Lenovo

Introduction to Windows Server 2016 Hyper-V Discrete Device Assignment

Introduces the new PCIe Device Passthrough feature of Microsoft Windows Server 2016

Describes how to make PCIe devices available to guest operating systems in Hyper-V

Demonstrates how to make an NVIDIA GPU available in a virtual machine

Helps IT Specialists understand the new features of Windows Server 2016

Liu Ping

David Tanaka

LENOVO PRESS

Abstract

This paper describes the steps on how to enable Discrete Device Assignment (also known as PCI Passthrough) available as part of the Hyper-V role in Microsoft Windows Server 2016.

Discrete Device Assignment is a performance enhancement that allows a specific physical PCI device to be directly controlled by a guest VM running on the Hyper-V instance. Specifically, this new feature aims to deliver a certain type of PCI device class, such as Graphics Processing Units (GPU) or Non-Volatile Memory express (NVMe) devices, to a Windows Server 2016 virtual machine, where the VM will be given full and direct access to the physical PCIe device.

In this paper we describe how to enable and use this feature on Lenovo servers using Windows Server 2016 Technical Preview 4 (TP4). We provide the step-by-step instructions on how to make an NVIDIA GPU available to a Hyper-V virtual machine.

This paper is aimed at IT specialists and IT managers wanting to understand more about the new features of Windows Server 2016.

At Lenovo Press, we bring together experts to produce technical publications around topics of importance to you, providing information and best practices for using Lenovo products and solutions to solve IT challenges.

See a list of our most recent publications at the Lenovo Press web site:

http://lenovopress.com

Do you have the latest version? We update our papers from time to time, so check whether you have the latest version of this document by clicking the **Check for Updates** button on the front page of the PDF. Pressing this button will take you to a web page that will tell you if you are reading the latest version of the document and give you a link to the latest if needed. While you're there, you can also sign up to get notified via email whenever we make an update.

Contents

Introduction

Discrete Device Assignment (DDA, also known as PCI Passthrough) is a performance enhancement in Microsoft Windows Server 2016 and Hyper-V. It allows a specific physical PCIe device installed on the host system to be directly and exclusively controlled by a guest virtual machine (VM).

In this paper we describe the steps of how to configure an NVIDIA Quadro 5000 GPU, which is installed in one of the PCIe slots in our server, and make it accessible and managable by a Hyper-V guest VM that is running.

In our lab tests we used Windows Server 2016 Technical Preview 4 (TP4, build 10586) on the following servers:

- Lenovo ThinkServer RD650
- Lenovo System x3560 M5

VT-d support is required: The main prerequsite for supporting Discrete Device Assignment is that Intel Virtualization Technology for Direct I/O (VT-d) be enabled in UEFI.

At the time of this writing, the two main types of devices that will be supported with Discrete Device Assignment are the following:

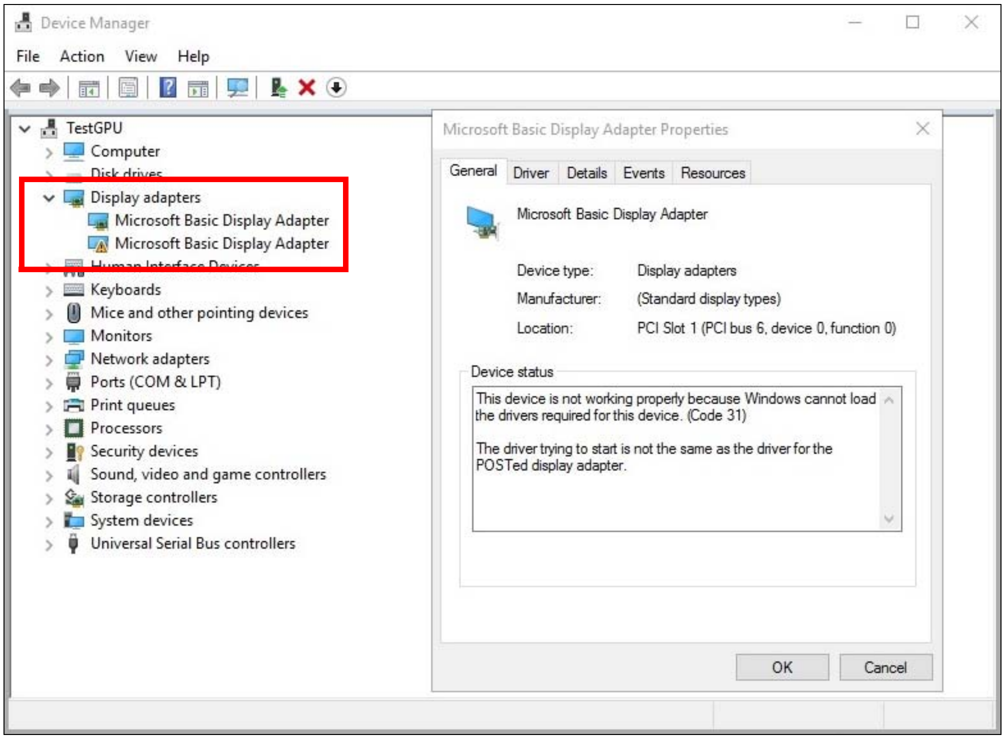
- GPUs and coprocessors
- NVMe (Non-Volatile Memory express) SSD controllers

In this paper, we provide the instructions on how to enable a Windows Server 2016 guest VM to utilize the Discrete Device Assignment functionality using a NVIDIA Quadro 5000 GPU running on the VM host. We also describe how to reverse the process and return the device to the host system.

Installing the GPU and creating a VM

The steps to get the GPU installed and the virtual machine operational are as follows:

1. Verify in UEFI that VT-d is enabled in the UEFI setup of the server. In some servers it is enabled by default.
2. Power off the server and install the NVIDIA Quadro 5000 GPU into a supported slot. Note that we are using the GPU for computational workloads, *not* as the video console for the server.
3. Install Windows Server 2016 (full product including the GUI). We installed Technical Preview 4 in our tests.
4. Login as an administrator and launch Device Manager. Click Display Adapters. Two display adapters will be listed, one of them being the NVIDIA Quadro 5000 GPU. Because the device driver for the GPU isn’t native to Windows Server 2016, the GPU device will appear as a yellow bang with a Code 28 device status, as shown in Figure 1 on page 4.



