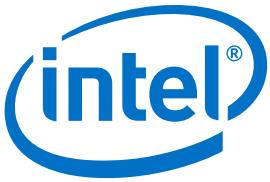
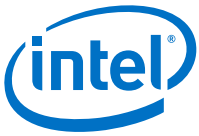


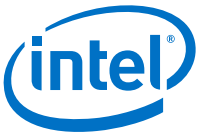
Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology

Application Note

August 2017

Document Number: 330689-006





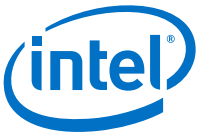
Contents

Tables

Revision History

Date	Revision	Description
August 2017	006	• Removed section on using QEMU* KVM command line interface • Added device ID for Intel® C62x Chipset and Intel Atom® C3000 Processor Product Family
July 2016	005	Updated: • Section 1.2.2, “Limitations” on page 5 • Section 1.4, “Software Requirements” on page 6 • Section 2.2, “Installing and Configuring the Host Operating System” on page 8 • Section 2.3, “Installing the Guest OS Image” on page 9 • Section 2.4.1, “Using the libvirt* Virtual Machine Manager GUI” on page 10 • Section 2.4.3, “Pass-through the PCI Device” on page 12 • Section 2.4.2.1, “Installing Updated QEMU* KVM” on page 14 • Section 2.4.2.2, “Pass-through the PCI Device” on page 15 • Section 2.4.2.4, “Starting the Guest” on page 16 • Section 2.5, “Running Acceleration Services Simultaneously in Host and Guest” on page 15 • Appendix A, “FAQ”
February 2015	004	Updated: Section 1.4, “Software Requirements” on page 6 Section 2.4.1, “Installing Intel® QuickAssist Technology Software on Host” on page 10 Section 2.5, “Running Acceleration Services Simultaneously in Host and Guest” on page 15
November 2014	003	• Updated Section 2.5, “Running Acceleration Services Simultaneously in Host and Guest” on page 15. • Added Appendix A, “Yes. If you want to run Intel® QuickAssist Technology Software and Intel® DPDK in the guest you will need Host CPU. By default, the hypervisor provides emulated CPU to the guest. If you are using the Virt-Manager GUI, go to **Guest Machine details, then Processor > Configuration and select Copy host CPU configuration. If you are using the qemu command line, use the** `-cpu host` parameter to provide host CPU to guest.”
September 2014	002	• Updated Section 2.5, “Running Acceleration Services Simultaneously in Host and Guest” on page 15.
July 2014	001	Updates include: • First “public” version of the document. Based on “Intel confidential” document number 476488-1.4 with the revision history of that document retained for reference purposes. • Updated Section 1.3.1, “Where to Find Current Software and Documentation” on page 5. • Removed Fedora 14 information from Section 1.4, “Software Requirements” on page 6 and Section 2.4.1, “Using the libvirt* Virtual Machine Manager GUI” on page 10. • Added new step at the end of Section 2.2, “Installing and Configuring the Host Operating System” on page 8. • Updated Section 2.5, “Running Acceleration Services Simultaneously in Host and Guest” on page 15.
March 2014	1.4	Updates include: • Modified step 8 in Section 2.3, “Installing the Guest OS Image” on page 9. • Added Section 2.5, “Running Acceleration Services Simultaneously in Host and Guest” on page 15.
June 2013	1.3	Updates to make applicable to multiple platforms that use Intel® QuickAssist Technology.
February 2013	1.2	• Added new FAQ items: 1 and 1; deleted outdated FAQ items.
October 2012	1.1	Added Limitation.
September 2012	1.0	Initial release of this document.

§ §



1.0 Introduction

1.1 About this Document

This document discusses the following topics related to using Intel® Virtualization Technology (Intel® VT) with the Intel® QuickAssist Technology Software:

- Features and limitations
- Build and installation

Users of this document are expected to be familiar with virtualization technologies.

In this document, for convenience:

- *Software package* is used as a generic term for the Intel® QuickAssist Technology Software package.
- *Acceleration drivers* is used as a generic term for the software that allows the QuickAssist Software Library APIs to access the Intel® QuickAssist Accelerator(s) integrated in the Intel® QuickAssist Technology.

1.2 About the Software

This section lists the features and limitations.

1.2.1 Features

- PCI pass-through with Kernel-based Virtual Machine (KVM)
- SR-IOV with KVM

1.2.2 Limitations

- SR-IOV may not work on GNU*/Linux* kernel versions older than 2.6.38.
- KVM limitation: the maximum number of Virtual Functions that can be mapped to a single VM is specific to qemu-kvm version.

