



SR-IOV Configuration Guide

Intel® Ethernet 800 Series on Red Hat Enterprise Linux 8

Technical Brief

Ethernet Products Group (EPG)

June 2021

Revision 1.2
630211-003

Revision History

Revision	Date	Comments
1.2	June 10, 2021	Updates include the following: <ul style="list-style-type: none"> Updated Section 2.2, "VF Creation".
1.1	September 21, 2020	Updates include the following: <ul style="list-style-type: none"> Updated Section 1.0, "Introduction". Updated Section 1.1, "Intel® Ethernet 800 Series". Updated Section 1.2, "I/O Virtualization". Updated Section 1.2.1, "Hardware Requirements". Updated Section 1.2.2, "Software Requirements". Updated Section 2.1, "Server Setup". Updated Section 2.2, "VF Creation".
1.0	July 23, 2020	Initial public release

Contents

1.0	Introduction	5
1.1	Intel® Ethernet 800 Series	5
1.2	I/O Virtualization	6
1.2.1	Hardware Requirements	6
1.2.2	Software Requirements	6
2.0	Installation and Configuration	7
2.1	Server Setup	7
2.2	VF Creation	9
2.3	VM Setup	13
3.0	Customer Support	17
4.0	Product Information	17



NOTE: *This page intentionally left blank.*

1.0 Introduction

This document shows how to make use of Intel® Ethernet 800 Series Virtual Functions using Linux KVM, which is an integral part of Red Hat Enterprise Linux.

This document is provided as guidance, and the steps in the document are likely to work for future OS releases. There are no current plans to update this document for new releases of RHEL.

1.1 Intel® Ethernet 800 Series

The Intel® Ethernet 800 Series (800 Series) supports up to 100 Gb/s throughput for the mainstream market for a variety of workloads used in Enterprise, Communications, and Cloud markets.

Following is an abbreviated list of features offered by the 800 Series. This list can change as software-enabled features are added from release to release. For an up-to-date and comprehensive list of features, refer the *Intel® Ethernet Controller E810 Feature Support Matrix* available at <https://cdrdv2.intel.com/v1/dl/getContent/630155>.

- Support for both 50 Gb/s PAM4 and 25 Gb/s NRZ (E810-CAM2/E810-CAM1 only).
- Dynamic Device Personalization (DDP) with fully programmable pipeline that can add or modify protocols with selectable packages per device, allowing for fast-paced innovations.
- Application Device Queues (ADQ) feature to increase application predictability, reduce application latency, and improve application throughput.
- Support for both iWARP and RoCEv2 RDMA, selectable via software per port for low latency, high throughput workload.
- Support for 8x10 GbE connections for appliance designs (E810-CAM2 only).
- Optimized Tx Scheduler for steady Tx traffic flow to avoid burst send and in-network congestion.
- IEEE 1588 support for precision time measurement.
- Enhanced Data Plane Development Kit (DPDK) for Network Functions Virtualization acceleration, advanced packet forwarding, and highly efficient packet processing.
- More resources to support higher density server virtualization deployments: 256 VFs, 768 VSIs.
- Management key features:
 - Firmware Management Protocol (FMP)
 - MCTP over PCIe and SMBus/I²C
 - PLDM over MCTP; PLDM Monitoring; PLDM Firmware Update; PLDM for RDE
 - NET2BMC/OS2BMC
- Security:
 - Hardware based Root of Trust
 - Authentication on Read and Power On; Whitelist approach on NVM update
 - Built-in detection of firmware/critical setting corruption with automated device recovery

Following are the controller and the single-port and dual-port adapter offerings based on the 800 Series:

- Intel® Ethernet Controller E810-CAM2
- Intel® Ethernet Controller E810-CAM1

Did this document help answer your questions?

- Intel® Ethernet Controller E810-XXVAM2
- Intel® Ethernet Network Adapter E810-CQDA2
- Intel® Ethernet Network Adapter E810-CQDA1
- Intel® Ethernet Network Adapter E810-XXVDA4
- Intel® Ethernet Network Adapter E810-XXVDA2
- Intel® Ethernet Network Adapter E810-CQDA2 for OCP 3.0
- Intel® Ethernet Network Adapter E810-CQDA1 for OCP 3.0

Note: These do **NOT** ship with optics installed. Optics must be purchased separately.

1.2 I/O Virtualization

The 800 Series delivers numerous industry-leading features that are helping data center administrators implement innovative solutions for difficult and challenging connectivity problems. I/O Virtualization is one of the fastest growing usage models within the data center.

The 800 Series provides the ability to create Virtual Functions (VFs) that are identical instantiations of the Physical Functions (PFs). VFs provide up to 100 GbE connectivity to Virtual Machines (VMs) within a virtualized operating system framework. The 800 Series supports up to 256 VFs. These VFs are split evenly between the ports on the adapter, meaning a 4-port card supports 64 VFs per port, while a 2-port card support 128 VFs per port.

This document shows how to make use of these VFs using Linux KVM, which is an integral part of Red Hat Enterprise Linux 7 and 8.