TestGPU
- Computer
- Disk drives
- Display adapters
  - Microsoft Basic Display Adapter
  - Microsoft Basic Display Adapter
- Human Interface Devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
- Print queues
- Processors
- Security devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

Microsoft Basic Display Adapter Properties
General Driver Details Events Resources

Microsoft Basic Display Adapter

Device type: Display adapters
Manufacturer: (Standard display types)
Location: PCI Slot 1 (PCI bus 6, device 0, function 0)

Device status

This device is not working properly because Windows cannot load the drivers required for this device. (Code 31)

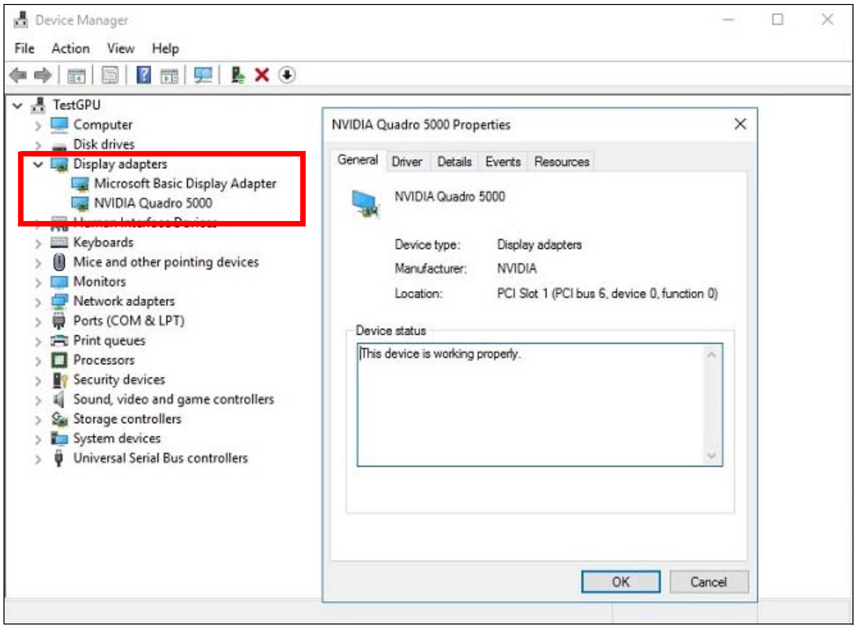
The driver trying to start is not the same as the driver for the POSTed display adapter.

Figure 1 Device Manager before the GPU device driver is installed on the host system

5. Install the appropriate device driver.

   At the time of this writing, the Windows device driver for this specific graphics card wasn't included with Windows Server 2016 TP4. NVIDIA provided us with a signed Windows 10 driver package that we were able to install under Windows Server 2016.

6. The device now appears in Windows Device Manager as shown in Figure 2.


Device Manager
File Action View Help

TestGPU
- Computer
- Disk drives
- Display adapters
  - Microsoft Basic Display Adapter
  - NVIDIA Quadro 5000
- Human Interface Devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
- Print queues
- Processors
- Security devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

NVIDIA Quadro 5000 Properties
General Driver Details Events Resources

NVIDIA Quadro 5000

Device type: Display adapters
Manufacturer: NVIDIA
Location: PCI Slot 1 (PCI bus 6, device 0, function 0)

Device status

This device is working properly.

Figure 2 Device Manager after the GPU device driver is installed on the host system

7. Install the Hyper-V role. Reboot the server when prompted.
8. Create a Hyper-V virtual machine (generation 2) and install Windows Server 2016 TP4 as the guest operating system.
9. Start the virtual machine, log in, and open Device Manager in the VM. Under Display Adapters, there will be a single display adapter, as shown in Figure 3.

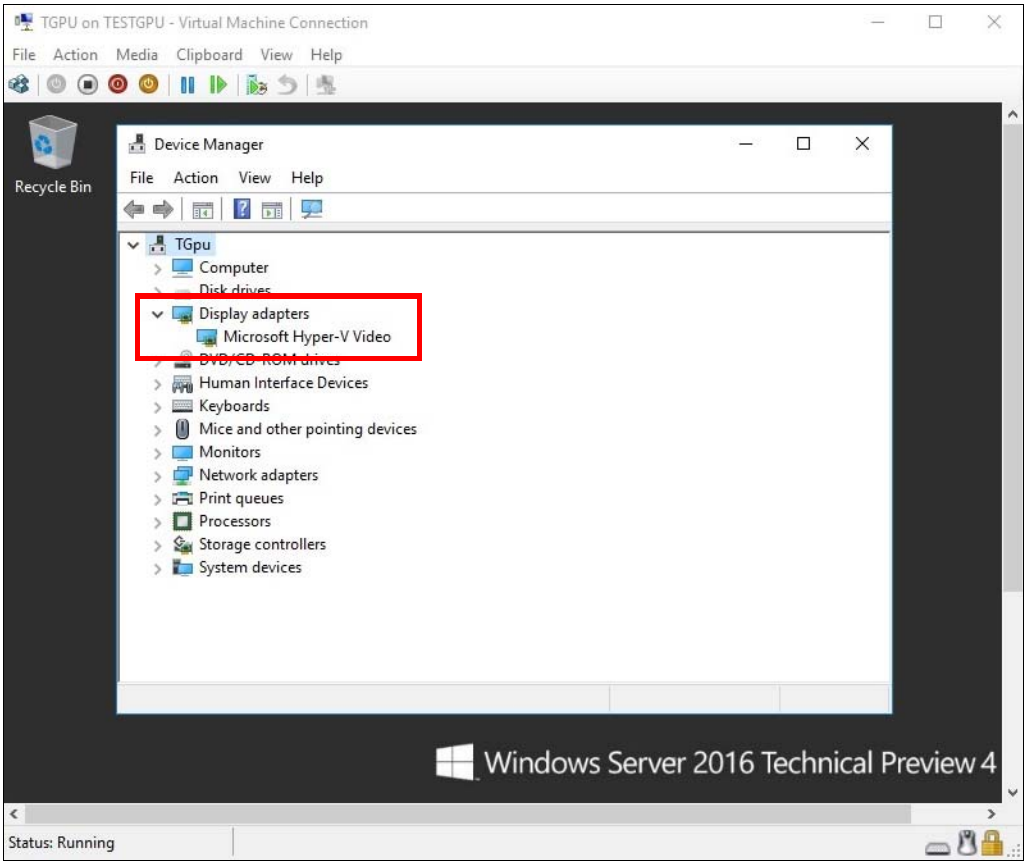


Figure 3 Device Manager inside the VM showing the display adapters

Enabling the device inside the VM

Now that the VM is operational, the next steps are to enable the GPU to be made available to the VM. Before the physical device is allowed to be passed through to the VM, the device must be disabled on the host system. The physical device must be accessible/available exclusively to the VM only.

The steps to enable Discrete Device Assignment to make the GPU available exclusively to the virtual machine are as follows:

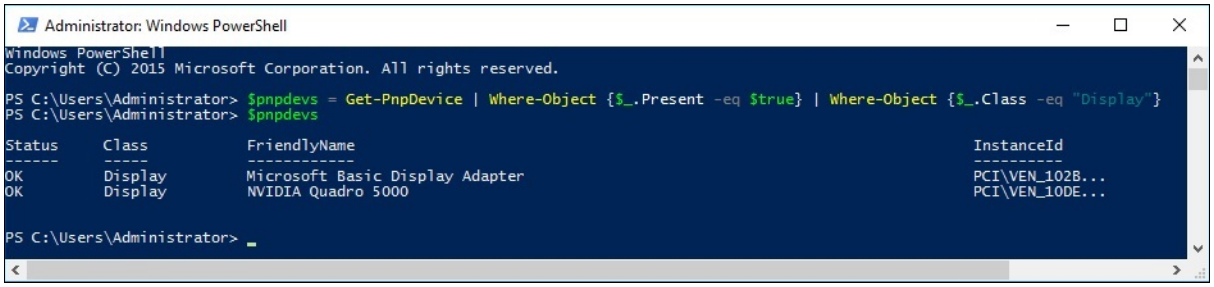
1. On the host system, open Windows PowerShell as an administrator on the host system.

2. Use the **Get-PnpDevice** command with a search condition to narrow down the PnpDdevice class you want to search for (in our example, to narrow down to just the “Display” class). The PowerShell commands are listed in Example 1.