1.3 Documentation

1.3.1 Where to Find Current Software and Documentation

Associated software and collateral can be found on the open source website: https://01.org/intel-quickassist-technology

Table 1 includes a list of related documentation.

1.3.2 Product Documentation

Documentation includes:

- Using Intel® Virtualization Technology (Intel® VT) with Intel® QuickAssist Technology Application Note (this document)
- Additional related documents listed in Table 1, which may be accessed as described in Section 1.3.1.

Table 1. Related Documents

Document Name	Number
Intel® QuickAssist Technology Software Release Notes	330683
Intel® Communications Chipset 8900 to 8920 Series Software for Linux* Getting Started Guide	330752
Intel® Communications Chipset 8925 to 8955 Series Software for Linux* Getting Started Guide	330750
Intel® Communications Chipset 8900 to 8920 Series Software Programmer's Guide	330753
Intel® Communications Chipset 8925 to 8955 Series Software Programmer's Guide	330751
Intel® QuickAssist Technology API Programmer's Guide	330684
Intel® QuickAssist Technology Cryptographic API Reference Manual	330685
Intel® QuickAssist Technology Data Compression API Reference Manual	330686
Intel® QuickAssist Technology for Linux* Release Notes	330683
Intel® QuickAssist Technology for Linux* Getting Started Guide	336212
Intel® Communications Chipset 89xx Series Datasheet	327879
Intel® QuickAssist Technology Software for Linux* - Getting Started Guide - HW version 1.7	336212
Intel® QuickAssist Technology Software for Linux* - Release Notes - HW version 1.7	336211
Intel® QuickAssist Technology Software for Linux* - Programmer's Guide - HW version 1.7	336210
Intel® QuickAssist Technology Driver for Linux* - HW version 1.7	

Note: Sample configuration files are included with the software package.

1.3.3 Documentation Conventions

The following conventions are used in this manual:

- `Courier font` - code examples, command line entries, API names, parameters, filenames, directory paths, and executables
- **Bold text** - graphical user interface entries and buttons

1.4 Software Requirements

Software requirements will vary by the particular use case.

- Required: the Intel® QuickAssist Technology Software for Linux*
  Intel recommends using the same version of the QuickAssist driver on both host and guest OS. Consult your Intel representative if you have a requirement to use different versions of the driver.
  However, changes in the QAT1.6 driver from QATmux1.1.0 onwards are generally done in a backwardly compatible manner and there is a version compatibility check between drivers running on Host and Guest when bringing up each device on the Guest.

The following table details the Intel® QuickAssist Technology driver compatibility.

- A driver running on the **Host** is compatible with a driver on the Guest running any version between its version and its LowestSupportVersion
- A driver running on the **Guest** is compatible with a driver on the Host running any version between its version and its LowestSupportVersion

Device	Driver	Driver Version	Lowest Supported Version (per device)	Comment
DH89xxcc	QAT1.5	x.y.z	x.y.z	Driver version must be the same on Host and Guest
DH895xcc	QAT1.6	2.0.x - 2.2.x	2.0.0	For example, version 2.2.1 on Host and 2.1.0 on Guest are compatible
DH895xcc	QAT1.6	2.3.x - latest	2.3.0	
Hardware v1.7	QAT1.7	x.y.z	x.y.z	Driver version must be the same on host and guest.

These instructions were tested against the following Linux distribution:

- CentOS*

1.5 Supported Intel® QuickAssist Technology (QAT) Endpoints and their Device IDs

Intel® QAT Endpoint	Physical Function (PF) Device ID	VF Device ID
8900-8920	0434	0442
8925-8955	0435	0443
Intel® C62x Chipset	37c8	37c9
Intel Atom® C3000 Processor Product Family	19e2	19e3

2.0 Using Intel® QAT Software with KVM

The Intel® Virtualization Technology can use both Single-Root I/O Virtualization (SR-IOV) and PCI pass-through for the acceleration services. SR-IOV enables the creation of Virtual Functions from a single Intel® QuickAssist Technology acceleration device to support acceleration for multiple virtual machines. If you do not need to share a single PCH device with accelerator capabilities between multiple virtual machines, PCI pass-through is sufficient. The following sections describe the steps necessary to enable this functionality, with a focus on the SR-IOV use case.

2.1 Updating the BIOS Settings

Note: The BIOS settings for your system may differ from the following steps.

1. Power on the development board. Watch closely for the prompt to enter BIOS setup. Press **F2** when prompted.
2. Enable the VT-d parameter in BIOS. The option may be available under:
   Advanced > System Agent (SA) Configuration > VT-d
3. Enable the SR-IOV parameter in BIOS. The option may be available under:
   Advanced > System Agent (SA) Configuration > SRIOV

   Note: Enabling the SR-IOV BIOS parameter is not required if you are not using SR-IOV.
4. Press **F4** to Save and Exit. The BIOS changes are saved and the system will boot.

