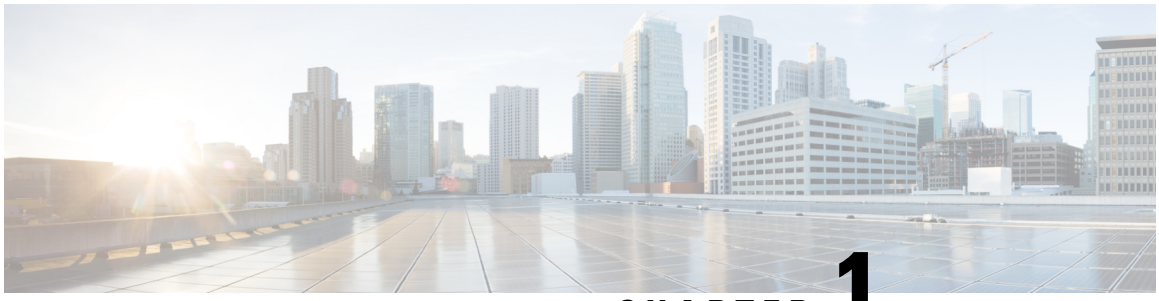
Cisco UCS Manager VM-FEX for KVM CLI Configuration Guide, Release 4.0

First Published: 2018-08-14

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

CONTENTS



CHAPTER 1

Introduction

Overview of Virtualization

Virtualization allows you to create multiple Virtual Machines (VMs) to run in isolation, side by side on the same physical machine.

Each virtual machine has its own set of virtual hardware (RAM, CPU, NIC) upon which an operating system and fully configured applications are loaded. The operating system sees a consistent, normalized set of hardware regardless of the actual physical hardware components.

In a virtual machine, both hardware and software are encapsulated in a single file for rapid provisioning and moving between physical servers. You can move a virtual machine, within seconds, from one physical server to another for zero-downtime maintenance and continuous workload consolidation.

The virtual hardware makes it possible for many servers, each running in an independent virtual machine, to run on a single physical server. The advantages of virtualization include better use of computing resources, greater server density, and seamless server migration.

Overview of Cisco Virtual Machine Fabric Extender

A virtualized server implementation consists of one or more VMs that run as guests on a single physical server. The guest VMs are hosted and managed by a software layer called the hypervisor or virtual machine manager (VMM). Typically, the hypervisor presents a virtual network interface to each VM and performs Layer 2 switching of traffic from a VM to other local VMs or to another interface to the external network.

Working with a Cisco virtual interface card (VIC) adapter, the Cisco Virtual Machine Fabric Extender (VM-FEX) bypasses software-based switching of VM traffic by the hypervisor for external hardware-based switching in the fabric interconnect. This method reduces the load on the server CPU, provides faster switching, and enables you to apply a rich set of network management features to local and remote traffic.

VM-FEX extends the IEEE 802.1Qbh port extender architecture to the VMs by providing each VM interface with a virtual Peripheral Component Interconnect Express (PCIe) device and a virtual port on a switch. This solution allows precise rate limiting and quality of service (QoS) guarantees on the VM interface.

Important In Cisco UCS Manager Release 4.0(1), VM-FEX is not supported with Cisco UCS 6454 Fabric Interconnects.

Virtualization with a Virtual Interface Card Adapter

A Cisco VIC adapter is a converged network adapter (CNA) that is designed for both bare metal and VM-based deployments. The VIC adapter supports static or dynamic virtualized interfaces, which includes up to 116 virtual network interface cards (vNICs).

There are two types of vNICs used with the VIC adapter—static and dynamic. A static vNIC is a device that is visible to the OS or hypervisor. Dynamic vNICs are used for VM-FEX by which a VM is connected to a veth port on the Fabric Interconnect.

VIC adapters support VM-FEX to provide hardware-based switching of traffic to and from virtual machine interfaces.

Single Root I/O Virtualization

Single Root I/O Virtualization (SR-IOV) allows multiple VMs running a variety of guest operating systems to share a single PCIe network adapter within a host server. SR-IOV allows a VM to move data directly to and from the network adapter, bypassing the hypervisor for increased network throughput and lower server CPU burden. Recent x86 server processors include chipset enhancements, such as Intel VT-x technology, that facilitate direct memory transfers and other operations required by SR-IOV.

The SR-IOV specification defines two device types:

- Physical Function (PF)—Essentially a static vNIC, a PF is a full PCIe device that includes SR-IOV capabilities. PFs are discovered, managed, and configured as normal PCIe devices. A single PF can provide management and configuration for a set of virtual functions (VFs).
- Virtual Function (VF)—Similar to a dynamic vNIC, a VF is a full or lightweight virtual PCIe device that provides at least the necessary resources for data movements. A VF is not managed directly but is derived from and managed through a PF. One or more VFs can be assigned to a VM.

SR-IOV is defined and maintained by the Peripheral Component Interconnect Special Interest Group (PCI-SIG), an industry organization that is chartered to develop and manage the PCI standard. For more information about SR-IOV, see the following URL:

https://www.intel.com/content/www/us/en/pci-express/pci-sig-sr-iov-primer-sr-iov-technology-paper.html

The Linux KVM hypervisor supports SR-IOV.

The following Cisco Virtual Interface Cards support SR-IOV with VM-FEX:

- Cisco UCS Virtual Interface Card 1240
- Cisco UCS Virtual Interface Card 1280

- Cisco UCS Virtual Interface Card 1225
- Cisco UCS Virtual Interface Card 1225T
- Cisco UCS Virtual Interface Card 1227
- Cisco UCS Virtual Interface Card 1227T
- Cisco UCS Virtual Interface Card 1340
- Cisco UCS Virtual Interface Card 1380
- Cisco UCS Virtual Interface Card 1385
- Cisco UCS Virtual Interface Card 1387
- Cisco UCS Virtual Interface Card 1440
- Cisco UCS Virtual Interface Card 1480
- Cisco UCS Virtual Interface Card 1455
- Cisco UCS Virtual Interface Card 1457

VM-FEX for KVM

Overview of VM-FEX for KVM

The Kernel-based Virtual Machine (KVM) is a virtualization package for Linux on an x86 hardware platform. KVM uses x86 hardware virtualization extensions (for example, Intel VT-x) to implement a hypervisor that hosts VMs as userspace processes.