Example 1 Get-PnpDevice command

```
$pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object
{$_.Class -eq "Display"}

$pnpdevs
```

The result is shown in Figure 4.



```
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> $pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object {$_.Class -eq "Display"}
PS C:\Users\Administrator> $pnpdevs

Status     Class           FriendlyName                                                     InstanceId
------     -----           ------------                                                     ----------
OK         Display         Microsoft Basic Display Adapter                                  PCI\VEN_102B...
OK         Display         NVIDIA Quadro 5000                                               PCI\VEN_10DE...


PS C:\Users\Administrator> _
```

Figure 4 Get-PnpDevice command to display active Display devices

> **Tip:** Use class `SCSIAdapter` to find NVMe storage adapter devices.

3. Disable the GPU graphics device on the host system, using the **Disable-PnpDevice** command in the PowerShell command window shown in Example 2. It is assumed you have already run the commands in Example 1 in the same PowerShell window.

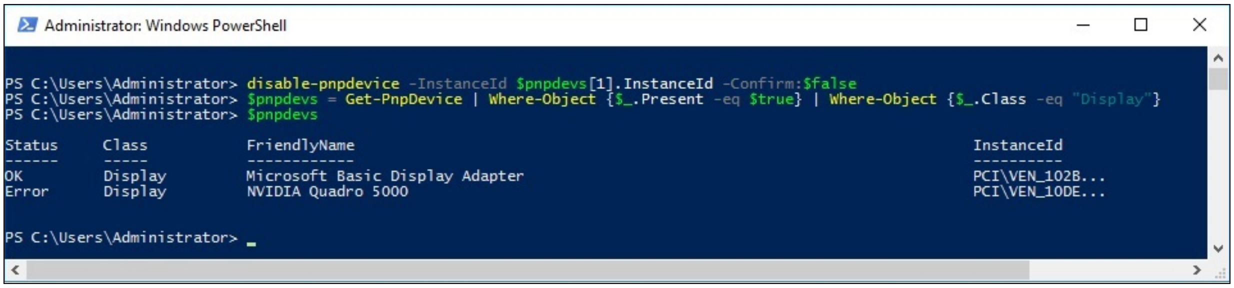
Example 2 Disable-PnpDevice command

```
Disable-PnpDevice -InstanceId $pnpdevs[1].InstanceId -Confirm:$false

$pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object
{$_.Class -eq "Display"}

$pnpdevs
```

In the example, the array index into the `$pnpdevs` variable will be 1, because the NVIDIA GPU device is the second entry in the list (the array is zero-based, where the first index is position 0).

The output of the commands is shown in Figure 5 on page 7.

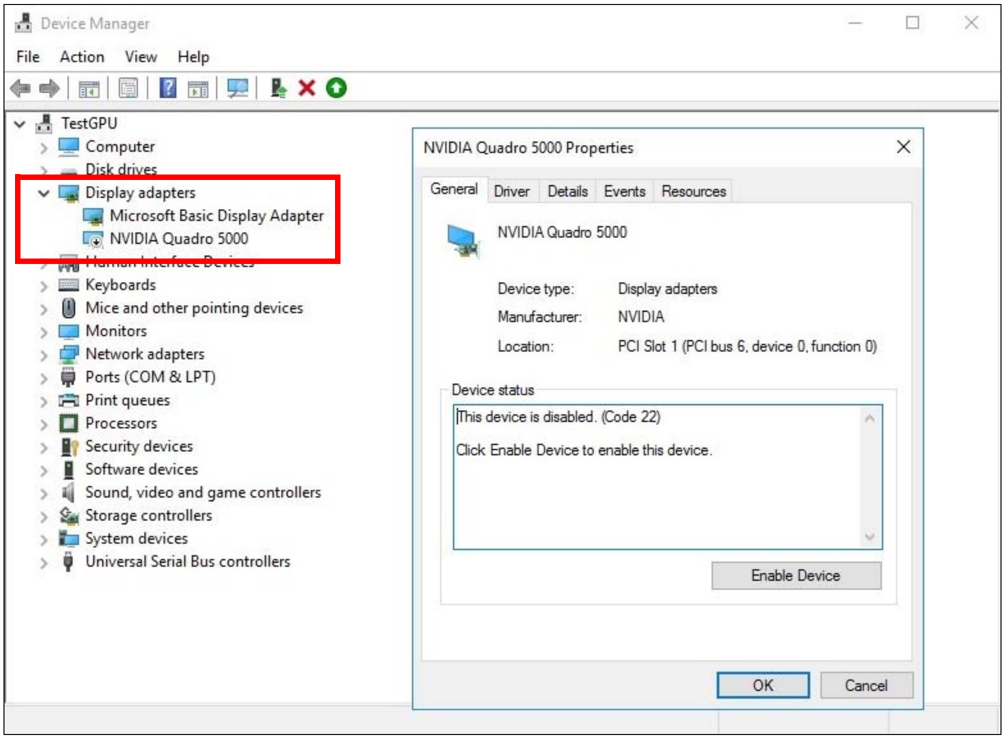
We ran Get-PnpDevice a second time to verify the status change. You can see that the GPU now displays a status of `Error`. This status is expected.



```
PS C:\Users\Administrator> disable-pnpdevice -InstanceId $pnpdevs[1].InstanceId -Confirm:$false
PS C:\Users\Administrator> $pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object {$_.Class -eq "Display"}
PS C:\Users\Administrator> $pnpdevs

Status     Class           FriendlyName                                                            InstanceId
------     -----           ------------                                                            ----------
OK         Display         Microsoft Basic Display Adapter                                         PCI\VEN_102B...
Error      Display         NVIDIA Quadro 5000                                                      PCI\VEN_10DE...

PS C:\Users\Administrator> _
```

Figure 5   Disable-PnpDevice command

4. Check Device Manager from the host system again to confirm that the NVIDIA GPU is disabled on the host system. See Figure 6.



NVIDIA Quadro 5000

Device type: Display adapters

Manufacturer: NVIDIA

Location: PCI Slot 1 (PCI bus 6, device 0, function 0)

Device status

This device is disabled. (Code 22)

Click Enable Device to enable this device.

Figure 6   Device Manager on the host system indicating that the GPU is now disabled

5. Dismount the device from the host system by first obtaining the PCI location of the physical device. The **Get-PnpDeviceProperty** command retrieves the PCI location path of the pass-through device. The **Dismount-VmHostAssignableDevice** command will then dismount the physical device so that it's no longer accessible on the Parent Partition.

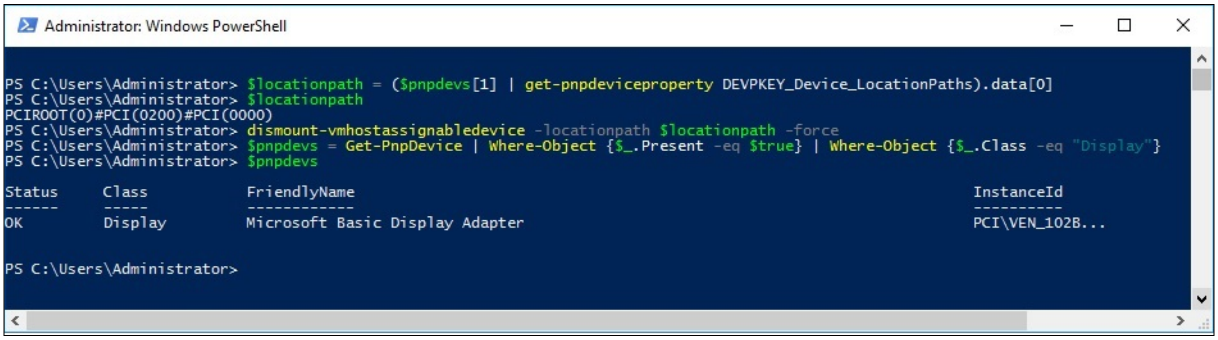
   We ran `Get-PnpDevice` again to verify that the change was successful. The commands are shown in Example 3.