2.2 Installing and Configuring the Host Operating System

1. Install the CentOS 7 64-bit version. Consult the *Getting Started Guide* section "*Installing the OS on a Development Board*" if necessary, taking note that this guide assumes one of those CentOS 7 64-bit versions as the host OS when SR-IOV is used.

   Note: The *Getting Started Guide* uses the kernel boot parameter `intel_iommu=off`. That kernel boot parameter must not be included for the purposes of this guide. If this kernel boot parameter is set, remove it, using the *Getting Started Guide* as a reference to update grub2, and reboot.

   Note: Some Linux distributions may require the kernel boot parameter `intel_iommu=on` if using SR-IOV (refer to question 2 in Appendix A, "FAQ")

   Note: CentOS 7 requires the `intel_iommu=on` kernel boot parameter to use SR-IOV and VT-d functionality.

2. Install virtualization related packages using the following command (root privileges required):

   ```
   # yum -y install @virtualization
   ```

   Note: Alternatively, use `yum -y groupinstall Virtualization`

   Note: This will install qqemu-kvm qemu-img virt-manager libvirt libvirt-python python-virtinst libvirt-client virt-install virt-viewer and all of the dependencies that are needed.

3. If the libvirtd service is not running, start it by using the commands:

   ```
   # chkconfig libvirtd on
   # service libvirtd start
   ```

4. Verify SR-IOV hardware capabilities using the command:

   ```
   # lspci -vnc 8086:<Device ID>
   ```

   For Intel® Communications Chipset 8900 to 8920 Series:

   ```
   # lspci -vnd 8086:0434
   ```

   Refer to Section 1.5 for supported devices and their device IDs.

   It should display one of the capabilities as:

   ```
   Capabilities: [140] Single Root I/O Virtualization (SR-IOV)
   ```

5. If the Intel® QuickAssist Technology software package has a configure script, enable the SR-IOV build on the host by using:

   ```
   # ./configure --enable-icp-sriov=host
   ```

6. Verify BIOS settings using the command:

   ```
   # lsmod | grep kvm
   kvm_intel              42122  0
   kvm                   257132  1 kvm_intel
   ```

7. Ensure that the system supports VT extensions:

   ```
   # egrep '^flags.*(vmx|svm)' /proc/cpuinfo
   ```

   Note: If nothing is printed out after executing the above command, then the system does not support VT extensions.

8. If kernel boot parameters changed, shut down the system:

   ```
   # shutdown -h now
   ```

9. Power on the system and proceed with the instructions in the following sections.

10. Once the system is restarted, check for DMAR and IOMMU messages, similar to the following:

    ```
    # dmesg | grep -e DMAR -e IOMMU
    [    0.000000] ACPI: DMAR 000000007b79c000 00080 (v01 INTEL  INTEL ID 00000001
    INTL 20091013)
    ```

```
[    0.000000] Intel-IOMMU: enabled
[    0.064454] dmar: IOMMU 0: reg_base_addr fbffc000 ver 1:0 cap d2078c106f0466
ecap f020df
[    0.065560] IOAPIC id 8 under DRHD base  0xfbffc000 IOMMU 0
[    0.065919] IOAPIC id 9 under DRHD base  0xfbffc000 IOMMU 0
[    2.168898] DMAR: No ATSR found
[    2.169358] IOMMU 0 0xfbffc000: using Queued invalidation
[    2.169728] IOMMU: Setting RMRR:
[    2.170091] IOMMU: Setting identity map for device 0000:00:1d.0 [0x7a23f000
- 0x7a241fff]
[    2.170767] IOMMU: Prepare 0-16MiB unity mapping for LPC
[    2.171133] IOMMU: Setting identity map for device 0000:00:1f.0 [0x0 -
0xffffff]
```

Note: If the above command fails, a BIOS update or kernel reconfiguration may be required.

2.3 Installing the Guest OS Image

This section describes how to use the libvirt* Virtual Machine Manager GUI to create the guest OS installation.

Note: The instructions in this section use the Graphical User Interface (GUI) approach; information on using the command line interface (CLI) is available at: http://libvirt.org/virshcmdref.html

Note: Using the steps below, enter the root password when prompted.