With VM-FEX for KVM, the hypervisor performs no switching of VM traffic. Working with an installed VIC adapter, the hypervisor acts as an interface virtualizer and performs the following functions:

- For traffic going from a VM to the VIC, the interface virtualizer identifies the source vNIC so that the VIC can explicitly tag each packet that is generated by that vNIC.
- For traffic that is received from the VIC, the interface virtualizer directs the packet to the specified vNIC.

All switching is performed by the external fabric interconnect, which can switch not only between physical ports, but also between virtual interfaces (VIFs) that correspond to the vNICs on the VMs.

For more information about KVM, see the following URL: https://www.linux-kvm.org.

Cisco UCS Manager Components

Clusters

Cisco UCS clusters is a grouping of hypervisors that can be distributed across multiple hosts. In a KVM system, clusters are analogous to the distributed virtual switch (DVS) in a VMware ESX system.

In the current Cisco UCS KVM implementation, clusters define the scope of the port profile and are the boundary of the migration domain. When multiple KVM hosts are associated to a cluster, you can migrate a VM from one host to another within the cluster.

Note In the current Cisco UCS implementation of VM-FEX for KVM, only one cluster, the default cluster, is used. Although you can create additional clusters, you can specify only the default cluster for a VM on the KVM host.

Port Profiles

Port profiles contain the properties and settings that are used to configure virtual interfaces in Cisco UCS. The port profiles are created and administered in Cisco UCS Manager.

Important After a port profile is created, assigned to, and actively used by a cluster, any changes made to the networking properties of the port profile in Cisco UCS Manager are immediately applied to the cluster with no need for a host reboot.

Cisco UCS 6454 Fabric Interconnects do not support configurations related to port profiles and distributed virtual switches (DVS)

Port Profile Client

The port profile client is a cluster to which a port profile is applied.

Note In the current Cisco UCS implementation of VM-FEX for KVM, the default cluster is the only available port profile client.

KVM Components

Hypervisor

The hypervisor supports multiple VMs that run a variety of guest operating systems by providing connectivity between the VMs and the network. The hypervisor for KVM is a host server with Red Hat Enterprise Linux (RHEL) installed. The earliest supported release for VM-FEX is RHEL 6.1, but some features (such as SR-IOV) require a later version.

The host server with hypervisor must have a Cisco VIC adapter installed.

For more information about virtualization using Red Hat Enterprise Linux, see the *Red Hat Enterprise Virtualization for Servers Installation Guide* available at the following URL: https://www.redhat.com.

libvirt

Libvirt is an open source toolkit that allows you to manage various virtualization technologies such as KVM, Xen, and VMware ESX. Libvirt, which runs on the hypervisor as a service named libvirtd, provides a command-line interface (virsh) and provides the toolkit for a graphical user interface package (virt-manager).

Each virtual machine created and managed by libvirt is represented in the form of a domain XML file.

For more information about the libvirt virtualization API, see the following URL: https://www.libvirt.org.

For more information about the virsh CLI, see the following URLs:

- https://linux.die.net/man/1/virsh
- https://www.libvirt.org/virshcmdref.html

MacVTap

MacVTap is a Linux driver that allows the direct attachment of a VM's vNIC to a physical NIC on the host server.

For more information about the MacVTap driver, see the following URL: https://virt.kernelnewbies.org/MacVTap.

VirtIO

The VirtIO paravirtualized network driver (virtio-net) runs in the guest operating system of the VM and provides a virtualization-aware emulated network interface to the VM.

For more information about the VirtIO driver, see the following URL: https://wiki.libvirt.org/page/Virtio.

Driver Topologies

Several driver topologies (modes) are available to implement a VM-FEX connection between a VM vNIC and the host VIC adapter. In each of these topologies, VM traffic is sent only to or from the VIC adapter. Traffic from one VM to another VM on the same host must first exit the host for switching by the external fabric interconnect.

Note In any topology, the configuration of the Quick EMUlator (QEMU) PCI layer might limit the number of PCI devices that the host can assign to a VM.

MacVTap Direct (Private)

The MacVTap Linux driver is installed in the hypervisor (VMM) and connects each VM's VirtIO interface to a physical PCIe port of the VIC adapter. The MacVTap driver mode is private, which means that all VM traffic is sent directly to and from the host adapter with external switching. The number of supported VMs is limited to the number of VIC adapter ports. Live migration is supported.

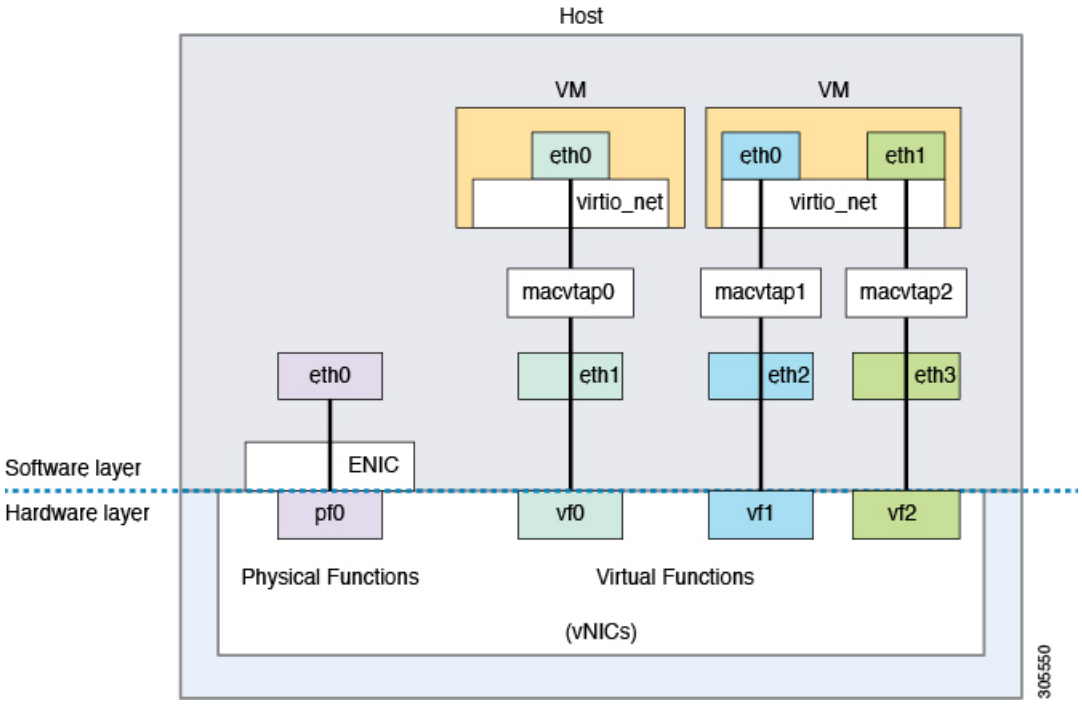
Note Beginning with Cisco UCS Release 2.1, the MacVTap Direct (Private) topology is no longer supported.