Example 3   Get-PnpDeviceProperty and Dismount-VmHostAssignableDevice

```
$locationpath = ($pnpdevs[1] | Get-PnpDeviceProperty
DEVPKEY_Device_LocationPaths).data[0]

$locationpath

Dismount-VmHostAssignableDevice -locationpath $locationpath -force
```

```
$pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object
{$_.Class -eq "Display"}

$pnpdevs
```

The output of the commands is shown in Figure 7.



```
PS C:\Users\Administrator> $locationpath = ($pnpdevs[1] | get-pnpdeviceproperty DEVPKEY_Device_LocationPaths).data[0]
PS C:\Users\Administrator> $locationpath
PCIROOT(0)#PCI(0200)#PCI(0000)
PS C:\Users\Administrator> dismount-vmhostassignabledevice -locationpath $locationpath -force
PS C:\Users\Administrator> $pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object {$_.Class -eq "Display"}
PS C:\Users\Administrator> $pnpdevs

Status     Class           FriendlyName                                                            InstanceId
------     -----           ------------                                                            ----------
OK         Display         Microsoft Basic Display Adapter                                         PCI\VEN_102B...


PS C:\Users\Administrator>
```

Figure 7   Output of Get-PnpDeviceProperty and Dismount-VmHostAssignableDevice

6. After the dismount command is executed the NVIDIA GPU graphics device is no longer listed under the Display device class type. Open Device Manager on the host system again. The GPU is no longer listed under Display adapters (see Figure 8) but instead is listed under System devices as PCI Express Graphics Processing Unit - Dismounted (see Figure 9 on page 9)

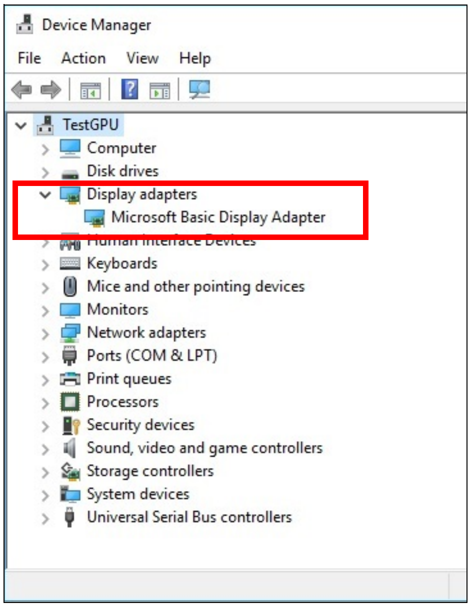
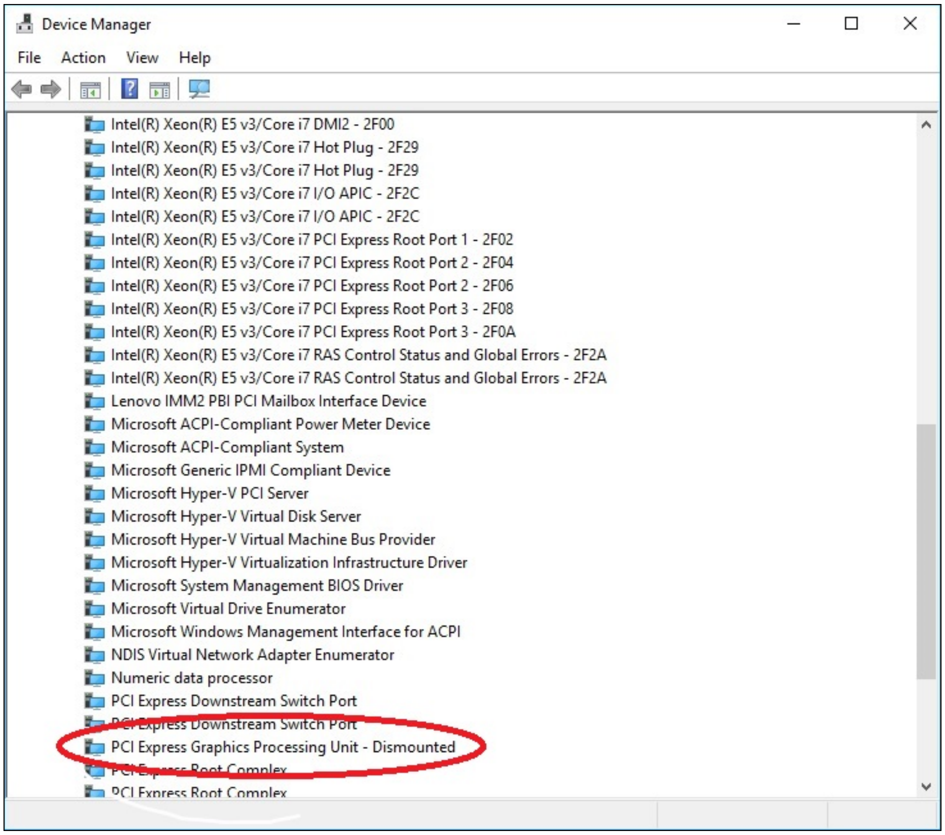


Figure 8   The GPU is no longer listed under Display adapters



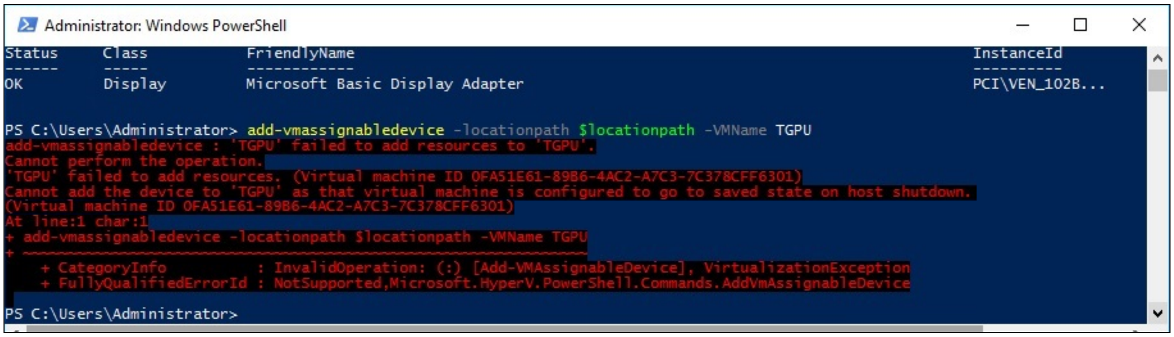
Intel(R) Xeon(R) E5 v3/Core i7 DMI2 - 2F00
Intel(R) Xeon(R) E5 v3/Core i7 Hot Plug - 2F29
Intel(R) Xeon(R) E5 v3/Core i7 Hot Plug - 2F29
Intel(R) Xeon(R) E5 v3/Core i7 I/O APIC - 2F2C
Intel(R) Xeon(R) E5 v3/Core i7 I/O APIC - 2F2C
Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 1 - 2F02
Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 2 - 2F04
Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 2 - 2F06
Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 3 - 2F08
Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 3 - 2F0A
Intel(R) Xeon(R) E5 v3/Core i7 RAS Control Status and Global Errors - 2F2A
Intel(R) Xeon(R) E5 v3/Core i7 RAS Control Status and Global Errors - 2F2A
Lenovo IMM2 PBI PCI Mailbox Interface Device
Microsoft ACPI-Compliant Power Meter Device
Microsoft ACPI-Compliant System
Microsoft Generic IPMI Compliant Device
Microsoft Hyper-V PCI Server
Microsoft Hyper-V Virtual Disk Server
Microsoft Hyper-V Virtual Machine Bus Provider
Microsoft Hyper-V Virtualization Infrastructure Driver
Microsoft System Management BIOS Driver
Microsoft Virtual Drive Enumerator
Microsoft Windows Management Interface for ACPI
NDIS Virtual Network Adapter Enumerator
Numeric data processor
PCI Express Downstream Switch Port
PCI Express Downstream Switch Port
PCI Express Graphics Processing Unit - Dismounted
PCI Express Root Complex
PCI Express Root Complex

