

vMX

---

# vMX Getting Started Guide for Contrail

Published  
2020-12-28

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://www.juniper.net)

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*vMX vMX Getting Started Guide for Contrail*

Copyright © 2020 Juniper Networks, Inc. All rights reserved.

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 | v](#)

1

## [Overview](#)

[Overview of vMX and Contrail | 2](#)

[Requirements for vMX on Contrail | 2](#)

[vMX Package Contents | 6](#)

2

## [Installing vMX on Contrail](#)

[Prepare vMX Installation on Contrail | 10](#)

[Preparing the Controller Node for vMX | 11](#)

[Preparing the Compute Nodes | 12](#)

[Preparing the Compute Node for vMX | 12](#)

[Configuring the Compute Node for SR-IOV Interfaces for Contrail 4.0 | 14](#)

[Configuring the Compute Node for SR-IOV Interfaces for Contrail 3.0 | 16](#)

[Installing vMX on Contrail | 20](#)

[Installing vMX on Contrail | 20](#)

[Setting Up the vMX Configuration File | 20](#)

[Specifying vMX Configuration File Parameters | 22](#)

[Creating OpenStack Flavors | 23](#)

[Installing vMX Images | 24](#)

[Starting a vMX Instance | 25](#)

[Modifying the Initial Junos OS Configuration | 25](#)

[Launching the vMX Instance | 25](#)

[vMX Heat Templates for Contrail | 27](#)

[vMX Heat Templates | 27](#)

[Modifying the Ports in a Heat Template for an Instance | 30](#)

[Defining a Topology | 32](#)

[Creating a vMX Instance in an Existing WAN Network | 33](#)

## 3

Connecting vMX Instances with vRouter | 33

## Configuring vMX Chassis-Level Features

Configuring the Number of Active Ports on vMX | 37

Naming the Interfaces | 37

Configuring the Media MTU | 38

Enabling Performance Mode or Lite Mode | 39

Tuning Performance Mode | 41

lite-mode | 42

performance-mode | 44

## 4

## Class of Service for vMX

CoS on vMX Overview | 48

CoS Features and Limitations on vMX | 50

Configuring Hierarchical CoS on vMX | 52

Enabling Flexible Queuing | 52

Mapping Forwarding Classes to Queues on vMX | 53

Configuring Traffic Control Profiles for vMX | 53

Configuring Schedulers on vMX | 53

Example: Configuring Hierarchical CoS on vMX | 55

Requirements | 55

Overview | 55

Configuration | 55

Configuring Four-Level Hierarchical Scheduling on vMX | 61

Packet Loss Priority and Drop Profiles on vMX | 62

Managing Congestion Using Drop Profiles and Packet Loss Priorities on vMX | 64

Configuring Drop Profiles | 64

Configuring Schedulers with Drop Profiles | 65

# About This Guide

Use this guide to install the virtual MX router in the Juniper Networks Contrail 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

## Overview

---

[Overview of vMX and Contrail | 2](#)

[Requirements for vMX on Contrail | 2](#)

[vMX Package Contents | 6](#)

---

# Overview of vMX and Contrail

vMX can be launched from Juniper Networks Contrail on Ubuntu systems with OpenStack. vMX can be used with Contrail vRouter in DPDK mode or kernel mode.

You must perform basic Contrail installation on the controller and compute nodes as described in the *Contrail Getting Started Guide* for [Release 3.0](#) or [Release 4.0](#) (starting with Junos OS Release 17.4R1).

You must perform basic OpenStack installation on the controller and compute nodes. At a minimum, you must install a controller node and a compute node before installing vMX.

## RELATED DOCUMENTATION

[Prepare vMX Installation on Contrail | 10](#)

[Installing vMX on Contrail | 20](#)

# Requirements for vMX on Contrail

[Table 1 on page 3](#) lists the hardware requirements.

**Table 1: Hardware Requirements**

Description	Value
Sample system configuration	<p>Intel Ivy Bridge processors or later</p> <p>Example of Ivy Bridge processor: Intel Xeon E5-2667 v2 @ 3.30 GHz 25 MB Cache</p> <p>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. Support for unmodified ixgbe driver and i40e driver is available from Junos OS Release 18.4R1 onwards.</p> <p><b>NOTE:</b> For Junos OS Releases prior to 18.4R1, a modified IXGBE driver must be installed to use SR-IOV.</p> <p><b>NOTE:</b> Support for modified drivers for i40e is not available starting in Junos OS Release 19.1 and later releases.</p>



Table 1: Hardware Requirements *(Continued)*

Description	Value
Number of cores	For lite mode: Minimum of 4
<b>NOTE:</b> Performance mode is the default mode, and the minimum value is based on one port.	<ul style="list-style-type: none"> <li>• 2 for VCP</li> <li>• 3 for VFP</li> </ul> <p>For performance mode with low-bandwidth (virtio) or high-bandwidth (SR-IOV) applications: Minimum of 9</p> <ul style="list-style-type: none"> <li>• 1 for VCP</li> <li>• 8 for VFP</li> </ul> <p>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:</p> <ul style="list-style-type: none"> <li>• Without QoS—<math>(4 * \text{number-of-ports}) + 4</math></li> <li>• With QoS—<math>(5 * \text{number-of-ports}) + 4</math></li> </ul> <p><b>NOTE:</b> All VFP vCPUs must be in the same physical non-uniform memory access (NUMA) node for optimal performance.</p> <p>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.</p>

Table 1: Hardware Requirements *(Continued)*

Description	Value
Memory	For lite mode: Minimum of 8 GB
<b>NOTE:</b> Performance mode is the default mode.	<ul style="list-style-type: none"> <li>• 4 GB for VCP</li> <li>• 4 GB for VFP</li> </ul>
	For performance mode: Minimum of 16 GB
	<ul style="list-style-type: none"> <li>• 4 GB for VCP</li> <li>• 12 GB for VFP</li> </ul>
	Additional 2 GB recommended for host OS
Storage	Minimum storage requirements: <ul style="list-style-type: none"> <li>• 40 GB for VCP</li> <li>• 4 GB for VFP</li> </ul>
vNICs	SR-IOV  <b>NOTE:</b> 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.
	VMXNET3

[Table 2 on page 6](#) lists the software requirements for Contrail on Ubuntu.