SR-IOV with MacVTap Passthrough (Emulation Mode)

The MacVTap Linux driver is installed in the hypervisor and connects each VM's VirtIO interface to a VF on an SR-IOV-capable VIC adapter. The MacVTap driver mode is 'passthrough' and all VM traffic is sent to and from the VF. When we apply a port profile to a VF, libvirt determines the PF associated with the VF, and

it configures the VF going through the PF. This topology is also known as MacVTap passthrough (emulation mode). An example of SR-IOV with MacVTap passthrough is shown in Figure 1. Figure 1 is a simplified version of the hardware and software components.

Figure 1: Figure 1

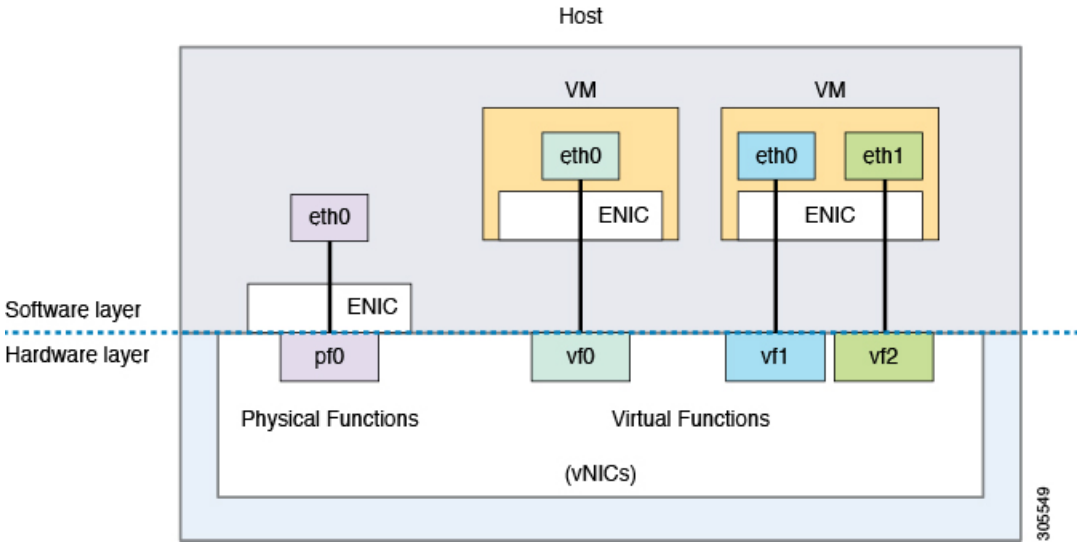


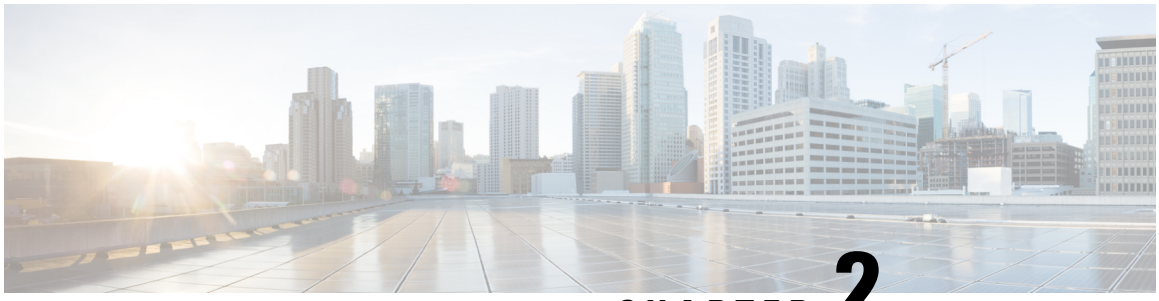
The maximum number of supported VMs is determined by the number of VFs provided by the VIC adapter. The number of VFs that you can assign to a PF might be further limited by the host Netlink protocol implementation (the limit is typically between 22 and 32 VFs per PF, depending on the OS version). Live migration is supported.

SR-IOV VF Passthrough (Hostdev Mode)

The MacVTap and VirtIO drivers are not used. Instead, the Ethernet driver (enic) of the VIC adapter is installed in the VM kernel and connects directly to a VF. You can configure the VF through the associated PF using libvirt. In libvirt documentation, this topology is called hostdev mode. This topology is also known as PCI passthrough. The number of supported VMs is determined by the number of VFs provided by the VIC adapter. Live migration is not supported. An example of SR-IOV with VF passthrough is shown in Figure 2. Figure 2 is a simplified version of the hardware and software components.

Figure 2: Figure 2





CHAPTER 2

Configuring VM-FEX for KVM

Guidelines and Prerequisites for KVM

Consider the following guidelines and prerequisites when configuring Kernel-based Virtual Machine (KVM):

- The host must be managed by Cisco UCS Manager Release 2.1 or later.
- On Red Hat Enterprise Linux (RHEL) hosts, disable generic receive offload (GRO) by using the `ethtool`-K *interface* **gro off** command. This issue occurs because Microsoft Windows VIRTIO does not support GRO, which results in very poor Ethernet performance compared with Linux VMs.
- The host operating system must be RHEL with KVM support.
  - The Single Root I/O Virtualization (SR-IOV) with MacVTap topology requires RHEL 6.2 or later.
  - The SR-IOV passthrough topology requires RHEL 6.3 or later.

  For more information about installing RHEL with KVM, see the *Red Hat Enterprise Virtualization for Servers Installation Guide*.
- The host must have libvirt with virsh or virt-manager installed for creating and managing the VMs.
- One or more Cisco VIC adapters must be installed in the host.