1. Start the Virtual Machine Manager GUI by selecting it from the top main menu: **Applications > System Tools > Virtual Machine Manager**.
2. Open a connection to a Hypervisor by choosing **File > Add Connection**.
3. Choose **QEMU/KVM** for Hypervisor.
4. Make sure **Connect to remote host** is NOT checked.
5. Make sure **Autoconnect** is checked.
6. Click the **Connect** button.
7. After a connection is opened, select the **localhost (QEMU)** and right click to select **New**.

   The **New VM** window is displayed.

   a. **Create a new virtual machine (Step 1 of 5)**: Enter the **Name** for the Guest virtual machine, select **Local install media (ISO image or CDROM)**, and click the **Forward** button.

   b. **Create a new virtual machine (Step 2 of 5)**: Select **Use CDROM or DVD**, insert the OS installation CDROM/DVD into the CDROM/DVD drive, and make sure that the mounted CDROM appears in box [**Media Unknown (dev/sr0)**]. Select the OS type and version, and click the **Forward** button.

   c. **Create a new virtual machine (Step 3 of 5):** Choose **Memory (RAM)** in MB and number of **CPUs** settings (assign a sufficient amount, but it should not affect the Host OS, e.g., for 4 GB RAM and 8 cores, allocate Guest OS < 2 GB RAM and 4 cores CPU). Click the **Forward** button.

   Note: Many platforms will show twice the actual number of cores due to simultaneous multithreading.

d. **Create a new virtual machine (Step 4 of 5)**: Make sure **Enable storage for this virtual machine** is checked. Select the **Create a disk image on the computer's hard drive** and specify a sufficient amount of hard drive space in GB (20 GB is recommended, and at least 18 GB may be required). Make sure **Allocate entire disk now** is checked. Click the **Forward** button.

e. **Create a new virtual machine (Step 5 of 5)**: Review the information from Steps 1 through 4. Note the **Ready to begin installation of <Name>** and the **Storage** path to the Guest virtual machine image (this will be used if using the QEMU CLI). Click the **Finish** button to begin the installation of the Guest OS.

8. Follow the steps provided in the "*Installing CentOS*" section of the appropriate Getting Started Guide to install the Guest OS.

   When installing CentOS, if the following error message appears:

   "Cannot display graphical console type 'spice': /usr/lib64/libspice-client-gLib-2.0.so.1: Undefined symbol: usbredirhost_get_guest_filter"

   Install the `usbredir` package with the following command:

   ```
   # yum install usbredir
   ```

9. Shut down the guest OS.

   By default, the guest image is created in the `/var/lib/libvirt/images` directory. This image can be used by libvirt APIs (virsh tools) and qemu-kvm to run the guest.

2.4 Installing and Configuring Intel® QuickAssist Technology Software

The following sections detail the steps to use the libvirt* Virtual Machine Manager GUI, though similar steps are possible using the command line interface.

2.4.1 Installing Intel® QuickAssist Technology Software on Host

Note: If you are not using SR-IOV and are instead passing through a Physical Function (PF) for acceleration services on one guest only, it is not required to install the Intel® QuickAssist Technology Software package on the host.

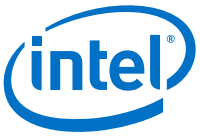
Note: The `installer.sh` or configure script included with the software package will automatically take care of certain build environment details, including setting any SRIOV environment variable to the correct value and copying over the correct sample configuration files. If you are not using an included script to build and install the software, you must perform these operations yourself, using the included script as a guide.

1. Install the Intel® QuickAssist Technology Software package on the host, choosing first the **Change Configuration** option and then the **Set SRIOV Mode to "Host"** option. Alternatively, configure the corresponding flags in the "configure" command. Consult the *Getting Started Guide* (refer to Table 1 on page 6) for more information.
2. Check the log file for any error messages. `InstallerLog.txt` is appended after each installation with the time/date and the output of the build/install. If any issues were seen during the installation, check the log file for details.

2.4.2 Verifying SR-IOV on the Host

Note: If you are not using SR-IOV, skip this section.

Note: Sample configuration files have been included in the software package.



1. Optional: View the sample SR-IOV configuration files that were copied to the `/etc` directory. Note that the sample SR-IOV configuration file has `SRIOV_Enabled = 1` and it sets the number of kernel service instances to 0.
2. Verify the VFs by running the following command in the host OS. As an example, with one Intel® Communications Chipset 8900 to 8920 Series SKU4 in the system, the output would have 16 0442 devices, as shown below:

```
# lspci | grep 0442
01:01.0 Co-processor: Intel Corporation Device 0442 (rev 21)
01:01.1 Co-processor: Intel Corporation Device 0442 (rev 21)
01:01.2 Co-processor: Intel Corporation Device 0442 (rev 21)
...
01:01.7 Co-processor: Intel Corporation Device 0442 (rev 21)
01:02.0 Co-processor: Intel Corporation Device 0442 (rev 21)
01:02.1 Co-processor: Intel Corporation Device 0442 (rev 21)
...
01:02.6 Co-processor: Intel Corporation Device 0442 (rev 21)
01:02.7 Co-processor: Intel Corporation Device 0442 (rev 21)
```