**Table 2: Software Requirements for Ubuntu**

Description	Value
Operating system	<ul style="list-style-type: none"> <li>Contrail 3.0</li> <li>OpenStack Liberty</li> <li>Ubuntu 14.04.4</li> <li>Linux 3.13.0-85-generic</li> <li>Starting with Junos OS Release 17.4R1</li> <li>Contrail 4.0</li> <li>OpenStack Newton</li> <li>Ubuntu 16.04</li> <li>Linux 4.4.0-62-generic</li> </ul>

**Release History Table**

Release	Description
18.4R1	Support for unmodified ixgbe driver and i40e driver is available from Junos OS Release 18.4R1 onwards.

**RELATED DOCUMENTATION**

[Prepare vMX Installation on Contrail](#) | 10

[Installing vMX on Contrail](#) | 20

## vMX Package Contents

Table 3 on page 7 lists the contents of the vMX KVM package that you download from the [vMX Download Software page](#) for use with Contrail.

Table 3: vMX Package Contents

Filename	Description
images/junos-vmx-x86-64-*.qcow2	Software image file for VCP.
images/vFPC*.img	Software image file for VFP.
openstack/1vmx_contrail.yaml	Sample Heat template for one vMX instance.
openstack/1vmx_contrailv2.yaml	<p>Sample Heat template for one vMX instance (for OpenStack Newton).</p> <p><b>NOTE:</b> Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 file for the Heat templates.</p>
openstack/scripts/vmx.conf	Configuration file for defining vMX parameters. See <a href="#">"Installing vMX on Contrail" on page 20</a> for more information.
openstack/scripts/vmx_osp_create_flavor.py	Script for creating VCP and VFP OpenStack flavors.
openstack/scripts/vmx_osp_images.sh	Script for installing VCP and VFP OpenStack glance images.
scripts/openstack/vmx-templates/liberty/vmx_baseline.conf	Junos OS configuration file that is loaded during boot.

Release History Table

Release	Description
17.4R1	Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 file for the Heat templates.

## RELATED DOCUMENTATION

[Prepare vMX Installation on Contrail | 10](#)

[Installing vMX on Contrail | 20](#)

# 2

CHAPTER

## Installing vMX on Contrail

---

[Prepare vMX Installation on Contrail | 10](#)

[Installing vMX on Contrail | 20](#)

[vMX Heat Templates for Contrail | 27](#)

---

# Prepare vMX Installation on Contrail

## IN THIS SECTION

- [Preparing the Controller Node for vMX | 11](#)
- [Preparing the Compute Nodes | 12](#)

If you are running vRouter in DPDK mode, keep the following points in mind when preparing the OpenStack environment:

- For Contrail 3.0, make sure you configure the **testbed.py** file for provisioning a Contrail node to use DPDK. See [Preparing the testbed.py File for Provisioning a Contrail Cluster Node with DPDK](#).

For Contrail 4.0, make sure you provision a Contrail node with DPDK. See [Preparing the server.json File for Provisioning a Contrail 4.0 Cluster Node with DPDK](#).

- Enable **iommu=pt** in **/etc/default/grub** on the compute node.
- If using the contrail-3.0.3.0-69 release, upgrade the compute nodes to the libvirt 1.2.16 packages manually with the following commands and restart the nova-compute and libvirt-bin services.

```
dpkg -i /opt/contrail/contrail_install_repo/libvirt-  
bin_1.2.16-2ubuntu11.15.10.4~cloud0_amd64.deb  
dpkg -i /opt/contrail/contrail_install_repo/  
libvirt0_1.2.16-2ubuntu11.15.10.4~cloud0_amd64.deb
```

- Enable Huge Pages for all VMs to allow them to transmit traffic.
- Verify if DPDK is enabled on the compute nodes with the **contrail-status** command. The status of **contrail-vrouter-dpdk** is active.
- Make sure the NIC cards used for the contrail-vhost network (that connects compute and controller nodes) and the SR-IOV network (that is provided to VFD process) are different network cards. Use the **ethtool -i** command to verify that bus-info field is from different cards.

To prepare the OpenStack environment to install vMX, perform these tasks:

## Preparing the Controller Node for vMX

To prepare the controller node:

1. Configure the controller node to enable Huge Pages and CPU affinity by editing the **scheduler\_default\_filters** parameter in the **/etc/nova/nova.conf** file. Make sure the following filters are present:

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,Compute
Filter,
ComputeCapabilitiesFilter,ImagePropertiesFilter,CoreFilter,NUMATopologyFilter,
AggregateInstanceExtraSpecsFilter,PciPassthroughFilter,ServerGroupAffinityFilt
er,
ServerGroupAntiAffinityFilter
```

**NOTE:** We recommend this configuration whether CPU pinning is on or off. Huge Pages is always on.

Restart the scheduler service with the **service nova-scheduler restart** command.

2. Update the default quotas.

```
nova quota-class-update --cores 100 default
nova quota-class-update --ram 102400 default
nova quota-class-update --instances 100 default
```

**NOTE:** We recommend these default values, but you can use different values if they are appropriate for your environment. Make sure the default quotas have enough allocated resources.

Verify the changes with the **nova quota-defaults** command.

3. (For Contrail 3.0 only) To enable SR-IOV interfaces on the controller node, apply the patch by downloading the vMX KVM software package from the [vMX Download Software page](#).



Uncompress the package, change directory to root, and apply the patches from the **openstack/kilo/openstack\_vfd\_patches** directory.