  For more information about installing a Cisco VIC adapter, see the *Cisco UCS 5108 Server Chassis Hardware Installation Guide*.

Consider the following guidelines and prerequisites when configuring an SR-IOV topology:

- In service profiles, under the **Policies** tab, assign default SRIOV in the BIOS Policy.

  For more information about configuring Cisco UCS server BIOS settings, see the *Cisco UCS Manager CLI Configuration Guide*.
- For SR-IOV topologies, configure a dynamic connection policy in a service profile. Apply the service profile on a static vNIC to specify the number of VFs, the fabric preference, and the adapter policy. The

static vNIC becomes a PF when you configure one or more VFs under it. VFs are provisioned as dynamic vNICs.

- When you upgrade Cisco UCS Manager to an SR-IOV capable release, the existing static and dynamic vNICs are not automatically enabled for SR-IOV. To convert to SR-IOV, you must disable any existing dynamic connection policy in the service profile and then specify a reference to a dynamic connection policy under a static vNIC.

Configuring VM-FEX for SR-IOV with MacVTap Topology

Before you begin

Prepare the host server as described in Guidelines and Prerequisites for KVM, on page 9.

Procedure

Step 1 In Cisco UCS Manager, configure a service profile for VM-FEX for KVM.

Create or modify a dynamic vNIC connection policy.

For more information, see Configuring a Service Profile with VM-FEX, on page 17.

Step 2 In Cisco UCS Manager, define a port profile and associate it with a port profile client.

Create a port profile to define the properties and settings used to configure the virtual interfaces. For KVM, you must select the default cluster as the port profile client.

For more information, see Configuring Port Profiles, on page 23.

Step 3 In Cisco UCS Manager, in the **Connection Policies** area of the vNIC in a service profile, choose the **Dynamic vNIC** radio button, then assign the created dynamic vNIC connection policy.

Step 4 On each KVM server, use virsh or virt-manager to create one or more virtual machines (VMs).

For more information about installing VMs using these libvirt-based utilities, see the documents listed in KVM Components, on page 4.

Note When creating a VM using virsh, or when editing the VM domain XML descriptor file, use care when entering data such as a universally unique identifier (UUID), as you will receive no indication of incorrect data values or formats.

Step 5 For each VM, edit the domain XML descriptor file (and any network XML files, if present) to configure a vNIC interface that is directly attached to the VIC and uses the port profile defined in Cisco UCS Manager.

For more information about configuring a VM interface, see Configuring the VM Interface, on page 11.

Step 6 Use the inbox enic driver for the VirtIO paravirtualized network driver (virtio-net) for the guest operating system.

Recent versions of most common operating systems provide default virtio-net drivers. For more information, contact Red Hat or the provider of the guest operating system.

Configuring VM-FEX for SR-IOV Passthrough Topology

Before you begin

Prepare the host server as described in Guidelines and Prerequisites for KVM, on page 9.

Procedure

Step 1 In Cisco UCS Manager, configure a service profile for VM-FEX for KVM.

Create or modify a dynamic vNIC connection policy.

For more information, see Configuring a Service Profile with VM-FEX, on page 17.

Step 2 In Cisco UCS Manager, define a port profile and associate it with a port profile client.

Create a port profile to define the properties and settings used to configure the virtual interfaces. For KVM, you must select the default cluster as the port profile client.

For more information, see Configuring Port Profiles, on page 23.

Step 3 In Cisco UCS Manager, in the **Connection Policies** area of the vNIC in a service profile, choose the **Dynamic vNIC** radio button, then assign the created dynamic vNIC connection policy.

Step 4 On the KVM host, activate the Intel VT-d extensions.

For more information about activating the VT-d extensions, see Activating Intel VT-d in the Kernel, on page 15.

Step 5 On each KVM server, use virsh or virt-manager to create one or more virtual machines (VMs).

For more information about installing VMs using these libvirt-based utilities, see the documents listed in KVM Components, on page 4.

Note When creating a VM using virsh, or when editing the VM domain XML descriptor file, use care when entering data such as a universally unique identifier (UUID), as you will receive no indication of incorrect data values or formats.

Step 6 For each VM, edit the domain XML descriptor file (and any network XML files, if present) to configure a vNIC interface that is directly attached to the VIC and uses the port profile defined in Cisco UCS Manager.

For more information about configuring a VM interface, see Configuring the VM Interface, on page 11.

Step 7 On each VM, install an enic driver that supports an SR-IOV VF.

With RHEL 6.3 or later, use the inbox enic driver.

Configuring the VM Interface

After creating a VM using a libvirt-based utility, you can add the network configuration, including the port profile information, either to the domain XML of the VM or to a separate network XML file that you can

reference from the domain XML. One of the advantages of adding the configuration information to a separate network XML file is that you can specify a pool of devices. For more information about the network XML file components and attributes, see https://libvirt.org/formatnetwork.html.

For more information about the domain XML file components and attributes, see the libvirt documentation at https://libvirt.org/formatdomain.html#elementsNICS.

Procedure

Step 1 Shut down the VM to be configured.

Step 2 Using the virsh editor, open the domain XML file of the VM for editing.

Example:

This example opens a domain XML file for editing in the virsh editor:

```
[root@chassis1blade5 qemu]# virsh edit vm1-rhel6.2
```

Step 3 In the devices section of the domain XML file, add an interface element that describes a vNIC for the VM.

The components and attributes of the interface element are described in the Example section.

Step 4 Restart the VM.

Example

Example for SR-IOV with MacVTap Mode

This example shows an interface element added to the domain XML file of a VM for connection in SR-IOV with MacVTap (MacVTap Passthrough) topology:

```
<domain type='kvm'>
  <name>vm1-rhel6.2</name>
  ...
  <devices>
    ...
    <interface type='direct'>
      <mac address='01:23:45:67:89:ab' />
      <source dev='eth4' mode='passthrough' />
      <virtualport type='802.1Qbh'>
        <parameters profileid='my-port-profile-3' />
      </virtualport>
      <model type='virtio' />
      <driver name='vhost' />
    </interface>
    ...
  </devices>
  ...
</domain>
```

This list describes the components and attributes of the interface element:

- `interface type='direct'`

  The `direct` type attribute value selects a direct logical attachment of the vNIC to the physical interface of the hypervisor, using the MacVTap driver.