1.2.1 Hardware Requirements

- An Intel® Ethernet 800 Series Network Adapter (or other adapter based on an Intel® Ethernet 800 Series Controller).
- A server platform that supports Intel® Virtualization Technology for Directed I/O (Intel® VT-d) and the PCI-SIG Single Root I/O Virtualization and Sharing (SR-IOV) specification.
- A server platform with an available PCI Express Gen 4.0/3.0 x16 or x8 slot, depending on the specific board.

1.2.2 Software Requirements

- Red Hat Enterprise Linux Version 8.0 (RHEL 8).
- Intel® Ethernet 800 Series Linux Drivers for PF and VF, available at:

<http://sourceforge.net/projects/e1000/files/>

<https://downloadcenter.intel.com>

Note: Refer to the Release Notes and/or Feature Support Matrix documents for the PF and VF drivers that are included in the driver source packages to verify which features are supported. Use the PF and VF drivers that are included in RHEL 8 distribution. If the feature is not supported by the RHEL 8 inbox driver, or if the feature is not working as expected, consider downloading the out-of-tree PF and VF drivers from either of the previously listed sources.

2.0 Installation and Configuration

2.1 Server Setup

This section shows various setup and configuration steps for enabling SR-IOV on server adapters based on the 800 Series.

1. Install an 800 Series Network Adapter into an available PCI Express x8 or x16 slot, depending on the board chosen.

Note: Ensure that the x16 or x8 slot is electrically connected as x16 or x8, respectively, as some slot's electrical support differs from what the physically appear to support. Verify this with your server manufacturer or system documentation.

2. Power up the server.
3. Enter the server's BIOS setup and ensure that the Virtualization Technology (VT) and Intel® VT-d features are enabled. On certain servers, SR-IOV might need to be enabled in the BIOS.

4. Install RHEL 8 on the server. Ensure that all Linux KVM modules, libraries, user tools, and utilities are installed during the operation system installation.

Note: The Red Hat Enterprise Linux installation process might require a server reboot upon successful operating system install.

5. Log in to the newly-installed Red Hat Enterprise Linux operating system using the *root* user account and password.
6. I/O Memory Management Unit (IOMMU) support is required for a VF to function properly when assigned to a VM. The following kernel boot parameter is required to enable IOMMU support for Linux kernels:

```
intel_iommu=on
```

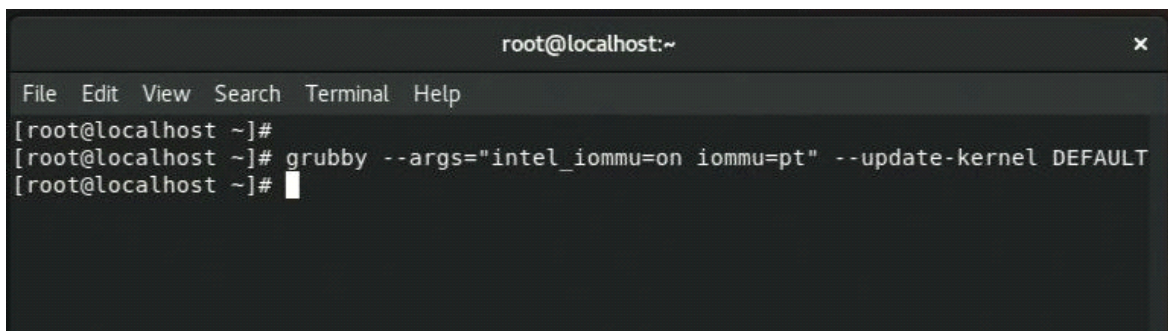
Additionally, adding the optional parameter might improve IO performance in the host:

```
iommu=pt
```

These parameters can be applied using the **grubby** Tool. To update the kernel arguments with the options listed above, run the following command (as shown in [Figure 1](#)):

```
#grubby --args="intel_iommu=on iommu=pt" --update-kernel DEFAULT
```

This command can be run with a singular parameter as well if not including the `iommu=pt` option.

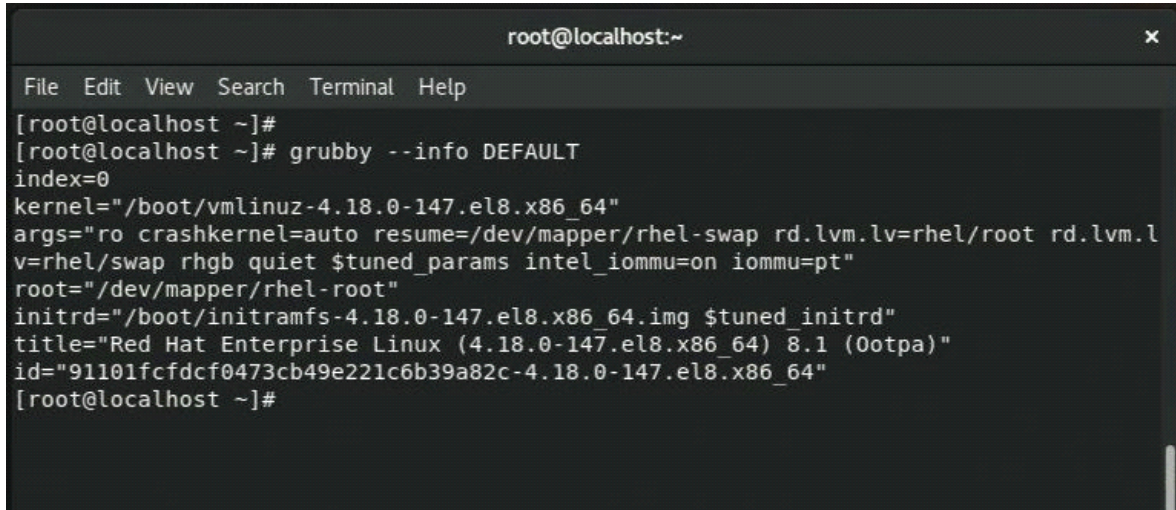


```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]#  
[root@localhost ~]# grubby --args="intel_iommu=on iommu=pt" --update-kernel DEFAULT  
[root@localhost ~]#
```