```
tar xvf package-name
cd /
patch -p1 < package-location/openstack/kilo/openstack_vfd_patches/heat.patch
patch -p1 < package-location/openstack/kilo/openstack_vfd_patches/
neutron.patch
```

4. (For Contrail 4.0, starting with Junos OS Release 17.4R1) To enable SR-IOV interfaces on the controller node, verify the Contrail patch described in <https://bugs.launchpad.net/opencontrail/+bug/1709822> has been applied. Otherwise, you might see the **Unknown Neutron Exception** error message.

## Preparing the Compute Nodes

### IN THIS SECTION

- [Preparing the Compute Node for vMX | 12](#)
- [Configuring the Compute Node for SR-IOV Interfaces for Contrail 4.0 | 14](#)
- [Configuring the Compute Node for SR-IOV Interfaces for Contrail 3.0 | 16](#)

### Preparing the Compute Node for vMX

To prepare the compute node:

1. Configure each compute node to enable IOMMU (**intel\_iommu=on**) and support Huge Pages at boot time and reboot. Add the configuration to **/etc/default/grub** under the **GRUB\_CMDLINE\_LINUX\_DEFAULT** parameter.

**NOTE:** If you are running vRouter in DPDK mode or you are using VLAN provider OVS networks for virtio, you must also include **iommu=pt**.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on
iommu=pt hugepagesz=2M hugepages=24576 default_hugepagesz=2M"
```

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

2. (Optional) Ensure that VMs with Huge Pages enabled are successfully deployed.

If you see the **unable to create backing store for hugepages: Permission denied** message in the **/var/log/nova/nova-compute.log** file on compute nodes, perform these tasks:

- a. Create a new directory for libvirt to mount or use Huge Pages.

```
mkdir -p /run/hugepages/kvm/
```

- b. Mount huge table fs over this directory.

```
mount -t hugetlbfs hugetlbfs-kvm /run/hugepages/kvm/
```

- c. Provide this directory input explicitly in **/etc/libvirt/qemu.conf** as the mount location for hugetlbfs.

```
# cat /etc/libvirt/qemu.conf | grep hugetlbfs
# If provided by the host and a hugetlbfs mount point is configured,
hugetlbfs_mount = ["/dev/hugepages2M", "/dev/hugepages1G"] hugetlbfs_mount
= "/run/hugepages/kvm"
```

- d. Add **KVM\_HUGEPAGES=1** in **etc/default/qemu-kvm**.

```
# cat /etc/default/qemu-kvm
# Set to 1 to enable KSM, 0 to disable KSM, and AUTO to use default
settings.
# After changing this setting restart the qemu-kvm service.
KSM_ENABLED=AUTO
SLEEP_MILLISECS=200
# To load the vhost_net module, which in some cases can speed up
# network performance, set VHOST_NET_ENABLED to 1.
VHOST_NET_ENABLED=0

# Set this to 1 if you want hugepages to be available to kvm under
```

```
# /run/hugepages/kvm
KVM_HUGEPAGES=1
```

e. Reboot the compute node.

After the reboot, verify that Huge Pages are allocated with the `cat /proc/meminfo | grep Huge` command.

## Configuring the Compute Node for SR-IOV Interfaces for Contrail 4.0

Starting in Junos OS Release 17.4R1, the following procedure is only for Contrail 4.0

(For Contrail 4.0) To configure the SR-IOV interfaces:

1. Enable VT-d in BIOS. (We recommend that you verify the process with the vendor because different systems have different methods to enable VT-d.)
2. Enable ASPM in BIOS.  
To verify that ASPM is enabled, use the `lspci -vv | grep ASPM | grep Enabled` command.
3. (Optional) If you are using Intel Ivy Bridge processors, add `options kvm-intel enable_apicv=N` to the `/etc/modprobe.d/kvm-intel.conf` file and reboot.
4. Load the modified IXGBE driver.

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

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

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

```
tar xvf package-name
cd package-location/drivers/ixgbe-3.19.1/src
make
rmmod ixgbe
make install
```

Create the virtual function (VF) on the physical device. vMX needs only one VF for the SR-IOV traffic (for example, eth4). The following line specifies that there is no VF for eth2 (first IXGBE NIC) and one VF for eth4 (second IXGBE NIC).

```
modprobe ixgbe max_vfs=0,1
```

Specify these values according to your configuration.

Verify the driver version (3.19.1) on the eth2 and eth4 interfaces.

```
# ethtool -i eth2
# ethtool -i eth4
```

5. To make sure that the interfaces are up and SR-IOV traffic can pass through them, execute these commands to complete the configuration.

```
# ifconfig eth4 up
# ifconfig eth4 promisc
# ifconfig eth4 allmulti
# ifconfig eth4 mtu 9192
# ip link set eth4 vf 0 spoofchk off
```

To confirm the status of VF 0:

```
# ip link show eth4
```

```
19: eth4: <BROADCAST,MULTICAST,ALLMULTI,PROMISC,UP,LOWER_UP> mtu
9192 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether 08:9e:01:82:ab:39 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking off, link-state
    auto
```

The example only displays VF 0.

6. Edit the `/etc/nova/nova.conf` file to add the PCI passthrough allowlist entry for the SR-IOV NIC card.

```
#pci_passthrough_whitelist =
pci_passthrough_whitelist = {"address":"interface-address",
"physical_network": "physical-network"}
```

For example, this entry adds an entry for the SR-IOV interface card (eth4) for the physical network physnet2.

```
#pci_passthrough_whitelist =
pci_passthrough_whitelist = {"address": "0000:04:10.0",
"physical_network": "physnet2"}
```

Restart the compute node service with the **service nova-compute restart** command.

## Configuring the Compute Node for SR-IOV Interfaces for Contrail 3.0

**NOTE:** The following procedure is only for Contrail 3.0 and VFD agent.

(For Contrail 3.0 only) To configure the SR-IOV interfaces:

1. Enable VT-d in BIOS. (We recommend that you verify the process with the vendor because different systems have different methods to enable VT-d.)
2. Enable ASPM in BIOS.

To verify that ASPM is enabled, use the **lspci -vv | grep ASPM | grep Enabled** command.

3. (Optional) If you are using Intel Ivy Bridge processors, add **options kvm-intel enable\_apicv=N** to the **/etc/modprobe.d/kvm-intel.conf** file and reboot.
4. Apply the patch by downloading the vMX KVM software package from the [vMX Download Software page](#).