Figure 9 The GPU now appears as ***dismounted*** *under System devices in Device Manager*

> **Tip:** Remember, even though the device is dismounted on the host, the device is still enabled and therefore the device's I/O resources will remain allocated to the physical device on the host system.

7. Change the automatic stop action of the host to turn off the VM.

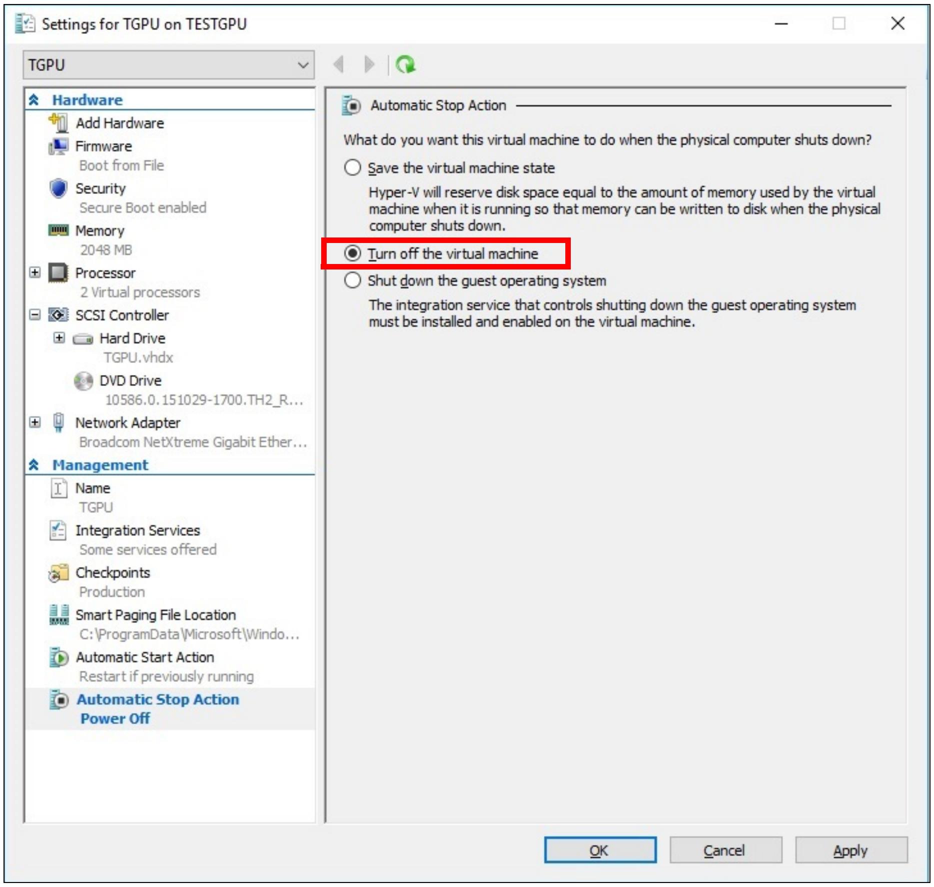
   By default, when the host server is shut down, the state of running virtual machines is saved. However, this setting prevents Discrete Device Assignment from being enabled. If you attempt to enable DDA, you will get the error shown in Figure 10.



```
Status     Class         FriendlyName                                              InstanceId
------     -----         ------------                                              ----------
OK         Display       Microsoft Basic Display Adapter                           PCI\VEN_102B...

PS C:\Users\Administrator> add-vmassignabledevice -locationpath $locationpath -VMName TGPU
add-vmassignabledevice : 'TGPU' failed to add resources to 'TGPU'.
Cannot perform the operation.
'TGPU' failed to add resources. (Virtual machine ID 0FA51E61-89B6-4AC2-A7C3-7C378CFF6301)
Cannot add the device to 'TGPU' as that virtual machine is configured to go to saved state on host shutdown.
(Virtual machine ID 0FA51E61-89B6-4AC2-A7C3-7C378CFF6301)
At line:1 char:1
+ add-vmassignabledevice -locationpath $locationpath -VMName TGPU
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [Add-VMAssignableDevice], VirtualizationException
    + FullyQualifiedErrorId : NotSupported,Microsoft.HyperV.PowerShell.Commands.AddVmAssignableDevice

PS C:\Users\Administrator>
```

Figure 10 Error attempting to run Add-VMAssignableDevice

In the properties for the VM, go to Automatic Stop Action and change the setting to **Turn off virtual machine** as shown in Figure 11 on page 10.



Settings for TGPU on TESTGPU

TGPU

Hardware
- Add Hardware
- Firmware
  Boot from File
- Security
  Secure Boot enabled
- Memory
  2048 MB
- Processor
  2 Virtual processors
- SCSI Controller
  - Hard Drive
    TGPU.vhdx
  - DVD Drive
    10586.0.151029-1700.TH2_R...
- Network Adapter
  Broadcom NetXtreme Gigabit Ether...

Management
- Name
  TGPU
- Integration Services
  Some services offered
- Checkpoints
  Production
- Smart Paging File Location
  C:\ProgramData\Microsoft\Windo...
- Automatic Start Action
  Restart if previously running
- **Automatic Stop Action**
  Power Off

Automatic Stop Action

What do you want this virtual machine to do when the physical computer shuts down?

○ Save the virtual machine state

Hyper-V will reserve disk space equal to the amount of memory used by the virtual machine when it is running so that memory can be written to disk when the physical computer shuts down.

● Turn off the virtual machine

○ Shut down the guest operating system

The integration service that controls shutting down the guest operating system must be installed and enabled on the virtual machine.

Figure 11 Setting the Automatic Stop Action for the VM

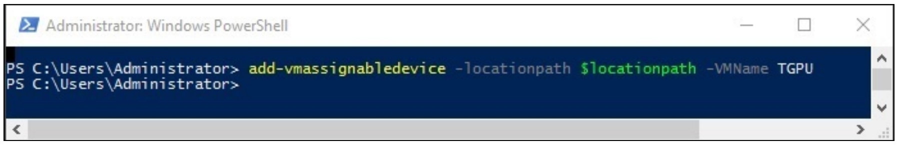
8. Issue the **Add-VMAssignableDevice** command on the host system to enable Discrete Device Assignment as shown in Example 4, below.