Figure 1. Kernel Argument Update with grubby Tool

7. To display the updated kernel options, run the command (as shown in [Figure 2](#)):

```
#grubby --info DEFAULT
```



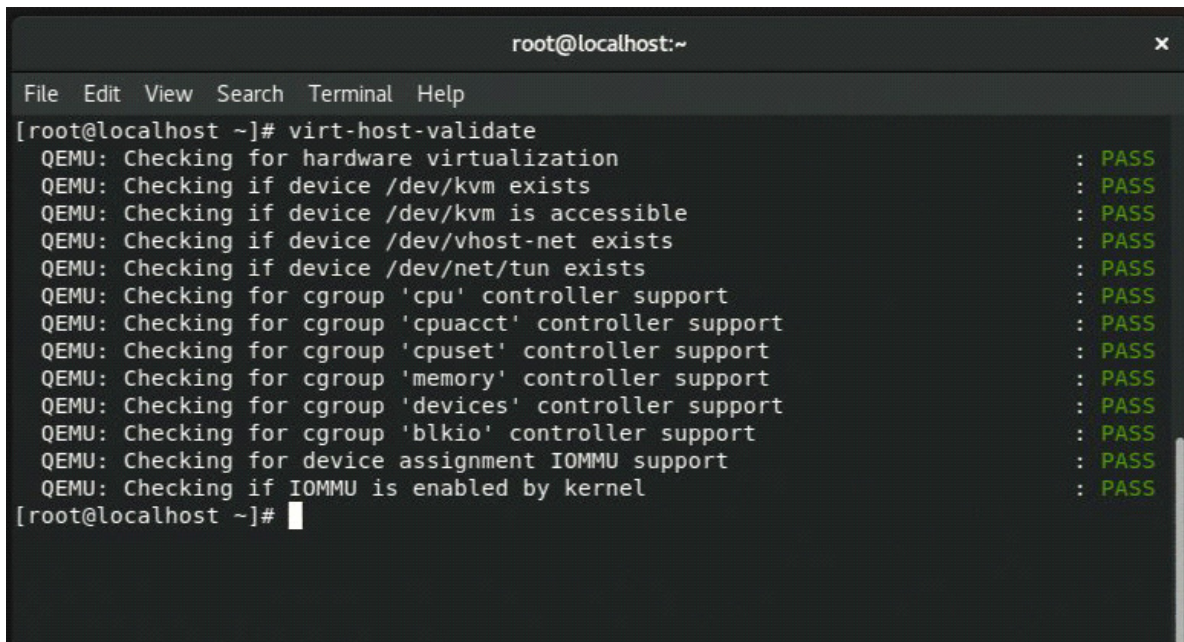
```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]#  
[root@localhost ~]# grubby --info DEFAULT  
index=0  
kernel="/boot/vmlinuz-4.18.0-147.el8.x86_64"  
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.l  
v=rhel/swap rhgb quiet $tuned_params intel_iommu=on iommu=pt"  
root="/dev/mapper/rhel-root"  
initrd="/boot/initramfs-4.18.0-147.el8.x86_64.img $tuned_initrd"  
title="Red Hat Enterprise Linux (4.18.0-147.el8.x86_64) 8.1 (Ootpa)"  
id="91101fcfdcf0473cb49e221c6b39a82c-4.18.0-147.el8.x86_64"  
[root@localhost ~]#
```

Figure 2. grubby Output with IOMMU Enabled

8. Reboot the server for the `iommu` change to take effect.
9. Use the following command to confirm that the machine is fully enabled for virtualization:

```
#virt-host-validate
```

The example in [Figure 3](#) shows the output from a successfully configured host.