Uncompress the package, change directory to root, and apply the patches from the **openstack/kilo/openstack\_vfd\_patches** directory.

```
tar xvf package-name
cd /
patch -p1 < package-location/openstack/kilo/openstack_vfd_patches/nova.patch
patch -p1 < package-location/openstack/kilo/openstack_vfd_patches/sriov.patch
```

5. Unload the IXGBEVF driver.

```
modprobe -r ixgbevf
```

Load the vfio-pci module.

```
modprobe vfio-pci
```

Create the virtual function (VF) on the physical device. vMX needs only one VF for the SR-IOV traffic (for example, eth4). Note that we are creating 32 VFs, though we are using only one.

```
echo '32' > /sys/class/net/eth4/device/sriov_numvfs
```

To verify that the VF was created, the output of the `lspci -nn | grep Ether | grep Virtual | wc -l` command displays **32**.

6. To make sure that the interfaces are up and SR-IOV traffic can pass through them, execute these commands to complete the configuration.

```
# ifconfig eth4 up
# ifconfig eth4 promisc
# ifconfig eth4 allmulti
# ifconfig eth4 mtu 9192
# ip link set eth4 vf 0 spoofchk off
```

To confirm the status of VF 0:

```
# ip link show eth4
```

```
19: eth4: <BROADCAST,MULTICAST,ALLMULTI,PROMISC,UP,LOWER_UP> mtu
9192 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether 08:9e:01:82:ab:39 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, spoof checking off, link-state
    auto
```

The example only displays VF 0, but all 32 VFs appear in the output.

7. Edit the `/etc/nova/nova.conf` file to add the PCI passthrough allowlist entry for the SR-IOV NIC card.

```
#pci_passthrough_whitelist =
pci_passthrough_whitelist = {"address": "interface-address",
    "physical_network": "physical-network"}
```

For example, this entry adds an entry for the SR-IOV interface card (eth4) for the physical network physnet2.

```
#pci_passthrough_whitelist =
pci_passthrough_whitelist = {"address": "0000:04:10.0",
"physical_network": "physnet2"}
```

Add the VF-agent entry to **nova.conf**.

```
[vf_agent]
use_vf_agent=True
```

Restart the compute node service with the **service nova-compute restart** command.

8. Install the VFD agent. Use the VFD 1.14 Debian package version.

```
apt-get update
apt-get install python-docopt
apt-get clean
apt-get update
dpkg -i attlrvfd_1.14-1_amd64.deb
```

Copy **vfd.cfg** from the sample file.

```
cp /etc/vfd/vfd.cfg.sample /etc/vfd/vfd.cfg
```

Edit the **default\_mtu** value and the **pciids id** values in the **vfd.cfg** file.

```
"default_mtu": <value>,

"pciids": [
    { "id": "<pciid1>" },
    { "id": "<pciid2>", "mtu": <value> },
    ...
]
```

For example, the edited values might resemble the following with the ID of the physical function (PF):

```
"default_mtu": 9001,

"pciids": [
  { "id": "0000:04:00.0" },
]
```

Start the VFD process with the **service vfd start** command.

Verify that the process has started successfully with the **service vfd status** command.

**NOTE:** While the VFD process is running, the interfaces attached to VFD are not displayed with **ifconfig** or **ip link** commands. Use the **iplex show all** command to verify that the link for the PF is up.

9. Ensure that the correct permissions are provided for VFIO. Add the **"/dev/vfio/vfio"** entry to **/etc/libvirt/qemu.conf** under the **cgroup\_device\_acl** parameter.

```
cgroup_device_acl = [
  "/dev/null", "/dev/full", "/dev/zero",
  "/dev/random", "/dev/urandom",
  "/dev/ptmx", "/dev/kvm", "/dev/kqemu", "/dev/rtc",
  "/dev/hpet", "/dev/net/tun", "/dev/vfio/vfio",
]
```

Restart libvirtd with the **service libvirt-bin restart** command.

#### Release History Table

Release	Description
17.4R1	Starting in Junos OS Release 17.4R1, the following procedure is only for Contrail 4.0

#### RELATED DOCUMENTATION

[Requirements for vMX on Contrail | 2](#)

[vMX Package Contents | 6](#)



# Installing vMX on Contrail

## IN THIS SECTION

- Installing vMX on Contrail | 20
- Creating OpenStack Flavors | 23
- Installing vMX Images | 24
- Starting a vMX Instance | 25

Read this topic to understand how to install the virtual MX router in the Juniper Networks Contrail environment.

## Installing vMX on Contrail

### IN THIS SECTION

- Setting Up the vMX Configuration File | 20
- Specifying vMX Configuration File Parameters | 22

After preparing the OpenStack environment, we need to create nova flavors and glance images for the VCP and VFP VMs. Scripts create the flavors and images based on information provided in the startup configuration file.

### Setting Up the vMX Configuration File

The parameters required to configure vMX are defined in the startup configuration file (**vmx.conf**).

To set up the configuration file:

1. Download the vMX KVM software package from the [vMX Download Software page](#) and uncompress the package.

**`tar xvf package-name`**

2. Change directory to the location of the files.

**`cd package-location/openstack/scripts`**

3. Edit the **vmx.conf** text file with a text editor to create the flavors for a single vMX instance.

Based on your requirements, ensure the following parameters are set properly in the vMX configuration file:

- **re-flavor-name**
- **pfe-flavor-name**
- **vcpus**
- **memory-mb**

### Sample vMX Startup Configuration File

Here is a sample vMX startup configuration file for OpenStack:

```
#Configuration on the host side - management interface, VM images etc.
HOST:
    virtualization-type      : openstack
    cpu-pinning              : on
    compute                  : compute1,compute2

---
#vRE VM parameters
CONTROL_PLANE:
    re-flavor-name          : re-test
    vcpus                   : 2
    memory-mb               : 4096

---
#vPFE VM parameters
FORWARDING_PLANE:
    pfe-flavor-name         : pfe-test
    memory-mb               : 12288
    vcpus                   : 7
```

## Specifying vMX Configuration File Parameters

### IN THIS SECTION

- [Configuring the Host | 22](#)
- [Configuring the VCP VM | 22](#)
- [Configuring the VFP VM | 23](#)

The parameters required to configure vMX are defined in the startup configuration file (**vmx.conf**). The startup configuration file generates a file that is used to create flavors. To create new flavors with different **vcpus** or **memory-mb** parameters, you must change the corresponding **re-flavor-name** or **pfe-flavor-name** parameter before creating the new flavors.