   The variable `$locationpath` comes from the commands we ran in Example 3 on page 7. `TGPU` is the name of the virtual machine in our lab environment.

Example 4 Add-VMAssignableDevice

```
Add-VMAssignableDevice -locationpath $locationpath -VMname TGPU
```

The output of the command, when successful, is shown in Figure 12.



Administrator: Windows PowerShell

```
PS C:\Users\Administrator> add-vmassignabledevice -locationpath $locationpath -VMName TGPU
PS C:\Users\Administrator>
```

Figure 12 Add-VMAssignableDevice

9. The GPU is now available and accessible exclusively to the VM. Open Device Manager in the VM. The new device is listed under Display adapters, as shown in Figure 13 on page 11.

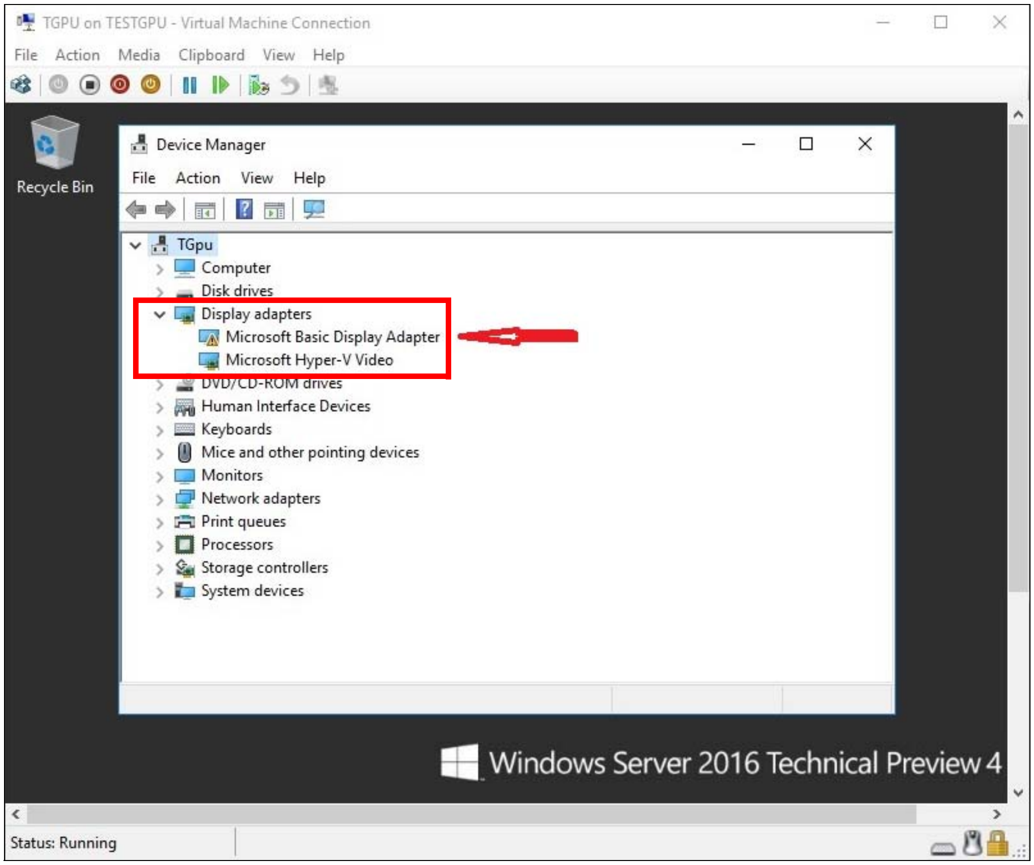
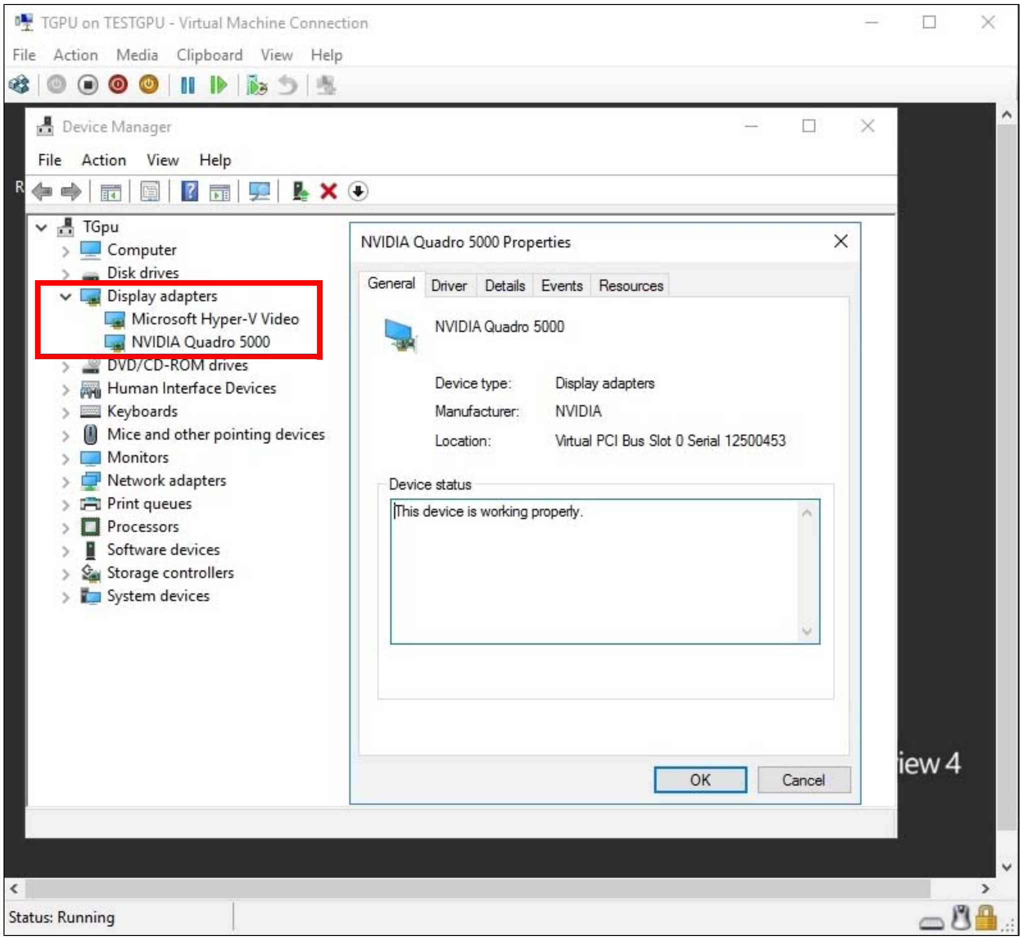


Figure 13   Device Manager in the VM showing the GPU is now accessible

10. Install the device driver for the GPU using the same driver we used in step 5 on page 4. The GPU will then be properly recognized by Device Manager in the VM, as shown in Figure 14 on page 12.



NVIDIA Quadro 5000

Device type: Display adapters

Manufacturer: NVIDIA

Location: Virtual PCI Bus Slot 0 Serial 12500453

Device status

This device is working properly.