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# virt-host-validate  
QEMU: Checking for hardware virtualization : PASS  
QEMU: Checking if device /dev/kvm exists : PASS  
QEMU: Checking if device /dev/kvm is accessible : PASS  
QEMU: Checking if device /dev/vhost-net exists : PASS  
QEMU: Checking if device /dev/net/tun exists : PASS  
QEMU: Checking for cgroup 'cpu' controller support : PASS  
QEMU: Checking for cgroup 'cpuacct' controller support : PASS  
QEMU: Checking for cgroup 'cpuset' controller support : PASS  
QEMU: Checking for cgroup 'memory' controller support : PASS  
QEMU: Checking for cgroup 'devices' controller support : PASS  
QEMU: Checking for cgroup 'blkio' controller support : PASS  
QEMU: Checking for device assignment IOMMU support : PASS  
QEMU: Checking if IOMMU is enabled by kernel : PASS  
[root@localhost ~]#
```

Figure 3. Configuration Output

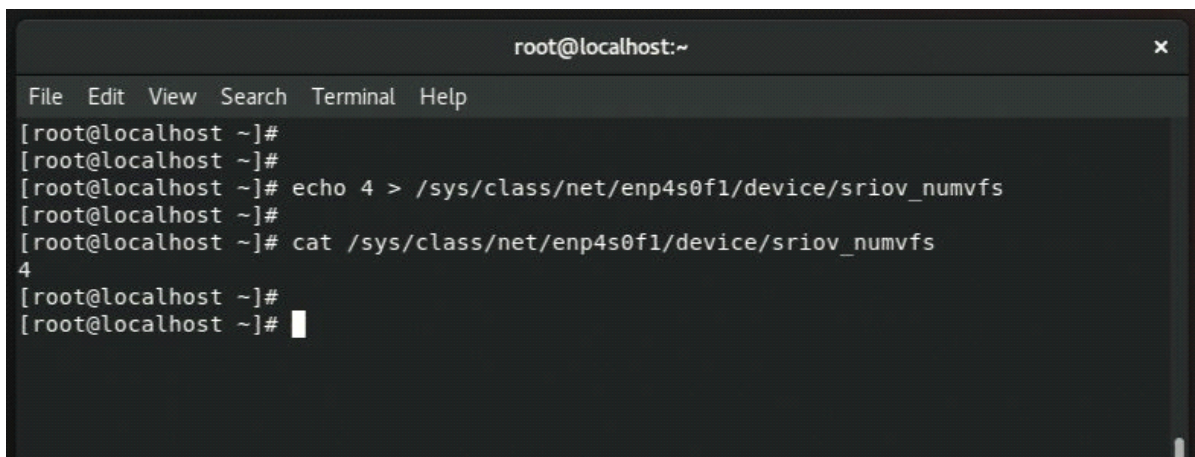
10. PF and VF drivers for the 800 Series are included in RHEL 8 distribution and are named as *ice* and *iavf*, respectively. These drivers are also available at Intel's Download Center and at SourceForge, as discussed in [Section 1.2.2](#). Using latest available drivers is strongly recommended.

2.2 VF Creation

1. The RHEL 8 installation does not create VFs by default. The 800 Series Network Adapters each support a total of 256 VFs. VFs can be created by writing an appropriate value to the `sriov_numvfs` parameter via `sysfs` interface.

```
#echo 4 > /sys/class/net/device name/device/sriov_numvfs
```

The example in [Figure 4](#) shows the creation of four VFs per port.



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# echo 4 > /sys/class/net/enp4s0f1/device/sriov_numvfs  
[root@localhost ~]#  
[root@localhost ~]# cat /sys/class/net/enp4s0f1/device/sriov_numvfs  
4  
[root@localhost ~]#  
[root@localhost ~]#
```

Figure 4. VF Creation via SysFS

The example in [Figure 4](#) shows four VFs being created on interface `enps0f1`, which in this case is the device name assigned by the Linux operating system to Port 1 of the 800 Series Network Adapter. The device name assigned on your system might be different.

The second command in [Figure 4](#) queries the `sriov_numvfs` parameter to verify that the value was updated.

2. Module options are not persistent from one boot to the next. To ensure that the desired number of VFs are created each time the server is power-cycled, append the above command to the `rc.local` file, located in the `/etc/rc.d/` directory. The Linux OS executes the `rc.local` script at the end of the boot process. The example in [Figure 5](#) shows contents of the `rc.local` file.

```

root@localhost:/etc/rc.d
File Edit View Search Terminal Help
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own systemd services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will NOT be run after all other services.
#
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
# that this script will be executed during boot.

touch /var/lock/subsys/local

echo 4 > /sys/class/net/enp4s0f1/device/sriov_numvfs
  
```

Figure 5. File Contents of rc.local

Note: Errors and informational messages during *iavf* driver load are logged in the */var/log/messages* file. It is good practice to review this file to confirm that the driver loaded successfully without warnings or errors.

- Use the `lspci` command to confirm that the VFs were successfully created. Use `grep` to narrow the results so that only virtual functions are displayed.

```
#lspci | grep Virtual
```

The example in [Figure 6](#) shows the result of this command.

```

root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]#
[root@localhost ~]# lspci | grep Virtual
04:01.0 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:01.1 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:01.2 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:01.3 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:11.0 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:11.1 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:11.2 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
04:11.3 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
[root@localhost ~]#
  