To customize the configuration, perform these tasks:

### Configuring the Host

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

- **virtualization-type**—Mode of operation; must be **openstack**.
- **cpu-pinning**—Specifies whether VCP and VFP are pinned to vCPU on host. By default, **cpu-pinning** is on and we recommend this setting.
- **compute**—(Optional) Names of the compute node on which to run vMX instances in a comma-separated list. If this parameter is specified, it must be a valid compute node. If this parameter is specified, vMX instance launched with flavors are only run on the specified compute nodes.

If this parameter is not specified, the output of the **nova hypervisor-list** command provides the list of compute nodes on which to run vMX instances.

### Configuring the VCP VM

To configure the VCP VM, you must provide the flavor name.

**NOTE:** We recommend unique values for the **re-flavor-name** parameter because OpenStack can create multiple entries with the same name.

To configure the VCP VM, navigate to **CONTROL\_PLANE** and specify the following parameters:

- **re-flavor-name**—Name of the nova flavor.
- **vcpus**—Number of vCPUs for the VCP; minimum is 2.
- **memory-mb**—Amount of memory for the VCP; minimum is 4 GB.

### Configuring the VFP VM

To configure the VFP VM, you must provide the flavor name. Based on your requirements, you might want to change the memory and number of vCPUs. See [Hardware Requirements on page 3](#) for minimum hardware requirements.

To configure the VFP VM, navigate to **FORWARDING\_PLANE** and specify the following parameters:

- **pfe-flavor-name**—Name of the nova flavor.
- **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).

**NOTE:** If you specify less than 7 vCPUs, the VFP automatically switches to lite mode.

## Creating OpenStack Flavors

To create flavors for the VCP and VFP, you must execute the script on the vMX startup configuration file (**vmx.conf**).

To create OpenStack flavors:

1. Run the **vmx\_osp\_create\_flavor.py** with the startup configuration file to generate the **vmx\_osp\_flavors.sh** file that creates flavors.  

```
./vmx_osp_create_flavor.py vmx.conf
```
2. Execute the **vmx\_osp\_flavors.sh** to create flavors.  

```
sh vmx_osp_flavors.sh
```

**NOTE:** If you are running on a vRouter in DPDK mode, you must enable Huge Pages for the VCP flavor with the **nova flavor-key re-flavor-name set hw:mem\_page\_size=2048** command.

## Installing vMX Images

To install the vMX software images for the VCP and VFP, you can execute the `vmx_osp_images.sh` script. The script adds the VCP image in qcow2 format and the VFP file in vmdk format.

To install the VCP and VFP images:

1. Download the vMX KVM software package from the [vMX page](#) and uncompress the package.  
`tar xvf package-name`
2. Verify the location of the software images from the uncompressed vMX package. See [vMX Package Contents on page 7](#).  
`ls package-location/images`
3. Change directory to the location of the vMX OpenStack script files.  
`cd package-location/openstack/scripts`
4. Run the `vmx_osp_images.sh` script to install the glance images.  
`sh vmx_osp_images.sh vcp-image-name vcp-image-location vfp-image-name vfp-image-location`

**NOTE:** You must specify the parameters in this order.

- **vcp-image-name**—Name of the glance image.
- **vcp-image-location**—Absolute path to the `junos-vmx-x86-64*.qcow2` file for launching VCP.
- **vfp-image-name**—Name of the glance image.
- **vfp-image-location**—Absolute path to the `vFPC-*.img` file for launching VFP.

For example, this command installs the VCP image as `re-test` from the `/var/tmp/junos-vmx-x86-64-17.3R1.8.qcow2` file and the VFP image as `fpc-test` from the `/var/tmp/vFPC-20170817.img` file.

```
sh vmx_osp_images.sh re-test /var/tmp/junos-vmx-x86-64-17.3R1.8.qcow2 fpc-test /var/tmp/vFPC-20170817.img
```

To view the glance images, use the `glance image-list` command.

## Starting a vMX Instance

### IN THIS SECTION

- [Modifying the Initial Junos OS Configuration | 25](#)
- [Launching the vMX Instance | 25](#)

To start a vMX instance, perform these tasks:

### Modifying the Initial Junos OS Configuration

When you start the vMX instance, the Junos OS configuration file found in **scripts/openstack/vmx-templates/liberty/vmx\_baseline.conf** is loaded. If you need to change this configuration, make any changes in this file before starting the vMX.

**NOTE:** If you create your own *vmx\_baseline.conf* file or move the file, make sure that the **scripts/openstack/vmx-templates/liberty/re.yaml** references the correct path.

### Launching the vMX Instance

To create and start the vMX instance:

1. Modify these parameters in the corresponding environment file for your configuration. Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 version of the Heat templates.
  - **net\_id1**—Network ID of the existing neutron network used for the WAN port. Use the **neutron net-list** command to display the network ID.
  - **public\_network**—Network ID of the existing neutron network used for the management (fxp0) port. Use the **neutron net-list | grep public** command to display the network ID.
  - **fpc\_img**—Name of the glance image for the VFP; same as the **vfp-image-name** parameter specified when running the script to install the vMX images.
  - **fpc\_flav**—Name of the nova flavor for the VFP; same as the **pfe-flavor-name** parameter specified in the vMX configuration file.
  - **junos\_flav**—Name of the nova flavor for the VCP; same as the **re-flavor-name** parameter specified in the vMX configuration file.

- **junos\_img**—Name of the glance image for the VCP; same as the **vcp-image-name** parameter specified when running the script to install the vMX images.
  - **project\_name**—Any project name. All resources will use this name as the prefix.
  - **gateway\_ip**—Gateway IP address.
2. Start the vMX instance with the **heat stack-create -f 1vmx\_contrail.yaml -e 1vmx\_contrail.env vmx-name** command.  
This sample configuration starts a single vMX instance with one WAN port and one FPC.
  3. Verify that the vMX instance is created with the **heat stack-list | grep vmx-name** command.
  4. Verify that the VCP and VFP VMs exist with the **nova-list** command.
  5. Access the VCP VM with the **nova get-vnc-console nova-id novnc** command, where *nova-id* is the ID of the instance on the WAN network.

