan_tos forwarding-class voice2 loss-priority
low code-points 110
set class-of-service classifiers inet-precedence vlan_tos forwarding-class video2 loss-priority
low code-points 111
set class-of-service traffic-control-profiles ge_0_0_1_vlan_100_tcp shaping-rate 50m
set class-of-service traffic-control-profiles ge_0_0_1_vlan_100_tcp scheduler-map vlan_smap
set class-of-service interfaces ge-0/0/1 unit 100 output-traffic-control-profile
ge_0_0_1_vlan_100_tcp
set class-of-service interfaces ge-0/0/0 unit 100 classifiers inet-precedence vlan_tos
set class-of-service scheduler-maps vlan_smap forwarding-class voice1 scheduler sched_voice1
set class-of-service scheduler-maps vlan_smap forwarding-class video1 scheduler sched_video1
set class-of-service scheduler-maps vlan_smap forwarding-class data1 scheduler sched_data1
set class-of-service scheduler-maps vlan_smap forwarding-class data2 scheduler sched_data2
set class-of-service scheduler-maps vlan_smap forwarding-class data3 scheduler sched_data3
set class-of-service scheduler-maps vlan_smap forwarding-class data4 scheduler sched_data4
set class-of-service scheduler-maps vlan_smap forwarding-class voice2 scheduler sched_voice2
set class-of-service scheduler-maps vlan_smap forwarding-class video2 scheduler sched_video2
set class-of-service schedulers sched_voice1 transmit-rate 15m
set class-of-service schedulers sched_video1 transmit-rate 15m
set class-of-service schedulers sched_data1 transmit-rate 5m
set class-of-service schedulers sched_data2 transmit-rate 5m
set class-of-service schedulers sched_data3 transmit-rate 5m
set class-of-service schedulers sched_data4 transmit-rate 5m
set class-of-service schedulers sched_voice2 transmit-rate 10m
set class-of-service schedulers sched_video2 transmit-rate 10m
```

Step-by-Step Procedure

To apply shaping and scheduling:

1. Map the forwarding classes to their respective queues.

```
[edit]
user@vmx# set class-of-service forwarding-classes class voice1 queue-num 0
user@vmx# set class-of-service forwarding-classes class video1 queue-num 1
user@vmx# set class-of-service forwarding-classes class data1 queue-num 2
user@vmx# set class-of-service forwarding-classes class data2 queue-num 3
user@vmx# set class-of-service forwarding-classes class data3 queue-num 4
user@vmx# set class-of-service forwarding-classes class data4 queue-num 5
user@vmx# set class-of-service forwarding-classes class voice2 queue-num 6
user@vmx# set class-of-service forwarding-classes class video2 queue-num 7
```

2. Configure the interfaces to enable two-level hierarchical scheduling and apply scheduling to the VLANs.

```
[edit]
user@vmx# set interfaces ge-0/0/0 hierarchical-scheduler maximum-hierarchy-levels 2
user@vmx# set interfaces ge-0/0/0 vlan-tagging
user@vmx# set interfaces ge-0/0/0 unit 100 vlan-id 100
user@vmx# set interfaces ge-0/0/0 unit 100 family inet address 10.2.2.1/24
user@vmx# set interfaces ge-0/0/1 hierarchical-scheduler maximum-hierarchy-levels 2
user@vmx# set interfaces ge-0/0/1 vlan-tagging
user@vmx# set interfaces ge-0/0/1 unit 100 vlan-id 100
user@vmx# set interfaces ge-0/0/1 unit 100 family inet address 10.1.1.1/24
```

3. Configure the classifiers.

```
[edit]
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class voice1
loss-priority low code-points 000
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class video1
loss-priority low code-points 001
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class data1
loss-priority low code-points 010
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class data2
loss-priority low code-points 011
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class data3
loss-priority low code-points 100
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class data4
loss-priority low code-points 101
```

```
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class voice2
loss-priority low code-points 110
user@vmx# set class-of-service classifiers inet-precedence vlan_tos forwarding-class video2
loss-priority low code-points 111
```

4. Configure the traffic control profiles.