```

Figure 6. Output of lspci

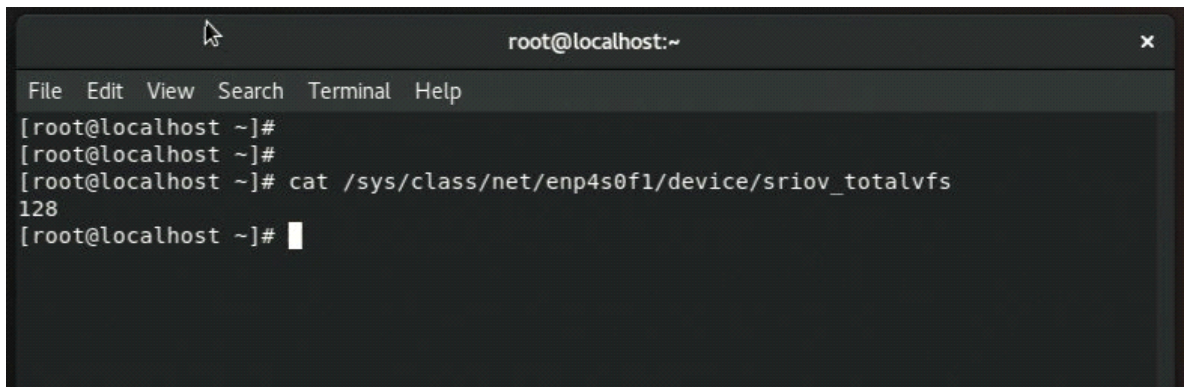
Figure 6 shows four VFs each for the physical Port 0 and Port 1 of the 800 Series Network Adapter. Each VF is identified by a unique bus, device, and function number. In the example, the first VF is assigned Bus #04, Device #01, and Function #0.

VFs with low device number belong to PF 0, which is Port 0. In the example above, the VF designated by 04:01:0 belongs to PF0. VFs with high device number belong to PF 1, which is Port 1. The VF designated by 04:11:0 belongs to PF1.

4. The maximum number of VFs per port supported by the adapter can be queried by reading the `sriov_totalvfs` parameter via `sysfs` interface.

```
#cat /sys/class/net/device name/device/sriov_totalvfs
```

The example in Figure 7 shows the maximum number of VFs supported by a given port.



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# cat /sys/class/net/enp4s0f1/device/sriov_totalvfs  
128  
[root@localhost ~]#
```

Figure 7. Total VFs Supported Query Output

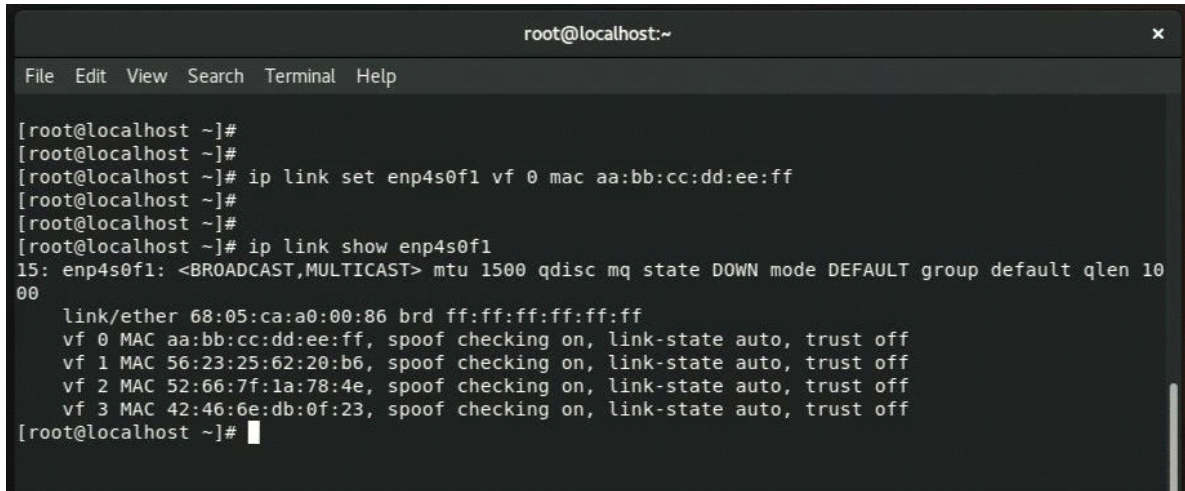
5. An application such as LibVirt or Virtual Machine Manager assigns a valid MAC Address to the VF before use. If a specific MAC Address needs to be used, the Intel *iavf* driver has built in security feature that allows system administrators to assign a valid MAC Address to a VF from within the host operating system. Once this is done, the VM that has the VF assigned to it is not allowed to change the VF MAC Address from within the VM. Ensure that each VF is assigned a unique MAC Address, as duplicate MAC Addresses cause loss of communication on the network. Use the following command to set a MAC Address for each VF.

```
#ip link set enp4s0f1 vf 0 mac aa:bb:cc:dd:ee:ff
```

Use the following command to confirm that the VF MAC Address assignment was completed successfully.