**NOTE:** You must shut down the vMX instance before you reboot host server using the request system halt command.



**CAUTION:** When you deploy a vMX instance with DPDK-enabled vRouter, the virtio interface (int interface) might be down if you restart vRouter service.

Workaround:

- Define metadata tunable upfront during the deployment of the vMX instance. The support for the new tunable is available from Junos OS Release 19.4 onwards.
- Add the metadata variable in RE yaml file or in FPC yaml file. Example: For updating RE yaml file, navigate to *package\_location/vmx/openstack/vmx-components/vms/re.yaml* and set the variable as shown below:

```
virtio_lockup_timeout: timeout-value
```

Example:

```
virtio_lockup_timeout: 10
```

In the example, lockup timeout value is set as 10 seconds. You can set the value in the range of 10 through 1000 seconds.

Release History Table

Release	Description
17.4R1	Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 version of the Heat templates.

RELATED DOCUMENTATION

<a href="#">Prepare vMX Installation on Contrail   10</a>
<a href="#">vMX Heat Templates for Contrail   27</a>

# vMX Heat Templates for Contrail

IN THIS SECTION

- [vMX Heat Templates | 27](#)
- [Modifying the Ports in a Heat Template for an Instance | 30](#)
- [Defining a Topology | 32](#)

Heat templates define vMX instances and topologies on contrail. Read this topic to understand how to create vMX instances that can be used with a topology template to define interconnection of vMX instances.

## vMX Heat Templates

IN THIS SECTION

- [vMX Instance Template | 28](#)
- [vMX Topology Template | 28](#)
- [Environment File | 29](#)



Heat templates define vMX instances and topologies on Contrail. The **vmx\_contrail.yaml** template defines a vMX instance, including the VCP, the VFP, and the bridge between them. It also defines any WAN ports and bridges. A topology template can have multiple vMX instances interconnected by bridges. Heat templates get their input parameters from the corresponding environment file.

You can download the vMX software package from the [vMX page](#) and uncompress the package for sample templates. Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 version of the Heat templates.

## vMX Instance Template

You can customize the **vmx\_contrail.yaml** or **vmx\_contrail\_sriov.yaml** templates to create vMX instances. The Heat template obtains its input parameters from the corresponding environment file.

In the vMX instance template, you modify these resources:

- **OS::Networking::VmxPort**—Defines the WAN port of the FPC for virtio interfaces as either `ge-x/x/x` or `xe/0/0/0`. The input parameters obtained from the environment file are the network ID of the network on which the WAN port is added (**vnetwork\_id**) and the name of the port (**pname**). The **stack\_name** does not change.
- **OS::Networking::VmxContrailSriovPort**—Defines the WAN port of the FPC for SR-IOV interfaces as either `ge-x/x/x` or `xe/0/0/0`. The input parameters obtained from the environment file are the network ID of the network on which the WAN port is added (**vnetwork\_id**) and the name of the port (**pname**). The **stack\_name** does not change.

You can find the sample instance templates at *package-location/openstack*. Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 file for the Heat templates.

## vMX Topology Template

You can customize the **vmx\_contrail.yaml** template to create vMX instances that can be used with a topology file to define how to interconnect vMX instances. The Heat template obtains its input parameters from the corresponding environment file.

In the topology template, you modify these resources:

- **OS::Networking::VmxNetContrail**—Defines the OVS bridge instance. The input parameters obtained from the environment file are the CIDR of the network (**net\_cidr**) and the name of the bridge (**bname**). The **stack\_name** is always set to `get_param: 'OS::stack_name'`.
- **OS::Nova::VmxContrail**—Defines the vMX instance. The input parameters obtained from the environment file are the network ID of the bridge (**net\_id1**) and the **stack\_name**.

You can find the sample instance templates at *package-location/openstack/vmx-topologies/contrail-topologies*.

## Environment File

The Heat template obtains its input parameters from the corresponding environment file.

The environment file for the vMX instance has the **net\_id1** parameter:

```
parameters:
  net_id1:
```

where **net\_id1** is the network ID of the existing neutron network with a given VLAN ID used for the WAN port.

The environment file for the topology has the **n1** parameter, which is the instance name.

The environment file has these parameter defaults:

```
parameter_defaults:
  public_network:
  fpc_img:
  fpc_flav:
  junos_flav:
  junos_img:
  project_name:
  gateway_ip:
```

where

- **public\_network**—Network ID of the existing neutron network used for the management (fxp0) port.
- **fpc\_img**—Name of the glance image for the VFP that was created by the script to install vMX images.
- **fpc\_flav**—Name of the nova flavor for the VFP that was created by the script to create OpenStack flavors.
- **junos\_flav**—Name of the nova flavor for the VCP that was created by the script to create OpenStack flavors.
- **junos\_img**—Name of the glance image for the VCP that was created by the script to install vMX images.
- **project\_name**—Any project name. All resources will use this name as the prefix.
- **gateway\_ip**—Gateway IP address.

## Modifying the Ports in a Heat Template for an Instance

You can modify the vMX instance template to add or delete ports. For example, the *package-location/openstack/1vmx\_contrail.yaml* file creates a single instance.

To add a port to the template:

1. Download the vMX KVM software package from the [vMX page](#) and uncompress the package.

```
tar xvf package-name
```

2. Change directory to the location of the files.

```
cd package-location/openstack
```

3. Edit the *1vmx\_contrail.yaml* file to modify the OS::Networking::VmxPort resource.

To add a virtio port, add another OS::Networking::VmxPort entry but change the port number to the next number, the **vnetwork\_id** parameter to net\_id2, and the **pname** parameter to the name of the new port. For example:

```
fpc0_ge_port2:
  type: OS::Networking::VmxPort
  properties:
    vnetwork_id: {get_param: net_id2}
    pname: fpc0_WAN_1
    stack_name: {get_param: stack_name}
```

**NOTE:** If you are adding virtio port entries (OS::Networking::VmxPort resource), make sure you do not include SR-IOV port entries (OS::Networking::VmxContrailSriovPort resource) because mixed interfaces are not supported.