   Note: Do not detach 8086:043e; it is used for debug purposes.

```
Co-processor [0b40]: Intel Corporation Device [8086:043e]
```

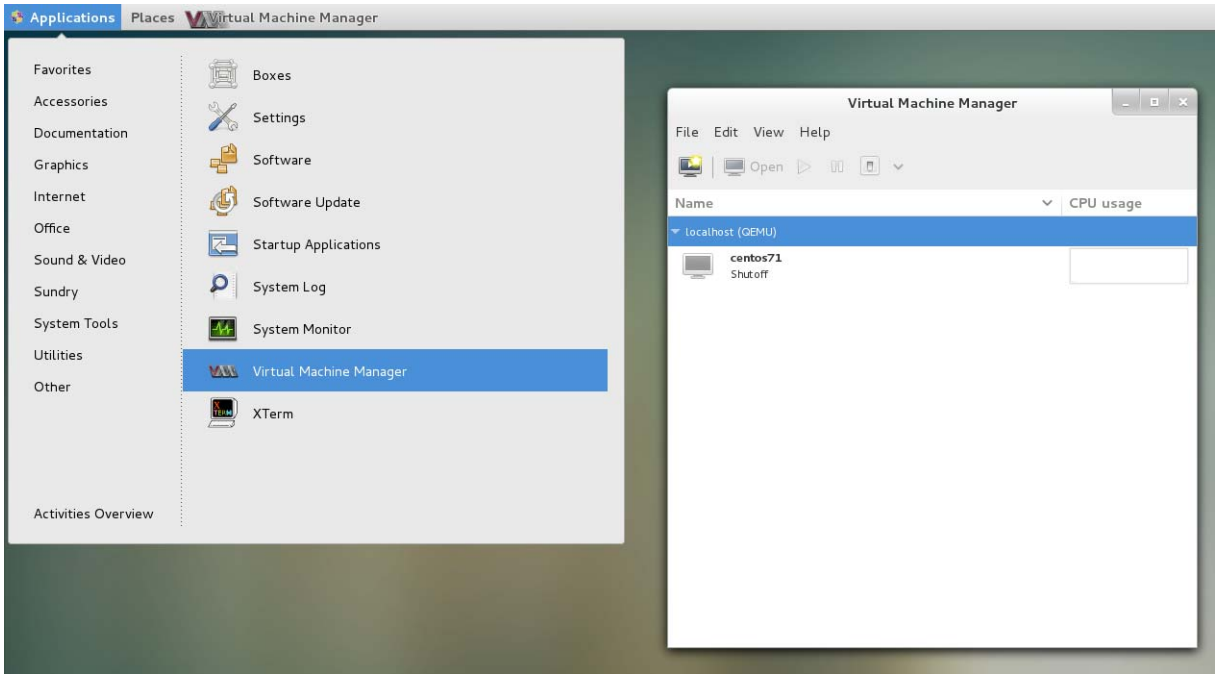
   As another example, with one Intel® Communications Chipset 8925 to 8955 Series device in the system, the output would have 32 0443 devices, as shown below:

```
# lspci | grep 0443
01:01.0 Co-processor: Intel Corporation Device 0443
01:01.1 Co-processor: Intel Corporation Device 0443
01:01.2 Co-processor: Intel Corporation Device 0443
...
01:01.7 Co-processor: Intel Corporation Device 0443
01:02.0 Co-processor: Intel Corporation Device 0443
01:02.1 Co-processor: Intel Corporation Device 0443
...
01:02.6 Co-processor: Intel Corporation Device 0443
01:02.7 Co-processor: Intel Corporation Device 0443
01:03.0 Co-processor: Intel Corporation Device 0443
01:03.1 Co-processor: Intel Corporation Device 0443
...
01:03.6 Co-processor: Intel Corporation Device 0443
01:03.7 Co-processor: Intel Corporation Device 0443
01:04.0 Co-processor: Intel Corporation Device 0443
01:04.1 Co-processor: Intel Corporation Device 0443
...
01:04.6 Co-processor: Intel Corporation Device 0443
01:04.7 Co-processor: Intel Corporation Device 0443
```