- `mac address='01:23:45:67:89:ab'`

  Explicit specification of the MAC address is optional. Enter a MAC address obtained from your network administrator. If this line is omitted, libvirt generates a MAC address for the vNIC.

  Note We recommend that you do not assign a MAC address used by another VM, even if that VM is currently shut down or is no longer used. If you must reuse a MAC address from a previous VM, make sure that the retention timer has expired and ensure that the previous VM is no longer present in the Cisco UCS Manager view.

- `source dev='eth4' mode='passthrough'`

  The `passthrough` mode attribute value specifies that each VM is connected to the network by a macvtap direct connection with a virtual function (VF). The source interface must be a VF, and not a physical function (PF).

- `virtualport type='802.1Qbh'`

  The `802.1Qbh` type attribute value specifies that the vNIC is connected to an 802.1Qbh extended port for external switching.

- `parameters profileid='my-port-profile-3'`

  This line specifies the name of the port profile to be associated with the interface. The port profile name is case sensitive.The specified port profile must be already defined in Cisco UCS Manager and use the naming syntax described in Creating a Port Profile, on page 23.

- `model type='virtio'`

  This line specifies that the guest interface uses the VirtIO paravirtualized front-end device driver.

- `driver name='vhost'`

  This line specifies that, for higher performance, the host side interface uses the vhost kernel back-end device driver and not the qemu userspace back-end driver.

Example for SR-IOV Passthrough Mode

This example shows an interface element that is added to the domain XML file of a VM for a connection in SR-IOV Passthrough topology:

```
<domain type='kvm'>
  <name>vm1-rhel6.3</name>
  ...
  <devices>
    ...
    <interface type='hostdev' managed='yes'>
      <source>
        <address type='pci' domain='0' bus='0x09' slot='0x0' function='0x01'/>
      </source>
      <mac address='01:23:45:67:89:ab' />
      <virtualport type='802.1Qbh'>
        <parameters profileid='my-port-profile-3' />
      </virtualport>
    </interface>
    ...
  </devices>
```

```
  ...
</domain>
```

This list describes the components and attributes of the interface element that differ from those described in the SR-IOV with MacVTap mode example:

- `interface type='hostdev'`

  The `hostdev` type attribute allows you to assign a host device directly to a guest.

- `address type='pci' domain='0' bus='0x09' slot='0x0' function='0x01'.`

  The `address type` attribute value specifies the PCI address of the host VF. To obtain the address information, you can run the `lspci` command at the Linux prompt. When you run the command, an address string is displayed, for example, `09:00.1 Ethernet controller: Cisco Systems Inc Device 0071 (rev a2)`. In the address string `09.00.1`, 09 indicates the bus, 00 indicates the slot, and 1 indicates the function.

- `mac address='01:23:45:67:89:ab'`

  Explicit specification of the MAC address is optional. Enter a MAC address that you obtained from your network administrator. If this line is omitted, libvirt generates a MAC address for the vNIC.

Note We recommend that you do not assign a MAC address used by another VM, even if that VM is currently shut down or is no longer used. If you must reuse a MAC address from a previous VM, make sure that the retention timer has expired and ensure that the previous VM is no longer present in the Cisco UCS Manager view.

Example of Using a Network XML File to Specify a Pool of Devices

This example shows how to use a network XML file to specify a pool of devices. In RHEL 6.2 or later, create the network file in /etc/libvirt/qemu/networks. List the devices and define a portgroup:

```
<network>
  <name>macvtap_passthru_network</name>
    <forward mode='passthrough'>
      <interface dev='eth2' />
      <interface dev='eth3' />
    </forward>
    <portgroup name='engineering'>
      <virtualport type='802.1Qbh'>
        <parameters profileid='my-port-profile-3' />
      </virtualport>
    </portgroup>
</network>
```

Edit the domain XML file of the VM to reference the network file and portgroup:

```
<domain type='kvm'>
  <name>vm1-rhel6.2</name>
  ...
  <devices>
    ...
```

```
    <interface type='network'>
      <mac address='01:23:45:67:89:ab' />
      <source network='macvtap_passthru_network' portgroup='engineering' />
      <model type='virtio' />
    </interface>
    ...
  </devices>
  ...
</domain>
```

After you create the new network XML file, use the `virsh net-define <new-xml-filename>` command to create the new network from the new network XML file.

Tip You can find the network-related virsh commands with `virsh help | grep net-`

You can view help on any virsh command with `virsh help <command-name>`

This list describes the components and attributes of the interface element that differ from those described in the SR-IOV with MacVTap mode example:

- `interface type='network'`

  The `network` type attribute value specifies an attachment of the VM vNIC to a PCI network device from the pool listed in a separate network file.

- `source network='macvtap_passthru_network' portgroup='engineering'`

  The `network` and `portgroup` attribute values specify the name of a network XML file and its pool of network devices.

Activating Intel VT-d in the Kernel

Perform this procedure on the KVM host to enable Intel VT-d extensions, which are required for SR-IOV passthrough.

For more information about this feature in Red Hat Enterprise Linux (RHEL) systems, see the *Virtualization Deployment and Administration Guide*.

Procedure

Step 1 On the KVM host, open the file `grub.conf` for editing.

The file is typically located in the /boot directory. In RHEL systems, you can also access it using the grub.conf link in the /etc directory.

Step 2 Locate the line beginning with **kernel**.

Step 3 Append the command **intel_iommu=on** to the kernel line..

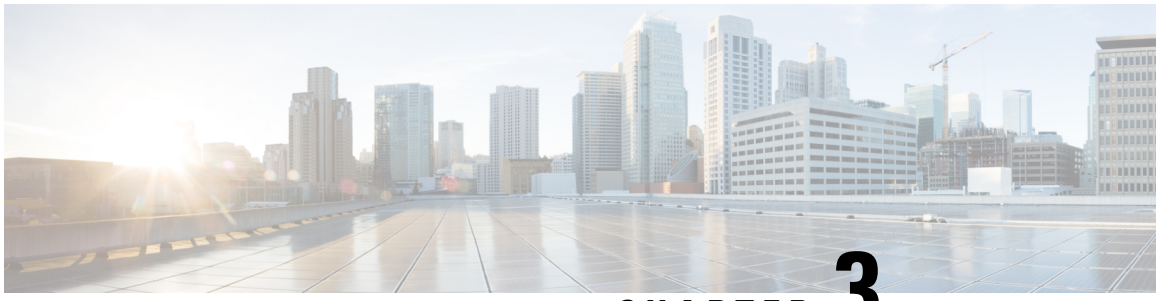
Example:

```
kernel /vmlinuz-2.6.18-190.e15 ro root=/dev/VolGroup00/LogVol00 \
               rhgb quiet intel_iommu=on
```

Step 4 Save the file.