4. Edit the *1vmx\_contrail.yaml* file to modify the OS::Networking::VmxContrailSriovPort resource.

To add another SR-IOV port, add another OS::Networking::VmxContrailSriovPort entry but change the port number to the next number, the **vnetwork\_id** parameter to the next number, and the **pname** parameter to the name of the new port. For example, if the template file already has two ports:

```
fpc0_ge_port3:
  type: OS::Networking::VmxContrailSriovPort
  properties:
    vnetwork_id: {get_param: net_id3}
    pname: fpc0_WAN_2
    stack_name: {get_param: stack_name}
    vlan_filter: [3501]
```

```

vlan_insert: true
vlan_strip: true
vlan_broadcast: true
vlan_unknown_multicast: true
vlan_unknown_unicast: true

```

**NOTE:** If you are adding SR-IOV port entries (OS::Networking::VmxContrailSriovPort resource), make sure you do not include virtio port entries (OS::Networking::VmxPort resource) because mixed interfaces are not supported.

5. If you have more than one port, you must modify the *package-location/openstack/vmx\_contrail.yaml* or *package-location/openstack/vmx\_contrail\_sriov.yaml* file to add another `{get_attr: [fpc0_ge_port1, port]}` entry (separated by a comma) in the `fpc0` resources section under `all_ports` properties for each additional port and changing the port number to the next number. For example, this configuration has two ports.

```

fpc0:
  type: OS::Nova::VmxFpcSingle
  properties:
    id: 0
    re0_ip: {get_attr: [fpc0_fixed_net, external_ip]}
    all_ports: [{get_attr: [fpc0_fixed_net, external_port]},
                {get_attr: [fpc0_fixed_net, internal_port]},
                {get_attr: [fpc0_ge_port1, port]},
                {get_attr: [fpc0_ge_port2, port]}]
    stack_name: {get_param: stack_name}

```

6. Add additional `net_id` parameters under the parameters section of the template file for each `vnetwork_id` that you added for the `VmxPort` or `VmxContrailSriovPort` resources.

```

parameters:
  net_id1: {description: ID of ge-network, type: string}
  net_id2: {description: ID of ge-network, type: string}
  net_id3: {description: ID of ge-network, type: string}

```

7. Add additional **net\_id** parameters under the parameters section of the corresponding environment file for each vnetwork\_id that you added for the VmxPort or VmxContrailSriovPort resources.

```
parameters:
  net_id1: network-id
  net_id2: network-id
  net_id3: network-id
```

**NOTE:** If you are adding SR-IOV interfaces, make sure you have prepared the controller and compute nodes properly. See ["Prepare vMX Installation on Contrail" on page 10](#).

8. Create the vMX instance with the **heat stack-create -f *heat-filename* -e *environment-filename* *vmx-name*** command.
9. Verify that the vMX instance is created with the **heat stack-list | grep *vmx-name*** command.

To delete a port, remove all the lines you included to add a port.

## SEE ALSO

[Prepare vMX Installation on Contrail | 10](#)

## Defining a Topology

### IN THIS SECTION

- [Creating a vMX Instance in an Existing WAN Network | 33](#)
- [Connecting vMX Instances with vRouter | 33](#)

You can create vMX instances that can be used with a topology template to define how to interconnect vMX instances. The Heat template obtains its input parameters from the corresponding environment file. The following examples demonstrate how to modify topologies.

## Creating a vMX Instance in an Existing WAN Network

The *package-location/openstack/vmx-topologies/contrail-topologies/1vmx1net\_contrail.yaml* file creates one vMX instance with virtio ports on an existing bridge.

To add a vMX instance to this template:

1. Copy the sample topology template file and environment file to the top-level directory.

```
cd openstack
cp vmx-topologies/contrail-topologies/1vmx1net_contrail.yaml ./
cp vmx-topologies/contrail-topologies/1vmx1net_contrail.env ./
```

2. Edit the *1vmx1net\_contrail.yaml* file to add another OS::Nova::VmxCtrail entry but change the last **stack\_name** parameter to the next number. For example:

```
vmx2:
  type: OS::Nova::VmxCtrail
  properties:
    net_id1: {get_attr: [br1, bridge_network]}
    stack_name: {list_join: ['- ', [{get_param: 'OS::stack_name'},
    {get_param: n2}]]}
```

3. Make sure the **net\_id1** parameter is listed under the parameters section of the corresponding environment file for the vMX instance (in this case, *1vmx1net\_contrail.env*).

```
parameters:
  net_id1: network-id
```

4. Start the vMX instances with the **heat stack-create -f *heat-filename* -e *environment-filename* *stack-name*** command.
5. Verify that the vMX instance is created with the **heat stack-list | grep *vmx-name*** command.

## Connecting vMX Instances with vRouter

The *package-location/openstack/vmx-topologies/contrail-topologies/2vmx1net\_contrail\_virt.yaml* file creates two instances with one port each that share a bridge.

To add a bridge that connects a second port on each vMX instance:

1. Copy the sample topology template file and environment file to the top-level directory.

```
cd openstack
cp vmx-topologies/contrail-topologies/2vmx1net_contrail_virt.yaml ./
cp vmx-topologies/contrail-topologies/2vmx1net_contrail_virt.env ./
```

2. Edit the **2vmx1net\_contrail\_virt.yaml** file to add another OS::Networking::VmxNetContrail entry but change the CIDR of the network (**net\_cidr**) and the name of the bridge (**bname**) to the next number. For example:

```
br2:
  type: OS::Networking::VmxNetContrail
  properties:
    net_cidr: {get_param: ge_cidr2}
    bname: br2
    stack_name: {get_param: 'OS::stack_name'}
```

3. To each OS::Nova::VmxContrail entry, add the next bridge network ID. For example:

```
vmx1:
  type: OS::Nova::VmxContrail
  properties:
    net_id1: {get_attr: [br1, bridge_network]}
    net_id2: {get_attr: [br2, bridge_network]}
    stack_name: {list_join: ['- ', [{get_param: 'OS::stack_name'},
    {get_param: n1}]]}

vmx2:
  type: OS::Nova::VmxContrail
  properties:
    net_id1: {get_attr: [br1, bridge_network]}
    net_id2: {get_attr: [br2, bridge_network]}
    stack_name: {list_join: ['- ', [{get_param: 'OS::stack_name'},
    {get_param: n2}]]}
```