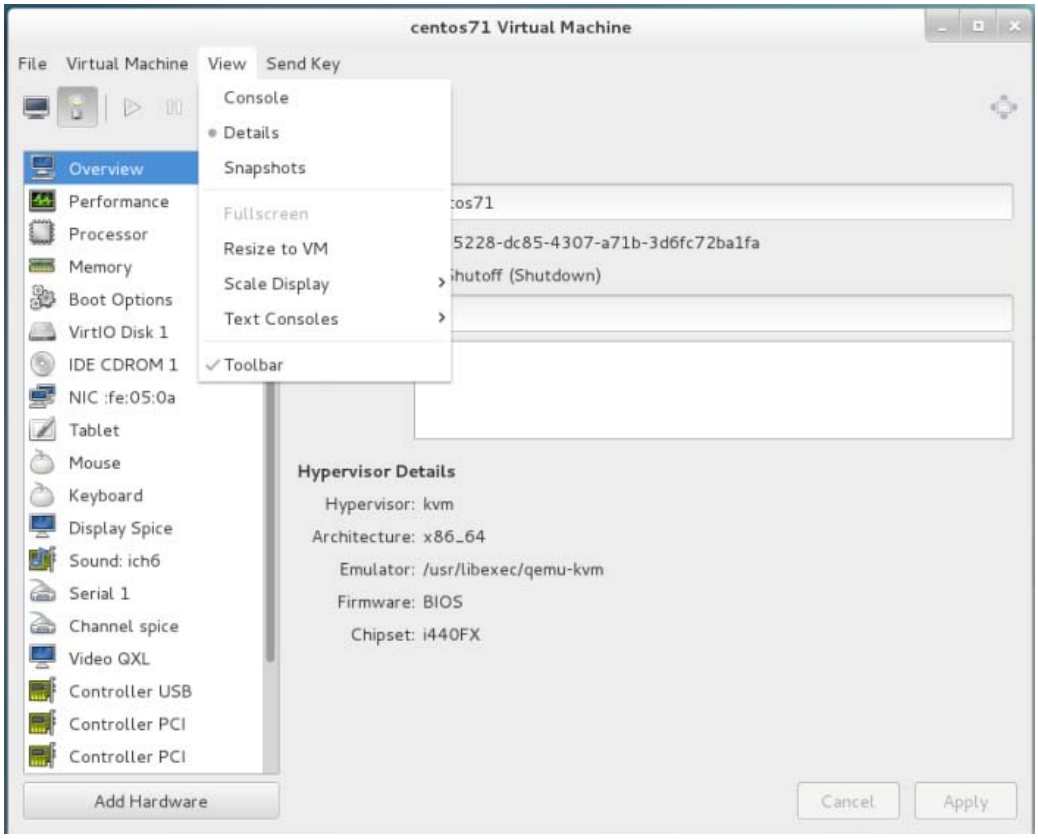
   Refer to Section 1.5 for supported devices and their device IDs.