What to do next

Reboot the host.



CHAPTER 3

Configuring a Service Profile with VM-FEX

Configuring Dynamic vNIC Connection Policies

Dynamic vNIC Connection Policy

Note In an SR-IOV topology, such as a Hyper-V or KVM cluster, a Virtual Function (VF) takes the place of the dynamic vNIC. The VF is essentially a restricted version of the dynamic vNIC, in which all system communication and configuration of the VF is performed through the associated physical function (PF).

The dynamic vNIC connection policy determines how the connectivity between VMs and dynamic vNICs is configured. This policy is required for Cisco UCS domains that include servers with VIC adapters on which you have installed VMs and configured dynamic vNICs.

Note Cisco UCS 6454 Fabric Interconnects do not support dynamic vNICs.

Ethernet Adapter Policy

Each dynamic vNIC connection policy includes an Ethernet adapter policy and designates the number of vNICs that can be configured for any server associated with a service profile that includes the policy.

For KVM, use the predefined Ethernet adapter policy named Linux.

Server Migration

Note If you migrate a server that is configured with dynamic vNICs, the dynamic interface used by the vNICs fails and notifies you of that failure.

When the server comes back up, assigns new dynamic vNICs to the server. If you are monitoring traffic on the dynamic vNIC, you must reconfigure the monitoring source.

Creating a Dynamic vNIC Connection Policy

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope org** *org-name*	Enters organization mode for the specified organization. To enter the root organization mode, enter / as the *org-name* .
Step 2	UCS-A /org # **create dynamic-vnic-conn-policy** *policy-name*	Creates the specified vNIC connection policy and enters organization vNIC connection policy mode. This name can be between 1 and 32 alphanumeric characters. You cannot use spaces or any special characters other than - (hyphen), _ (underscore), : (colon), and . (period), and you cannot change this name after the object is saved. **Note** Do not specify "default" as the value for the dynamic vNIC connection policy name. Cisco UCS Manager automatically resolves any empty policy references to "default". Any service profiles or service profile templates with only static vNICS defined will automatically reference the policy "default" when it is present. If you specify “default” for the dynamic vNIC connection policy name, then unexpected dynamic vNICs might be created on those service profiles or service profile templates.
Step 3	(Optional) UCS-A /org/dynamic-vnic-conn-policy # **set descr** *description*	Provides a description for the policy. Enter up to 256 characters. You can use any characters or spaces except ` (accent mark), \ (backslash), ^ (carat), " (double quote), = (equal

	Command or Action	Purpose
		sign), > (greater than), < (less than), or ' (single quote). If your description includes spaces or nonalphanumeric characters, you must begin and end your description with double quotation marks. The quotation marks do not appear in the description field of any **show** command output.
Step 4	UCS-A /org/dynamic-vnic-conn-policy # **set adapter-policy** *policy-name*	Specifies the Ethernet adapter policy to use for this policy. The adapter policy must already exist.
Step 5	UCS-A /org/dynamic-vnic-conn-policy # **set dynamic-eth** {*dynamic-eth-num* | **off**}	Specifies the number of dynamic vNICs to use for this policy. Enter an integer between 0 and 256. The default is 54. **Note** Components of your system may limit this number to fewer than 256 vNICs.
Step 6	UCS-A /org/dynamic-vnic-conn-policy # **set protection** {**protected** | **protected-pref-a** | **protected-pref-b**}	Dynamic vNICs are always protected in Cisco UCS, but this command allows you to select a preferred fabric, if any. You can choose one of the following options: • **protected**—Cisco UCS uses whichever fabric is available. • **protected-pref-a**—Cisco UCS attempts to use fabric A, but fails over to fabric B if necessary. • **protected-pref-b**—Cisco UCS attempts to use fabric B, but fails over to fabric A if necessary.
Step 7	UCS-A /org/dynamic-vnic-conn-policy # **commit-buffer**	Commits the transaction.

Example

The following example shows how to create a dynamic vNIC connection policy named MyDynVnicConnPolicy that uses the system-provided Linux Ethernet adapter policy for 12 dynamic vNICs and commit the transaction:

```
UCS-A# scope org /
UCS-A /org # create dynamic-vnic-conn-policy MyDynVnicConnPolicy
UCS-A /org/dynamic-vnic-conn-policy* # set adapter-policy Linux
UCS-A /org/dynamic-vnic-conn-policy* # set descr "Dynamic vNIC for Eth policy"
```

```
UCS-A /org/dynamic-vnic-conn-policy* # set dynamic-eth 12
UCS-A /org/dynamic-vnic-conn-policy* # commit-buffer
UCS-A /org/dynamic-vnic-conn-policy #
```

Deleting a Dynamic vNIC Connection Policy

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope org** *org-name*	Enters organization mode for the specified organization. To enter the root organization mode, enter / as the *org-name* .
Step 2	UCS-A /org # **delete dynamic-vnic-conn-policy** *policy-name*	Deletes the specified vNIC connection policy.
Step 3	UCS-A /org # **commit-buffer**	Commits the transaction.

Example

The following example shows how to delete the dynamic vNIC connection policy named MyDynVnicConnPolicy and commit the transaction:

```
UCS-A# scope org /
UCS-A /org # delete dynamic-vnic-conn-policy MyDynVnicConnPolicy
UCS-A /org* # commit-buffer
UCS-A /org #
```

Viewing Dynamic vNIC Properties in a VM

Before you begin

The VM must be operational.

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.
Step 2	UCS-A /system # **scope vm-mgmt**	Enters VM management mode.
Step 3	(Optional) UCS-A /system/vm-mgmt # **show virtual-machine**	Displays the running virtual machines.
Step 4	UCS-A /system/vm-mgmt # **scope virtual-machine** *uuid*	Enters command mode for the virtual machine that contains the dynamic vNIC.

	Command or Action	Purpose
Step 5	UCS-A /system/vm-mgmt/virtual-machine # **show vnic [detail]**	Displays the vNIC properties.