```
#ip link show enp4s0f1
```

Figure 8 shows an example of the results of this command.



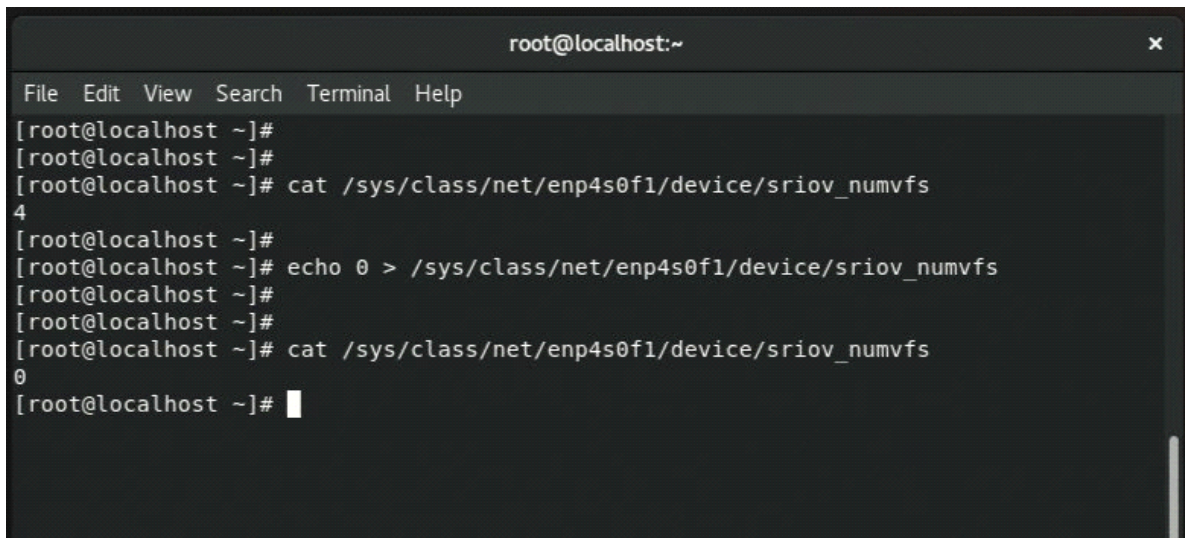
```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# ip link set enp4s0f1 vf 0 mac aa:bb:cc:dd:ee:ff  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# ip link show enp4s0f1  
15: enp4s0f1: <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN mode DEFAULT group default qlen 10  
00  
    link/ether 68:05:ca:a0:00:86 brd ff:ff:ff:ff:ff:ff  
    vf 0 MAC aa:bb:cc:dd:ee:ff, spoof checking on, link-state auto, trust off  
    vf 1 MAC 56:23:25:62:20:b6, spoof checking on, link-state auto, trust off  
    vf 2 MAC 52:66:7f:1a:78:4e, spoof checking on, link-state auto, trust off  
    vf 3 MAC 42:46:6e:db:0f:23, spoof checking on, link-state auto, trust off  
[root@localhost ~]#
```

Figure 8. VF MAC Address Query Result

6. To ensure each VF carries the same MAC Address assignment from one boot to the next, the commands from [Step 5](#) can be appended to the `rc.local` file, as is shown in [Step 2](#).
7. VFs can be destroyed or disabled by writing the value 0 to the `sriov_numvfs` parameter via `sysfs` interface.

```
#echo 0 > /sys/class/net/device name/device/sriov_numvfs
```

The example in [Figure 9](#) shows disabling SR-IOV on a given port.