Status: Running

Figure 14 The GPU properly recognized in Device Manager in the VM

Restoring the device to the host system

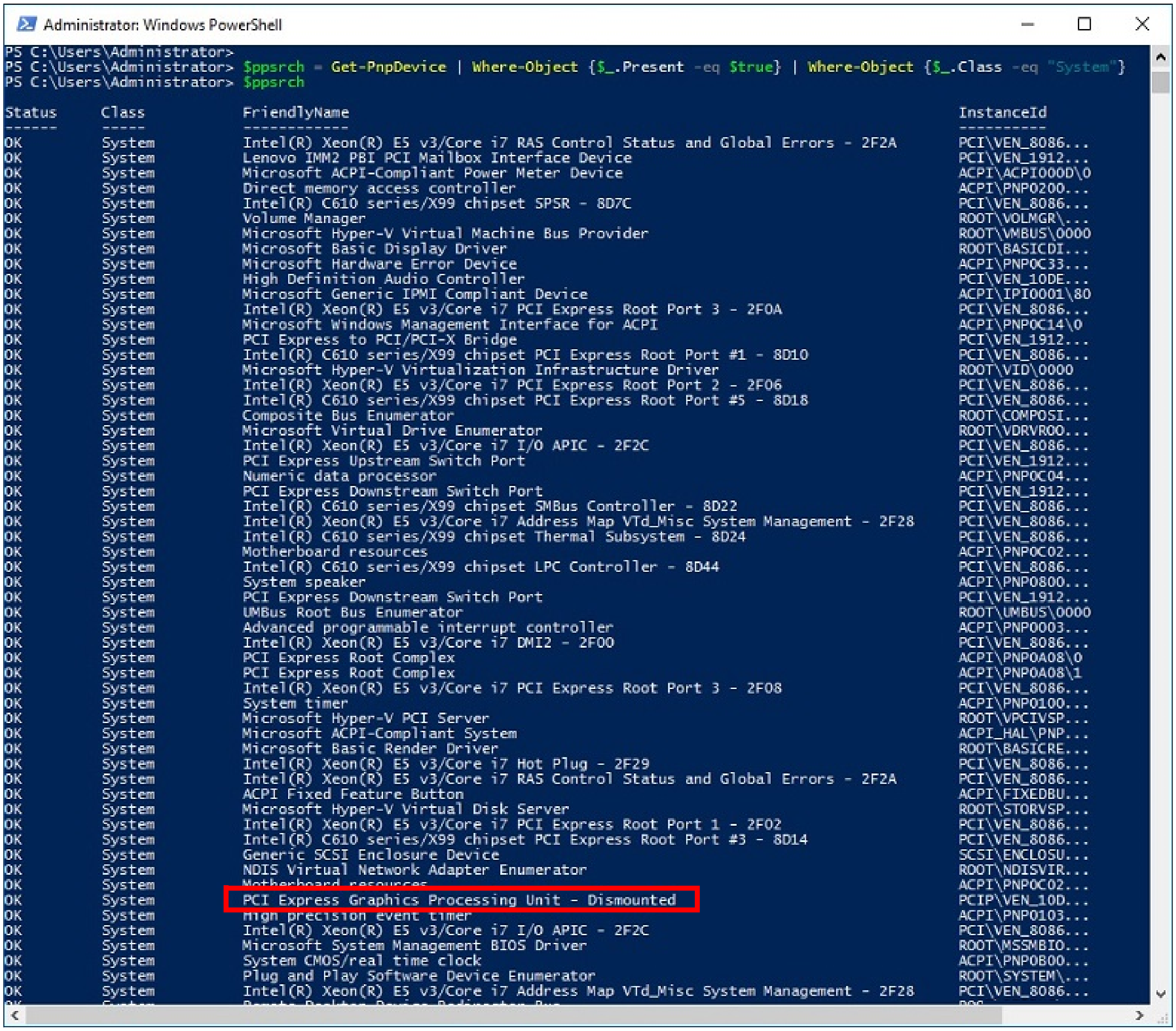
If the GPU is no longer needed by the virtual machine, you can restore the functionality of the device to the host system. The following steps describe the process.

1. Shut down the VM guest OS that s currently using the NVIDIA GPU graphics adapter.
2. Open PowerShell as Administrator on the host system.
3. Find the device’s location path InstanceId from the host system using the **Get-PnpDevice** command. The device is dismounted on the host system and is categorized as System class (as we confirmed in Figure 9 on page 9), so the Get-PnpDevice command will filter on class System, as shown in Example 5.

Example 5 Get-PnpDevice command to find all System class devices

```
$ppsrch = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object
{$_.Class -eq "System"}

$ppsrch
```

The output of the command is shown in Figure 15 on page 13. The GPU is highlighted in red.



```
PS C:\Users\Administrator>
PS C:\Users\Administrator> $ppsrch = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object {$_.Class -eq "System"}
PS C:\Users\Administrator> $ppsrch

Status     Class           FriendlyName                                                                     InstanceId
------     -----           ------------                                                                     ----------
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 RAS Control Status and Global Errors - 2F2A       PCI\VEN_8086...
OK         System          Lenovo IMM2 PBI PCI Mailbox Interface Device                                     PCI\VEN_1912...
OK         System          Microsoft ACPI-Compliant Power Meter Device                                      ACPI\ACPI000D\0
OK         System          Direct memory access controller                                                  ACPI\PNP0200...
OK         System          Intel(R) C610 series/X99 chipset SPSR - 8D7C                                     PCI\VEN_8086...
OK         System          Volume Manager                                                                   ROOT\VOLMGR\...
OK         System          Microsoft Hyper-V Virtual Machine Bus Provider                                   ROOT\VMBUS\0000
OK         System          Microsoft Basic Display Driver                                                   ROOT\BASICDI...
OK         System          Microsoft Hardware Error Device                                                  ACPI\PNP0C33...
OK         System          High Definition Audio Controller                                                 PCI\VEN_10DE...
OK         System          Microsoft Generic IPMI Compliant Device                                          ACPI\IPI0001\80
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 3 - 2F0A                    PCI\VEN_8086...
OK         System          Microsoft Windows Management Interface for ACPI                                  ACPI\PNP0C14\0
OK         System          PCI Express to PCI/PCI-X Bridge                                                  PCI\VEN_1912...
OK         System          Intel(R) C610 series/X99 chipset PCI Express Root Port #1 - 8D10                 PCI\VEN_8086...
OK         System          Microsoft Hyper-V Virtualization Infrastructure Driver                           ROOT\VID\0000
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 2 - 2F06                    PCI\VEN_8086...
OK         System          Intel(R) C610 series/X99 chipset PCI Express Root Port #5 - 8D18                 PCI\VEN_8086...
OK         System          Composite Bus Enumerator                                                         ROOT\COMPOSI...
OK         System          Microsoft Virtual Drive Enumerator                                               ROOT\VDRVROO...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 I/O APIC - 2F2C                                   PCI\VEN_8086...
OK         System          PCI Express Upstream Switch Port                                                 PCI\VEN_1912...
OK         System          Numeric data processor                                                           ACPI\PNP0C04...
OK         System          PCI Express Downstream Switch Port                                               PCI\VEN_1912...
OK         System          Intel(R) C610 series/X99 chipset SMBus Controller - 8D22                         PCI\VEN_8086...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 Address Map VTd_Misc System Management - 2F28     PCI\VEN_8086...
OK         System          Intel(R) C610 series/X99 chipset Thermal Subsystem - 8D24                        PCI\VEN_8086...
OK         System          Motherboard resources                                                            ACPI\PNP0C02...
OK         System          Intel(R) C610 series/X99 chipset LPC Controller - 8D44                           PCI\VEN_8086...
OK         System          System speaker                                                                   ACPI\PNP0800...
OK         System          PCI Express Downstream Switch Port                                               PCI\VEN_1912...
OK         System          UMBus Root Bus Enumerator                                                        ROOT\UMBUS\0000
OK         System          Advanced programmable interrupt controller                                       ACPI\PNP0003...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 DMI2 - 2F00                                       PCI\VEN_8086...
OK         System          PCI Express Root Complex                                                         ACPI\PNP0A08\0
OK         System          PCI Express Root Complex                                                         ACPI\PNP0A08\1
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 3 - 2F08                    PCI\VEN_8086...
OK         System          System timer                                                                     ACPI\PNP0100...
OK         System          Microsoft Hyper-V PCI Server                                                     ROOT\VPCIVSP...
OK         System          Microsoft ACPI-Compliant System                                                  ACPI_HAL\PNP...
OK         System          Microsoft Basic Render Driver                                                    ROOT\BASICRE...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 Hot Plug - 2F29                                   PCI\VEN_8086...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 RAS Control Status and Global Errors - 2F2A       PCI\VEN_8086...
OK         System          ACPI Fixed Feature Button                                                        ACPI\FIXEDBU...
OK         System          Microsoft Hyper-V Virtual Disk Server                                            ROOT\STORVSP...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 PCI Express Root Port 1 - 2F02                    PCI\VEN_8086...
OK         System          Intel(R) C610 series/X99 chipset PCI Express Root Port #3 - 8D14                 PCI\VEN_8086...
OK         System          Generic SCSI Enclosure Device                                                    SCSI\ENCLOSU...
OK         System          NDIS Virtual Network Adapter Enumerator                                          ROOT\NDISVIR...
OK         System          Motherboard resources                                                            ACPI\PNP0C02...
OK         System          PCI Express Graphics Processing Unit - Dismounted                                PCIP\VEN_10D...
OK         System          High precision event timer                                                       ACPI\PNP0103...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 I/O APIC - 2F2C                                   PCI\VEN_8086...
OK         System          Microsoft System Management BIOS Driver                                          ROOT\MSSMBIO...
OK         System          System CMOS/real time clock                                                      ACPI\PNP0B00...
OK         System          Plug and Play Software Device Enumerator                                         ROOT\SYSTEM\...
OK         System          Intel(R) Xeon(R) E5 v3/Core i7 Address Map VTd_Misc System Management - 2F28     PCI\VEN_8086...
```