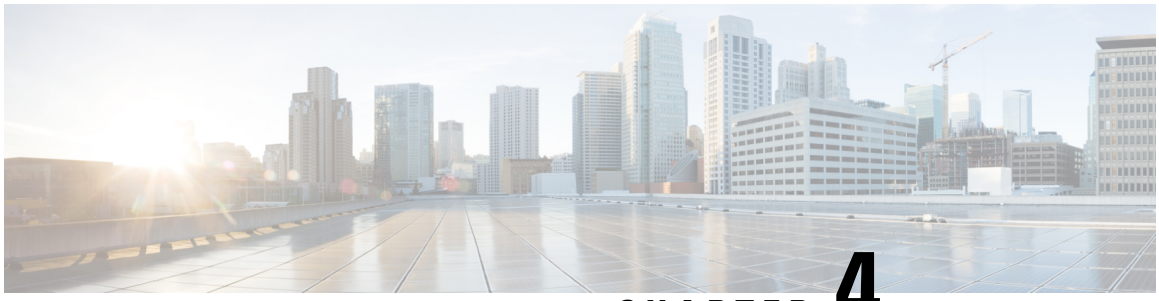
Example

The following example shows how to display the properties of a dynamic vNIC in a VM:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt # show virtual-machine
Virtual Machine:
    UUID: 420a00c8-934b-4ae3-6af5-2ce9b8bd0f44
    Service Profile: org-root/ls-PTS-ch6-7
    Server: sys/chassis-6/blade-7
    Status: Online
.
.
.
UCS-A /system/vm-mgmt # scope virtual-machine 420a08b0-cda7-9e0a-424f-01ec8653eea0
UCS-A /system/vm-mgmt/virtual-machine # show vnic detail

vNIC:
    Name: 4479
    Status: Online
    MAC Address: 00:50:56:8A:07:B5
    Profile Name: VM-mgmt
    Virtual Adapter: sys/chassis-1/blade-1/adapter-1/host-eth-9
    Vnic Dn: org-root/ls-PTS-ch1-1/ether-dynamic-prot-009
    Current Task:

UCS-A /system/vm-mgmt/virtual-machine #
```



CHAPTER 4

Configuring Port Profiles

Port Profiles

Port profiles contain the properties and settings that you can use to configure virtual interfaces in Cisco UCS for VM-FEX. The port profiles are created and administered in Cisco UCS Manager. After a port profile is created, assigned to, and actively used by one or more clusters, any changes made to the networking properties of the port profile in Cisco UCS Manager are immediately applied to those clusters.

Cisco UCS 6454 Fabric Interconnects do not support configurations related to port profiles and DVSes.

Creating a Port Profile

Note In a VM-FEX for KVM system, the following conditions apply:

- The **set max-ports** command applies to the cluster; there is no distributed virtual switch (DVS).
- The **set host-nwio-perf** command has no effect.

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.

	Command or Action	Purpose
Step 2	UCS-A /system # **scope vm-mgmt**	Enters system VM management mode.
Step 3	UCS-A /system/vm-mgmt # **scope profile-set**	Enters system VM management profile set mode.
Step 4	UCS-A /system/vm-mgmt/profile-set # **create port-profile** *profile-name*	Creates the specified port profile and enters system VM management profile set port profile mode.
Step 5	(Optional) UCS-A /system/vm-mgmt/profile-set/port-profile # **set descr** *description*	Provides a description for the port profile. **Note** If your description includes spaces, special characters, or punctuation, you must begin and end your description with quotation marks. The quotation marks do not appear in the description field of any **show** command output.
Step 6	(Optional) UCS-A /system/vm-mgmt/profile-set/port-profile # **set host-nwio-perf** {**high-performance** | **none**}	You can choose one of the following options: • **high-performance** • **none**
Step 7	UCS-A /system/vm-mgmt/profile-set/port-profile # **set max-ports** *max-num*	Specifies the maximum number of ports that the port profile can use. The maximum number of ports that can be associated to a port profile is 4096. The default is 64 ports. If the DVS has more than one associated port profile, each port profile client can be set to use a maximum of 4096 ports.
Step 8	UCS-A /system/vm-mgmt/profile-set/port-profile # **set nw-control-policy** *policy-name*	Specifies the network control policy to use for the port profile.
Step 9	UCS-A /system/vm-mgmt/profile-set/port-profile # **set pin-group** *group-name*	Specifies the LAN pin group to use for the port profile.
Step 10	UCS-A /system/vm-mgmt/profile-set/port-profile # **set qos-policy** *policy-name*	Specifies the QoS policy to use for the port profile.
Step 11	UCS-A /system/vm-mgmt/profile-set/port-profile # **commit-buffer**	Commits the transaction.

Example

The following example shows how to create and configure a port profile named MyProfile and commit the transaction:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt # scope profile-set
UCS-A /system/vm-mgmt/profile-set # create port-profile MyProfile
UCS-A /system/vm-mgmt/profile-set/port-profile* # set descr "This is my port profile"
UCS-A /system/vm-mgmt/profile-set/port-profile* # set max-ports 24
UCS-A /system/vm-mgmt/profile-set/port-profile* # set nw-control-policy ncp5
UCS-A /system/vm-mgmt/profile-set/port-profile* # set pin-group PinGroup54
UCS-A /system/vm-mgmt/profile-set/port-profile* # set qos-policy QosPolicy34
UCS-A /system/vm-mgmt/profile-set/port-profile* # commit-buffer
UCS-A /system/vm-mgmt/profile-set/port-profile #
```

What to do next

Add a port profile client to the port profile.

Deleting a Port Profile

You cannot delete a port profile if a VM is actively using that port profile.

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.
Step 2	UCS-A /system # **scope vm-mgmt**	Enters system VM management mode.
Step 3	UCS-A /system/vm-mgmt # **scope profile-set**	Enters system VM management profile set mode.
Step 4	UCS-A /system/vm-mgmt/profile-set # **delete port-profile** *profile-name*	Deletes the specified port profile and its associations.
Step 5	UCS-A /system/vm-mgmt/profile-set # **commit-buffer**	Commits the transaction. Cisco UCS Manager deletes the port profile and all associated port profile clients.