```
[edit]
user@vmx# set class-of-service traffic-control-profiles ge_0_0_1_vlan_100_tcp shaping-rate 50m
user@vmx# set class-of-service traffic-control-profiles ge_0_0_1_vlan_100_tcp scheduler-map
vlan_smap
```

5. Map the traffic control profiles to their respective interface.

```
[edit]
user@vmx# set class-of-service interfaces ge-0/0/1 unit 100 output-traffic-control-profile
ge_0_0_1_vlan_100_tcp
user@vmx# set class-of-service interfaces ge-0/0/0 unit 100 classifiers inet-precedence
vlan_tos
```

6. Configure the scheduler maps.

```
[edit]
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class voice1 scheduler
sched_voice1
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class video1 scheduler
sched_video1
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class data1 scheduler
sched_data1
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class data2 scheduler
sched_data2
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class data3 scheduler
sched_data3
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class data4 scheduler
sched_data4
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class voice2 scheduler
sched_voice2
user@vmx# set class-of-service scheduler-maps vlan_smap forwarding-class video2 scheduler
sched_video2
```

7. Configure the schedulers.

```
[edit]
user@vmx# set class-of-service schedulers sched_voice1 transmit-rate 15m
user@vmx# set class-of-service schedulers sched_video1 transmit-rate 15m
user@vmx# set class-of-service schedulers sched_data1 transmit-rate 5m
user@vmx# set class-of-service schedulers sched_data2 transmit-rate 5m
user@vmx# set class-of-service schedulers sched_data3 transmit-rate 5m
user@vmx# set class-of-service schedulers sched_data4 transmit-rate 5m
user@vmx# set class-of-service schedulers sched_voice2 transmit-rate 10m
user@vmx# set class-of-service schedulers sched_video2 transmit-rate 10m
```

RELATED DOCUMENTATION

- Configuring Hierarchical CoS on vMX | 131
- CoS on vMX Overview | 121
- CoS Features and Limitations on vMX | 123
- Configuring Hierarchical CoS on vMX | 131

Bypassing the Queuing Chip

When flexible queuing option is enabled, QoS is applied on all the configured ports. Applying QoS on the ports requires additional vCPU reserve for each port and this affects vCPU resource allocation. By default, all traffic passes through the queuing-chip, which decreases the available vCPU resource, there by affecting the performance.

Starting with Junos OS 18.2R1, you can bypass the queuing-chip on vMX routers to save vCPU resources when scheduling is not needed on an interface. In cases when you do not require QoS features such as hierarchical scheduling or per-vlan queuing on a particular interface, you can bypass the queuing-chip to increase the available bandwidth.

Use the following commands to enable bypass queue option:

1. Enable bypass the queuing-chip on vMX VM:

```
[edit interfaces]
user@router# set ge-0/0/1 bypass-queuing-chip
```

NOTE: Enabling the bypass queue option reboots the FPC.

When you configure the bypass queuing chip option, the `show interface queue` command does not display any output.

2. Optionally you can configure to share the resources (QoS scheduling and Workers) among a selected set of ports. This feature is supported for active-standby configuration of LAG.

```
[edit interfaces]
user@router# set interfaces ae0 aggregated-ether-options share-standby
```

When you configure the share-standby option, all the members of aggregated Ethernet (AE) interface share the same resources (vCPUs) for both Worker processing and QoS scheduling.

RELATED DOCUMENTATION

- Increasing Available Bandwidth on Rich-Queuing MPCs by Bypassing the Queuing Chip
- bypass-queuing-chip
- share-standby

CHAPTER 6

Troubleshooting vMX

Verifying Whether VMs Are Running

To verify that the VMs are running after vMX is installed, use the `virsh list` command. The `virsh list` command displays the name and state of the VM. The state can be: running, idle, paused, shutdown, crashed, or dying.

You can stop and start VMs with the following `virsh` commands.

- `virsh destroy`—Forcefully stop a VM while leaving its resources intact.
- `virsh start`—Start an inactive VM that was defined previously.

RELATED DOCUMENTATION

Connecting to VMs | 60

Viewing CPU Information

On the host server, use the `lscpu` command to display CPU information. The output displays such information as the total number of CPUs, the number of cores per socket, and the number of CPU sockets. For example:

```
root@vmx-host:~# lscpu
```

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                40
On-line CPU(s) list:   0-39
Thread(s) per core:    1
Core(s) per socket:    10
Socket(s):             4
NUMA node(s):          4
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 62
```

```
Stepping:              7
CPU MHz:               3191.766
BogoMIPS:              6385.87
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              38400K
NUMA node0 CPU(s):     0,4,8,12,16,20,24,28,32,36
NUMA node1 CPU(s):     1,5,9,13,17,21,25,29,33,37
NUMA node2 CPU(s):     2,6,10,14,18,22,26,30,34,38
NUMA node3 CPU(s):     3,7,11,15,19,23,27,31,35,39
```

RELATED DOCUMENTATION

- Verifying Whether VMs Are Running | **142**
- Viewing VFP Statistics | **143**
- Viewing VFP Log Files | **145**
- Troubleshooting VFP and VCP Connection Establishment | **146**

Viewing VFP Statistics

You can view the VFP statistics from a Web browser. The displayed statistics are not absolute counters; they are relative to the start of the HTTP session and start as all zero counters.

The RPIO Stats and Hostif Stats sections provide statistics about the internal communication between the VCP and the VFP. The RPIO session uses ports 3000 and 3001 and the HostIF session uses port 3002.

The Port Stats section provides statistics about the packets received from and transmitted to the NIC interfaces.

- There is a receive (`rx`) and transmit (`tx`) line for each port. Port 0 maps to the ge-0/0/0 interface, port 1 maps to the ge-0/0/1 interface, and so forth. rx0 displays statistics for packets received from port 0 and tx1 displays statistics for packets transmitted to port 1.
- Errors are miscellaneous errors reported by the physical layer NIC.

The Ring Stats section provides statistics about packet processing.

- There is an I/O thread (`io`) for packets received from a port.
- There is a Worker thread (`wk`) for each CPU core.
- The host interface (`host`) sends protocol packets to the VCP.
- The queue processes the packets. The columns provide this information about the queues:
  - The Producer and Consumer columns display the source and destination that generate packets for this queue. The values can be `io`, `wk`, `tx`, or `host`.
  - The Priority column displays the priority of the queue. The values can be `Normal` or `High` (only for control packets).
  - The Free and Used columns display the queue occupancy. The queue has 1024 entries.
  - The Enqueues and Dequeues columns display the number of queue operations.
  - The Drops column indicates whether the queue is being drained fast enough.

To view the statistics:

1. By default, you cannot log in to the Web browser window without configuring the username and password credentials and enabling HTTP access.

   From the VFP console, configure the username and password by invoking the `/home/pfe/riot/vfp_util.sh -setpass` command.

   ```
   root@vfp-vmx1:/home/pfe/riot# ./vfp_util.sh -setpass
   Enter new Username: pfe
   Enter new Password:
   Re-enter Password:
   Password successfully changed
   root@vfp-vmx1:/home/pfe/riot#
   ```

   To enable HTTP access, invoke this command.

   ```
   root@vfp-vmx1:/home/pfe/riot# ./vfp_util.sh -http_enable
   ```

2. Navigate to `http://`*`vfp-mgmt-ip`*`:8080/`, where *`vfp-mgmt-ip`* is the management IP address for the VFP VM.
3. When prompted, enter `pfe` as the username and the password configured in Step 1.
4. View the statistics displayed in the browser window.