2.4.3 Pass-through the PCI Device

1. Start Virtual Machine Manager using **Application > System Tools > Virtual Machine Manager**.

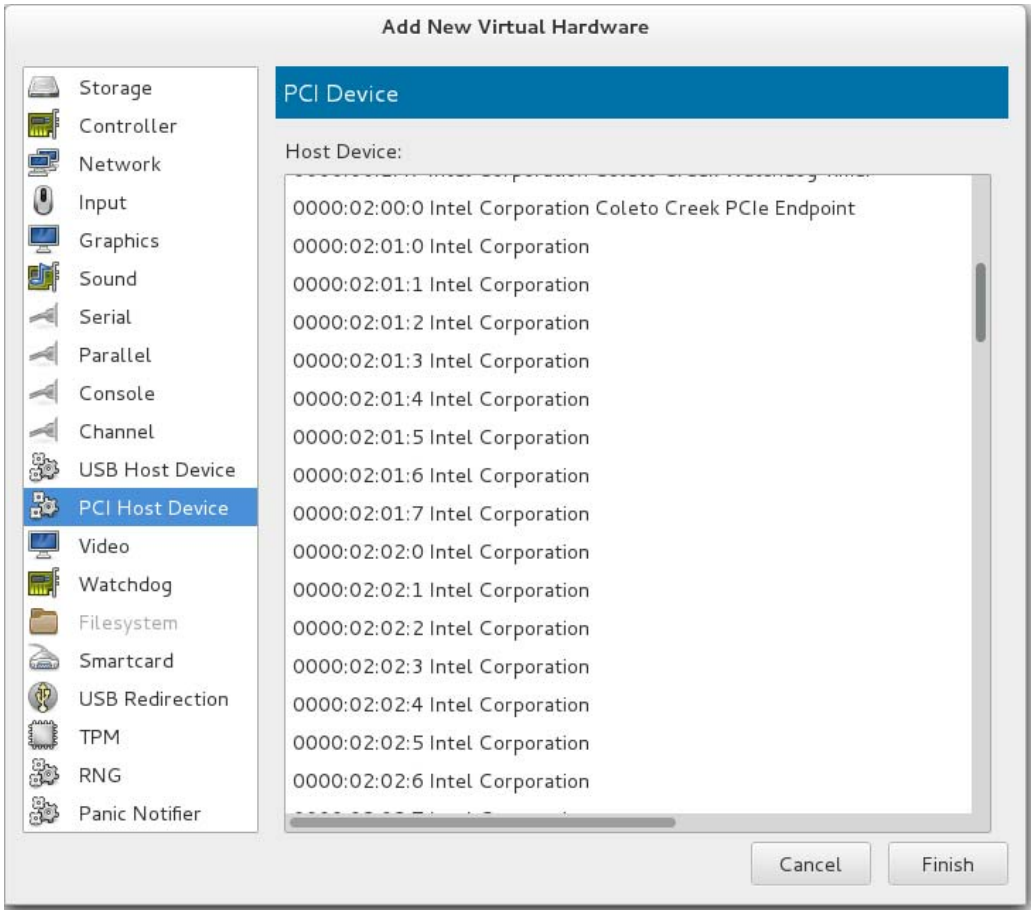


2. Right-click on the guest and click **Open** (Do not run the guest).
   A new window for the virtual machine is displayed. Go to **View > Details**.



Configure the processor, memory, boot options, and virtual hardware for the guest.

3. To add co-processor virtual functions (Refer to Section 1.5 for supported devices and their device IDs) or GigE ports, click **Add Hardware** in the bottom-left corner and click **PCI Host Device**.



   Select the appropriate PCI device (for instance, in the figure above, 02:01:1 is one of the 0442 devices) to attach to Guest and click **Finish**. This newly added device should appear in the left column of details for the Guest.

   Note: This action will internally unbind the PCI device from the Host driver currently being used and bind it to pci-stub (CentOS 6.5) or vfio-pci (CentOS 7.1). If using a CLI, a similar sequence is: `virsh-detach <pci_func>` and `virsh-attach <domain> <pci_func>`

   Note: Do not use pass-through for any 0442 devices with function 0 (for instance, “02:01.0”). These virtual functions are used by the physical functions.

4. Optional: To detach a PCI device from the guest, click the PCI device to be detached from the details page left column and click **Remove** (bottom row).

   Note: You can add and remove some PCI devices while the guest is running.

5. To run the guest, go to **Virtual Machine > Run** or click the **Play** radio button on the Menu bar.

6. To view the guest console, go to **View > Console**.

2.4.4 Installing Intel® QuickAssist Technology Software on the Guest

1. In the Guest OS, verify that the appropriate device has been passed through (see Section 2.4.3), as evidenced by the `lspci` command.
   a. For pass-through of Virtual Function (VF) using SR-IOV, this should be one or more 0442, 0443, 37C9, or 19e3 devices.
   b. For pass-through of PF, this should be a 0434, 0435, 37C8, or 19e2 device.
2. Install the Intel® QuickAssist Technology Software package on the Guest.
   a. For pass-through of VF using SR-IOV, choose the **Install SR-IOV Guest Acceleration** option.
   b. For pass-through of PF, choose the **Install Acceleration** option. Alternatively, set the corresponding configuration parameters.

   Refer to the Getting Started Guide (Table 1) for more information.
3. If the Intel® QuickAssist Technology software package has a configure script, enable the SR-IOV build on the host by using:
   ```
   # ./configure --enable-icp-sriov=guest
   ```
4. Check the log file for any error messages. `InstallerLog.txt` is appended after each installation with the time/date and the output of the build/install. If any issues were seen during the installation, check the log file for details.

Note: View the sample VM configuration file `/etc/dh<device>cc_qa_dev0.conf`. Note that this configuration file supports a limited number of service instances. Specifically, the limitations is a budget of 16 rings per VF. Refer to the relevant Programmer's Guide (Table 1) and Appendix A, "FAQ" for more information on the configuration file formats. More 0442 devices can be passed through if more service instances are required.

2.5 Running Acceleration Services Simultaneously in Host and Guest

Follow the steps below to run acceleration services simultaneously.

Using SR-IOV, acceleration running on the Host does not use the PF; instead, it uses one or more VFs. These VFs cannot then be assigned to any Guest.

Note: With the older driver, acceleration running on the host could use the PF, but not the VF.

Note: For CentOS 7.2 or 7.3, update the kernel boot parameters to include "intel_iommu=on iommu=pt". Refer to the Getting Started Guide (Table 1) as a reference to update grub2, and reboot.