4. Add the **ge\_cidr** parameter under the parameters section for the port.

```
parameters:
  ge_cidr1: {default: 10.10.15.0/24, description: CIDR GE net, type: string}
```

```
ge_cidr2: {default: 192.0.15.0/24, description: CIDR GE net, type: string}
```

5. Make sure the parameters section of the corresponding environment file for the vMX instance lists all the **net\_id** parameters.

```
parameters:
  net_id1: network-id
  net_id2: network-id
```

6. Start the vMX instances with the **heat stack-create -f *heat-filename* -e *environment-filename* *stack-name*** command.
7. Verify that the vMX instance is created with the **heat stack-list | grep *vmx-name*** command.

#### Release History Table

Release	Description
17.4R1	Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 version of the Heat templates.
17.4R1	Starting in Junos OS Release 17.4R1 for Contrail 4.0, use the v2 file for the Heat templates.

#### RELATED DOCUMENTATION

[Prepare vMX Installation on Contrail | 10](#)

[Installing vMX on Contrail | 20](#)



# 3

CHAPTER

## Configuring vMX Chassis-Level Features

---

[Configuring the Number of Active Ports on vMX](#) | 37

[Naming the Interfaces](#) | 37

[Configuring the Media MTU](#) | 38

[Enabling Performance Mode or Lite Mode](#) | 39

[Tuning Performance Mode](#) | 41

[lite-mode](#) | 42

[performance-mode](#) | 44

---

## 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 | 37](#)

---

[Configuring the Media MTU | 38](#)

---

[Enabling Performance Mode or Lite Mode | 39](#)

---

[Tuning Performance Mode | 41](#)

## 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 | 37](#)

[Configuring the Media MTU | 38](#)

[Enabling Performance Mode or Lite Mode | 39](#)

[Tuning Performance Mode | 41](#)

# 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 | 37](#)

[Naming the Interfaces | 37](#)

## 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
  PIC 0                               BUILTIN  BUILTIN  Virtual
```

Table 4 on page 40 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 4: 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 4: 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 | 41](#)
[lite-mode | 42](#)
[performance-mode | 44](#)

## 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 | 39](#)

[performance-mode | 44](#)

# lite-mode

## IN THIS SECTION

- [Syntax | 43](#)
- [Hierarchy Level | 43](#)
- [Release Information | 43](#)
- [Description | 43](#)
- [Options | 44](#)
- [Required Privilege Level | 44](#)

## 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](#) | 44

# performance-mode

#### IN THIS SECTION

- [Syntax](#) | 45
- [Hierarchy Level](#) | 45
- [Release Information](#) | 45
- [Description](#) | 45
- [Options](#) | 46
- [Required Privilege Level](#) | 46

## 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

<b>performance-mode</b>	<p>Enables performance mode.</p> <p>To disable performance mode, enable lite mode by including the <b>lite-mode</b> statement at the <b>[edit chassis fpc 0]</b> hierarchy level.</p>
<b>number-of-ucode-workers</b> <i>number-workers</i>	<p>Specifies the number of dedicated Workers for processing multicast and control traffic.</p> <ul style="list-style-type: none"> <li>• <b>Range:</b> 0 through 15</li> <li>• <b>Default:</b> 0 specifies that all available Workers are used to process all traffic.</li> </ul>

## Required Privilege Level

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

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

### RELATED DOCUMENTATION

| [lite-mode](#) | 42

# 4

CHAPTER

## Class of Service for vMX

---

CoS on vMX Overview | 48

CoS Features and Limitations on vMX | 50

Configuring Hierarchical CoS on vMX | 52

Example: Configuring Hierarchical CoS on vMX | 55

Configuring Four-Level Hierarchical Scheduling on vMX | 61

Packet Loss Priority and Drop Profiles on vMX | 62

Managing Congestion Using Drop Profiles and Packet Loss Priorities on vMX |  
64

---

# 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 | 50](#)

[Packet Loss Priority and Drop Profiles on vMX | 62](#)

# CoS Features and Limitations on vMX

## IN THIS SECTION

- [Weighted Round-Robin of Subscriber Traffic on a Port Limitations | 51](#)

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 | 52](#)

[CoS on vMX Overview | 48](#)

## Configuring Hierarchical CoS on vMX

#### IN THIS SECTION

- [Enabling Flexible Queuing | 52](#)
- [Mapping Forwarding Classes to Queues on vMX | 53](#)
- [Configuring Traffic Control Profiles for vMX | 53](#)
- [Configuring Schedulers on vMX | 53](#)

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 | 55](#)

[CoS on vMX Overview | 48](#)

[CoS Features and Limitations on vMX | 50](#)

# Example: Configuring Hierarchical CoS on vMX

## IN THIS SECTION

- [Requirements | 55](#)
- [Overview | 55](#)
- [Configuration | 55](#)

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 | 56](#)
- [Applying Shaping and Scheduling to VLANs | 56](#)

## 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
voicel 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 voicel scheduler
sched_voicel
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_voicel 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 | 52](#)

[CoS on vMX Overview | 48](#)

[CoS Features and Limitations on vMX | 50](#)

[Configuring Hierarchical CoS on vMX | 52](#)

# 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 | 48](#)

[CoS Features and Limitations on vMX | 50](#)

[Configuring Hierarchical CoS on vMX | 52](#)

# Packet Loss Priority and Drop Profiles on vMX

## IN THIS SECTION

- [Limitations | 63](#)

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 | 64](#)

[CoS on vMX Overview | 48](#)

[CoS Features and Limitations on vMX | 50](#)

# Managing Congestion Using Drop Profiles and Packet Loss Priorities on vMX

## IN THIS SECTION

- [Configuring Drop Profiles | 64](#)
- [Configuring Schedulers with Drop Profiles | 65](#)

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 | 62](#)

---

[CoS on vMX Overview | 48](#)

---

[CoS Features and Limitations on vMX | 50](#)