5. After troubleshooting, you can disable HTTP access to improve security with this command:

```
root@vfp-vmx1:/home/pfe/riot# ./vfp_util.sh -http_disable
```

RELATED DOCUMENTATION

Viewing VFP Log Files

vMX Overview | 2

Viewing VFP Log Files

The VFP saves the following files:

- VFP log files are saved in the **/var/log** directory.
- VFP crash files are automatically saved in the VCP **/var/crash** directory.

To view the VFP log or crash files:

1. Log in to the VFP console by using the `./vmx.sh --console vfp` *vmx-id* command, where *vmx-id* is the vMX identifier specified in the startup configuration file.
2. Navigate to the appropriate directory to determine whether there are any files to view.

```
# cd /var/crash
# ls -l
```

```
-rwxr-xr-x 1 root root 864678 Jan  4 02:14 core.riot.1420366466.8271.gz
```

3. (Optional) If necessary, unzip the file and view it using GDB.

```
# gunzip core.riot.1420366466.8271.gz
# gdb /build/app core.riot.1420366466.8271
```

The VFP is configured for remote logging of the **/var/log/messages** directory. You can configure the VCP syslog facility to record the VFP log messages:

```
user@vmx# set system syslog file messages any any
user@vmx# set system syslog server routing-instances all
user@vmx# set services app-engine monitor-cpu 50 100
user@vmx# commit
```

RELATED DOCUMENTATION

- Verifying Whether VMs Are Running | 142
- Viewing CPU Information | 142
- Viewing VFP Statistics | 143
- Troubleshooting VFP and VCP Connection Establishment | 146

Troubleshooting VFP and VCP Connection Establishment

IN THIS SECTION

Purpose

When the VCP and VFP connection is established, the `show interfaces terse` command in the VCP CLI displays the ge-0/0/*x* interfaces and the following messages appear in the VFP syslog file:

```
RPIO: Accepted connection from 128.0.0.1:50896 <-> vPFE:3000
RPIO: Accepted connection from 128.0.0.1:56098 <-> vPFE:3000
HOSTIF: Accepted connection
```

If the VCP cannot connect to the VFP, the VFP syslog file does not display the `RPIO` and `HOSTIF` messages.

Action

Run the `request chassis fpc slot 0 restart` command from the VCP CLI. If an `FPC is in transition` error message is displayed, then run `restart chassis-control`.

If these commands do not correct the problem, verify whether the VCP can ping the VFP from the `routing-instance __juniper_private1__`. The three management interfaces (for the host, VCP VM, and VFP VM) connected to the internal bridge should be able to reach each other. For example:

```
root> ping 128.0.0.16 routing-instance __juniper_private1__
PING 128.0.0.16 (128.0.0.16): 56 data bytes
64 bytes from 128.0.0.16: icmp_seq=0 ttl=64 time=0.273 ms
64 bytes from 128.0.0.16: icmp_seq=1 ttl=64 time=0.606 ms
```

If the VCP cannot ping the VFP, perform these tasks:

1. Use the `brctl show` command to verify the bridge configuration and connected interfaces.
2. Verify that the startup configuration file is correct.
3. Verify that the VFP and the VCP VMs are up and the correct IP addresses are assigned.
4. Restart the FPC from the VCP VM.
5. Restart the chassis management process from VCP VM.
6. Stop and start the VFP VM.
7. Stop and start the VCP VM.
8. Restart the host.

If the problem persists, contact the Juniper Networks Technical Assistance Center (JTAC).

RELATED DOCUMENTATION

- Connecting to VMs | 60
- Verifying Whether VMs Are Running | 142
- Viewing VFP Statistics | 143
- Viewing VFP Log Files | 145

Verifying BIOS Settings for SR-IOV

If you are having problems with the SR-IOV ports, make sure BIOS has the following settings:

- SR-IOV is enabled.
- VT-d is enabled.
- Hyperthreading is enabled.

We recommend that you verify the process with the vendor because different systems have different methods to access and change BIOS settings.

RELATED DOCUMENTATION

- vMX Package Contents | 18
- Installing vMX on KVM | 20
- Deploying and Managing vMX | 52