1. Install the SR-IOV Host acceleration on the Host machine as outlined in Section 2.4.1.
2. Edit the <dh89xxcc/dh895xcc/c6xx/c3xx>_qa_dev0.conf file in the host to add Crypto and Compression instances. Use the other configuration files that are included in the Intel® QuickAssist Technology Software package as a guide to ensure the correct syntax.

Note: If more than one acceleration device is present, the following commands will need to consider the configuration of dev1, dev2, etc.

The following sample shows the partial modifications to the USER section for establishing two crypto and one compression instance:

```
[SSL]
NumberCyInstances = 2
NumberDcInstances = 1
NumProcesses = 1
```

```
# Crypto - User instance #0
Cy0Name = "SSL0"
:
# Crypto - User instance #1
Cy1Name = "SSL1"
:
# Data Compression - User space  instance #0
Dc0Name = "UserDC0"
```

3. If using the "v1" config file (refer to Table 1 for related Programmer's Guide), note that the VFs will use bank numbers that correspond to their VF number. For instance, the first VF uses bank number 0, the second VF uses bank number 1, and so on. If the configuration file on the Host uses any resources from a given bank it is not possible to use that VF on a Guest.

   If using the "v2" config file, uncomment the line: `PF_bundle_offset = 0`. Change the value from 0 to the first bundle number that will be used to allocate instances for the PF. Then, for each instance that is configured in the User Process Instance Section, a bundle is assigned to that instance in sequence. Each bundle that is assigned is not available for use by any VF. Refer to the “General Parameters” section in the related Programmer's Guide for more information.

Note: Do not use pass-through for any 0442 devices with function 0 (for instance,“02:01.0 and 02:02.0”); these virtual functions are used by the physical functions.

4. Restart the acceleration service.

   ```
   # service qat_service restart
   ```

5. Use any of the free VFs and passthrough to the guest.
6. After verifying that `qat_service` is running without any issues in Host and Guest, user space code can be executed simultaneously in the Host and Guest.

2.6 Enabling Virtual Functions in QAT

To enable the virtual functions for Intel® QuickAssist Technology endpoints designed to work with the Hardware Version 1.7 software packages, write a `1` to the sys file for that endpoint, replacing `$bdf` with the `bus/device/function` reference for that endpoint. For instance:

```
– # echo 1 > /sys/bus/pci/devices/0000:$bdf/sriov_numvfs
```

Note that `qat_service` handles writing to `sriov_numvfs` to restore the VFs, but `adf_ctl` does not.

Virtual functions can now be used on the host (managed via `qat_service` or `adf_ctl`) as well as the guest. Because the VFs can be used on the host, so-called “PF/VF concurrency” is not supported. Furthermore, physical function passthrough of QAT endpoints may not be supported.

Appendix A FAQ

1. **How can I achieve load balancing when multiple VFs are being used?**

   Note: This FAQ only applies to the Intel® Communications Chipset 8900 to 8920 Series.

   The top two PCH SKUs have two accelerators, each with two crypto engines and one compression engine. For the top SKU, VF numbers 1-7 use the hardware resources (i.e., engines) of the first accelerator, and VF numbers 9-15 use the hardware resources (i.e., engines) of the second accelerator. To achieve load balancing when using one or more VFs from one PCH device, it is necessary to take special consideration to select sensible values for the AcceleratorNumber parameter (plus the ExecutionEngine parameter, if using configuration file version 2) in the particular guest configuration file(s).

   For instance, take the case where VF#1 is passed through to a guest OS. By default, the configuration file for the guest has two crypto instances: one is assigned to the first crypto engine for the first accelerator, and one is assigned to the second crypto engine for the first accelerator. If running crypto on this VF, the application can load balance between these two engines. If two VFs are passed through, the second VF should be chosen from the range 9 to 15 for the best load balancing, since the second VF would be otherwise competing with the first VF for use of the crypto engines. Likewise, to have the best compression performance, an application should have access to at least one compression instance from a VF in the range 1-7 and at least one compression instance from a VF in the range 9-15.

   For more information on the configuration file options, see the appropriate Programmer's Guide (refer to Table 1).

2. **Can I get Host CPU in the Guest? How?**

   Yes. If you want to run Intel® QuickAssist Technology Software and Intel® DPDK in the guest you will need Host CPU. By default, the hypervisor provides emulated CPU to the guest. If you are using the Virt-Manager GUI, go to **Guest Machine details**, then **Processor > Configuration** and select **Copy host CPU**

configuration. If you are using the qemu command line, use the `-cpu host` parameter to provide host CPU to guest.