Example

The following example shows how to delete the port profile named MyProfile and commit the transaction:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt # scope profile-set
UCS-A /system/vm-mgmt/profile-set # delete port-profile MyProfile
```

```
UCS-A /system/vm-mgmt/profile-set* # commit-buffer
UCS-A /system/vm-mgmt/profile-set #
```

Adding a VLAN to a Port Profile

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.
Step 2	UCS-A /system # **scope vm-mgmt**	Enters system VM management mode.
Step 3	UCS-A /system/vm-mgmt # **scope profile-set**	Enters system VM management profile set mode.
Step 4	UCS-A /system/vm-mgmt/profile-set # **scope port-profile** *profile-name*	Enters system VM management profile set port profile mode for the specified port profile.
Step 5	UCS-A /system/vm-mgmt/profile-set/port-profile # **create vlan** *vlan-name*	Specifies a VLAN to use for the port profile. **Note** You can create multiple VLANs for guest VLAN trunking.
Step 6	(Optional) UCS-A /system/vm-mgmt/profile-set/port-profile/vlan # **set default-net** {**no** | **yes**}	Sets the default-net VLAN as the native VLAN in UCS Manager.
Step 7	UCS-A /system/vm-mgmt/profile-set/port-profile/vlan # **commit-buffer**	Commits the transaction.

Example

The following example shows how to add the VLAN named accounting to the port profile named MyProfile, set the VLAN as non-native, and commit the transaction:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt# scope profile-set
UCS-A /system/vm-mgmt/profile-set # scope port-profile MyProfile
UCS-A /system/vm-mgmt/profile-set/port-profile # create vlan accounting
UCS-A /system/vm-mgmt/profile-set/port-profile/vlan* # set default-net no
UCS-A /system/vm-mgmt/profile-set/port-profile/vlan* # commit-buffer
UCS-A /system/vm-mgmt/profile-set/port-profile/vlan #
```

Removing a VLAN from a Port Profile

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.
Step 2	UCS-A /system # **scope vm-mgmt**	Enters system VM management mode.
Step 3	UCS-A /system/vm-mgmt # **scope profile-set**	Enters system VM management profile set mode.
Step 4	UCS-A /system/vm-mgmt/profile-set # **scope port-profile** *profile-name*	Enters system VM management profile set port profile mode for the specified port profile.
Step 5	UCS-A /system/vm-mgmt/profile-set/port-profile # **delete vlan** *vlan-name*	Removes the specified VLAN from the port profile.
Step 6	UCS-A /system/vm-mgmt/profile-set/port-profile # **commit-buffer**	Commits the transaction.

Example

The following example shows how to remove the VLAN named accounting from the port profile named MyProfile and commit the transaction:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt# scope profile-set
UCS-A /system/vm-mgmt/profile-set # scope port-profile MyProfile
UCS-A /system/vm-mgmt/profile-set/port-profile # delete vlan accounting
UCS-A /system/vm-mgmt/profile-set/port-profile* # commit-buffer
UCS-A /system/vm-mgmt/profile-set/port-profile #
```

Port Profile Clients

The port profile client determines the cluster or clusters to which a port profile is applied.

Adding a Port Profile Client to a Port Profile

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.

	Command or Action	Purpose
Step 2	UCS-A /system # **scope vm-mgmt**	Enters system VM management mode.
Step 3	UCS-A /system/vm-mgmt # **scope profile-set**	Enters system VM management profile set mode.
Step 4	UCS-A /system/vm-mgmt/profile-set # **scope port-profile** *profile-name*	Enters system VM management profile set port profile mode for the specified port profile.
Step 5	UCS-A /system/vm-mgmt/profile-set/port-profile # **create client** *client-name*	Creates the specified port profile client and enters system VM management profile set port profile client mode. The port profile client determines the clusters to which the port profile is applied. By default, a port profile applies to all clusters; however, you can use the optional **set data-center**, **set folder**, and **set cluster** commands to apply the port profile to all clusters in a specific datacenter or datacenter folder, or to a specific cluster.
Step 6	(Optional) UCS-A /system/vm-mgmt/profile-set/port-profile/client # **set descr** *description*	Provides a description for the port profile client. **Note** If your description includes spaces, special characters, or punctuation, you must begin and end your description with quotation marks. The quotation marks do not appear in the description field of any **show** command output.
Step 7	UCS-A /system/vm-mgmt/profile-set/port-profile/client # **commit-buffer**	Commits the transaction.

Example

The following example shows how to create a port profile client named MyClient that applies the port profile and commit the transaction:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt # scope profile-set
UCS-A /system/vm-mgmt/profile-set # scope port-profile MyProfile
UCS-A /system/vm-mgmt/profile-set/port-profile* # create client MyClient
UCS-A /system/vm-mgmt/profile-set/port-profile/client* # set descr "This is the client for
 my port profile"
UCS-A /system/vm-mgmt/profile-set/port-profile/client* # commit-buffer
UCS-A /system/vm-mgmt/profile-set/port-profile/client #
```

Deleting a Port Profile Client from a Port Profile

You cannot delete a port profile client if a VM is actively using the port profile with which the client is associated.

Procedure

	Command or Action	Purpose
Step 1	UCS-A# **scope system**	Enters system mode.
Step 2	UCS-A /system # **scope vm-mgmt**	Enters system VM management mode.
Step 3	UCS-A /system/vm-mgmt # **scope profile-set**	Enters system VM management profile set mode.
Step 4	UCS-A /system/vm-mgmt/profile-set # **scope port-profile** *profile-name*	Enters system VM management profile set port profile mode for the specified port profile.
Step 5	UCS-A /system/vm-mgmt/profile-set/port-profile # **delete client** *client-name*	Deletes the specified port profile client.
Step 6	UCS-A /system/vm-mgmt/profile-set/port-profile # **commit-buffer**	Commits the transaction.

Example

The following example shows how to delete the port profile client named OtherClient from the port profile named MyProfile and commit the transaction:

```
UCS-A# scope system
UCS-A /system # scope vm-mgmt
UCS-A /system/vm-mgmt# scope profile-set
UCS-A /system/vm-mgmt/profile-set # scope port-profile MyProfile
UCS-A /system/vm-mgmt/profile-set/port-profile # delete client OtherClient
UCS-A /system/vm-mgmt/profile-set/port-profile* # commit-buffer
UCS-A /system/vm-mgmt/profile-set/port-profile #
```