```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# cat /sys/class/net/enp4s0f1/device/sriov_numvfs  
4  
[root@localhost ~]#  
[root@localhost ~]# echo 0 > /sys/class/net/enp4s0f1/device/sriov_numvfs  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# cat /sys/class/net/enp4s0f1/device/sriov_numvfs  
0  
[root@localhost ~]#
```

Figure 9. Disabling/Destroying VFs Example

2.3 VM Setup

RHEL 7.x includes tools for creating and managing VMs. These tools offer both Command Line Interface (CLI) and Graphical User interface (GUI). Virt-Manager is a GUI tool for creating and managing VMs.

1. Use virt-manager to create a VM.
2. Install the operating system of choice on the newly-created VM. For the purpose of this document, Ubuntu 16.04 desktop Linux was installed in the VM. See the example in [Figure 10](#).

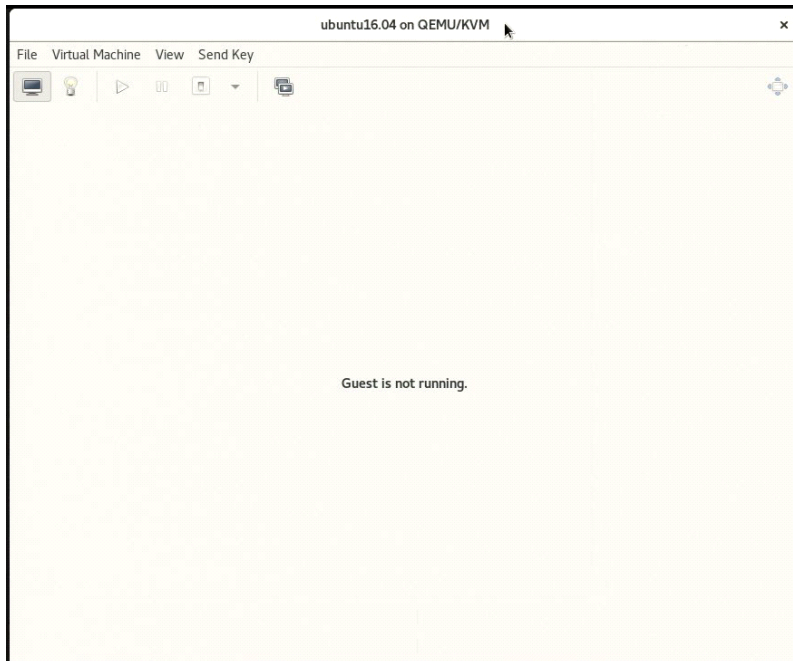



Figure 10. Virtual Machine “ubuntu16.04” Console Screen

3. Click on the  icon to edit the VM properties.
4. Click on the **Add Hardware** icon to start the **Add New Virtual Hardware** wizard, as shown in [Figure 11](#).

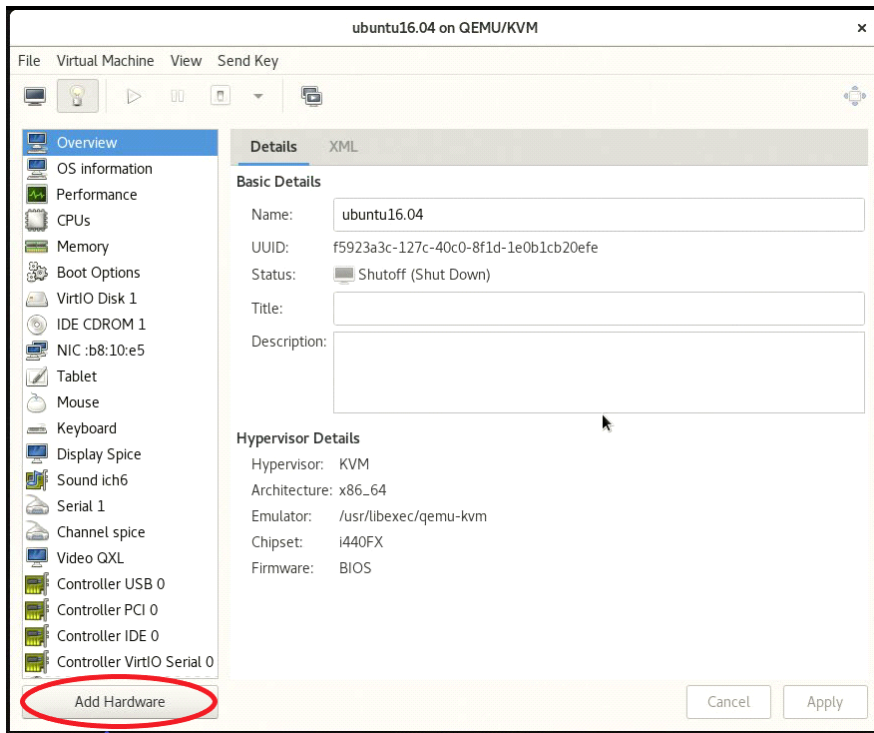


Figure 11. Virtual Machine Configuration Page

5. Click **PCI Host Device** to display the **Add New Virtual Hardware** window as shown in [Figure 12](#).

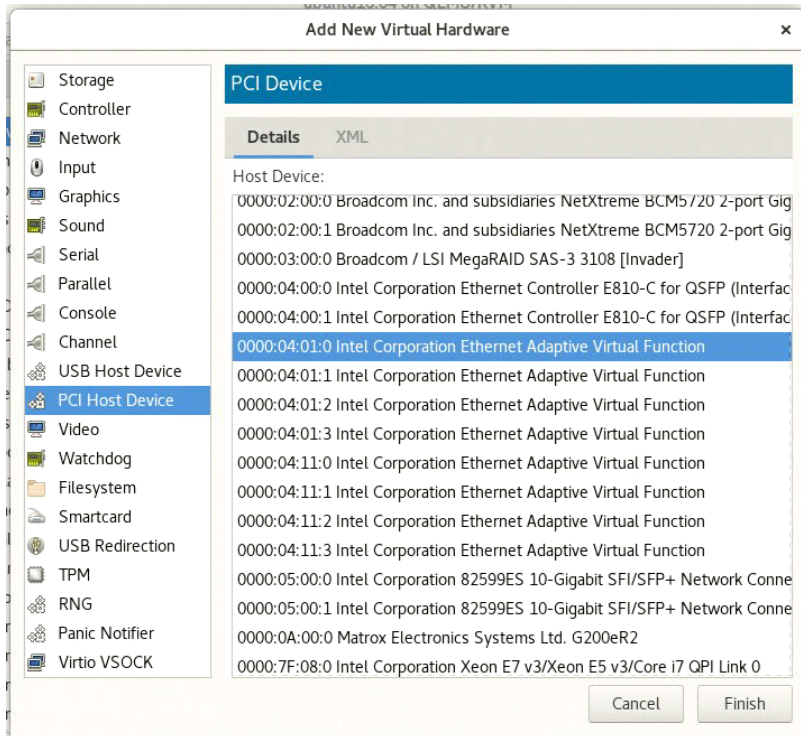


Figure 12. PCI Device Selection Page

In Figure 12, the Network Adapter's Virtual Functions are listed as "Intel Corporation Ethernet Adaptive Virtual Function". One or more VFs can be assigned to a VM. Upon successful assignment, the VM is ready to use.

6. Select a Virtual Function and click the **Finish**.
7. Power up the VM and log in using the credentials created during the VM installation process (see Section 2.1).
8. At the Linux Console, use the Linux `lspci` utility to confirm that the assigned VF is shown within the VM's PCIe hierarchy, as shown in Figure 13.



Figure 13. lspci Output of the VM

Did this document help answer your questions?

- Use the Linux `lsmod` utility to confirm that `iavf` driver for the VF has loaded successfully, as shown in Figure 14.

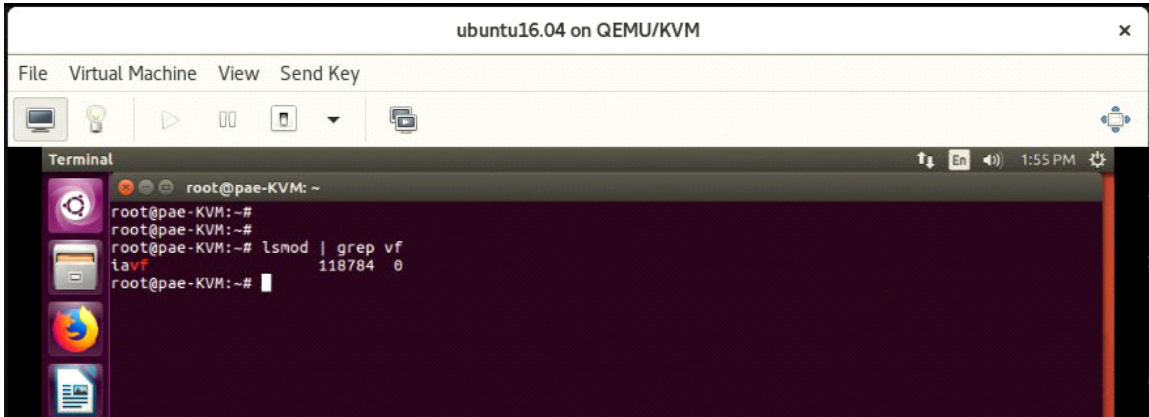


Figure 14. `lsmod` Output

- Use the Linux `iproute2` utility to confirm that the newly assigned VF is ready for use. The VF is named `ens9` in Section 15.

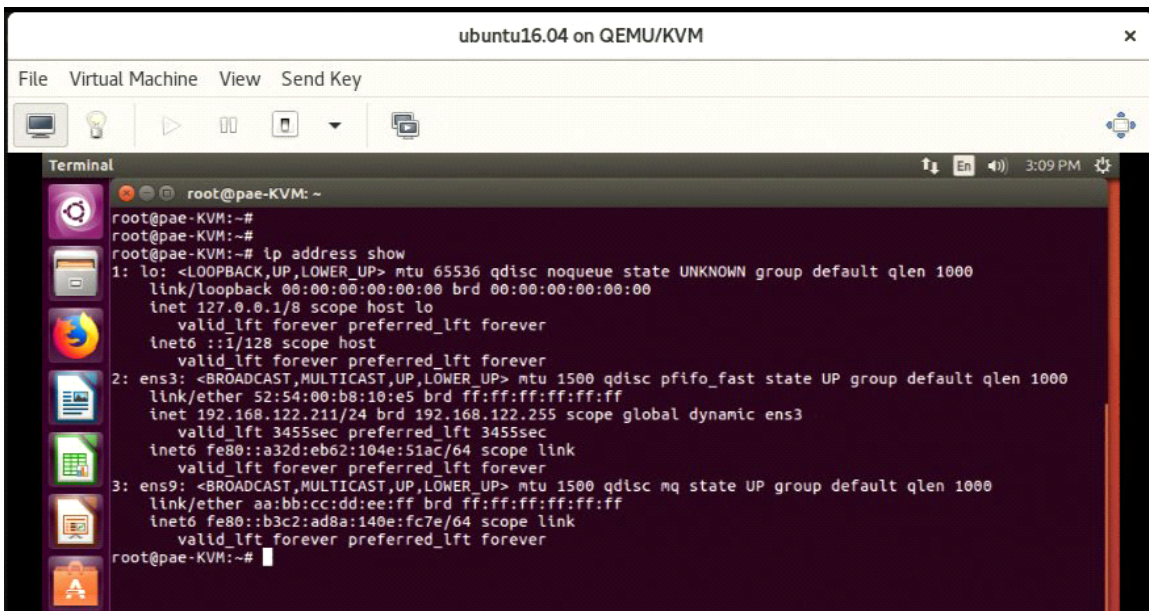


Figure 15. `lsmod` Output

- The VF can be configured for DHCP or static IP Address assignment. The VF is ready to communicate once it has an IP Address assigned.

3.0 Customer Support

Intel® Customer Support Services offers a broad selection of programs, including phone support and warranty service. For more information, contact us at:

www.intel.com/content/www/us/en/support.html

Service and availability may vary by country.

4.0 Product Information

To see the full line of Intel Network Adapters for PCI Express, visit:

<https://www.intel.com/content/www/us/en/products/network-io/ethernet.html>

To speak to a customer service representative regarding Intel products, please call 1-800-538-3373 (U.S. and Canada) or visit www.intel.com/content/www/us/en/support.html for the telephone number in your area.



LEGAL

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document (and any related software) is Intel copyrighted material, and your use is governed by the express license under which it is provided to you. Unless the license provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this document (and related materials) without Intel's prior written permission. This document (and related materials) is provided as is, with no express or implied warranties, other than those that are expressly stated in the license.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

Other names and brands may be claimed as the property of others.

© 2020-2021 Intel Corporation.