Figure 15   Showing all devices of class System on the host system

Figure 15 shows the variable `$ppsrch` is an array containing various “location paths” InstanceId names. The index is zero-based (that is, the first item is counted as 0) and therefore the index for the dismounted GPU device is 50.

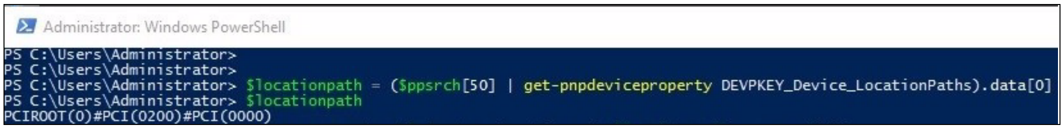
Note: You will have to manually determine the index for the device in the System class by counting the number of entries on the screen. Remember that the first entry is index 0.

4. Use the **Get-PnpDeviceProperty** command to obtain the path location for the device as shown in Example 6.

Example 6   Get-PnpDeviceProperty command

```
$locationpath = ($ppsrch[50] | Get-PnpDeviceProperty
DEVPKEY_Device_LocationPaths).data[0]

$locationpath
```

The output of the command is shown in Figure 16.



```
PS C:\Users\Administrator>
PS C:\Users\Administrator>
PS C:\Users\Administrator> $locationpath = ($ppsrch[50] | get-pnpdeviceproperty DEVPKEY_Device_LocationPaths).data[0]
PS C:\Users\Administrator> $locationpath
PCIROOT(0)#PCI(0200)#PCI(0000)
```

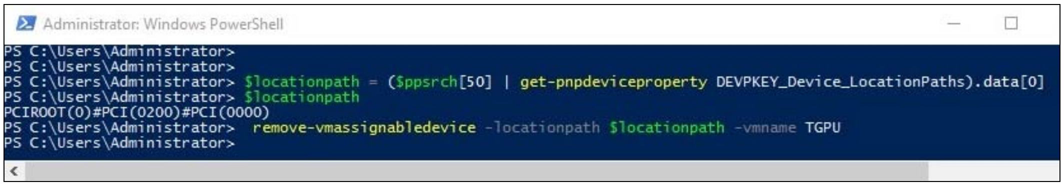
Figure 16   Get-PnpDeviceProperty output

5. Use the **Remove-VMAssignableDevice** command to remove the GPU based on its path location that we just assigned to variable $locationpath, using the following command.

Example 7   Remove-VMAssignableDevice command

```
Remove-VMAssignableDevice -location $locationpath -vmname TGPU
```

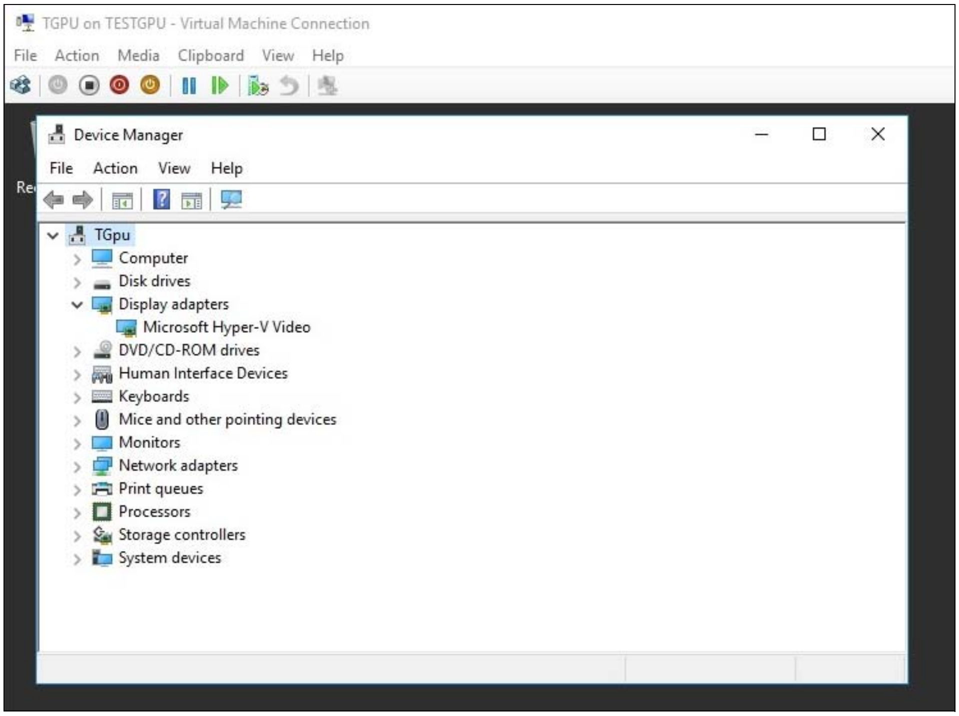
The output of the command is shown in Figure 17



```
PS C:\Users\Administrator>
PS C:\Users\Administrator>
PS C:\Users\Administrator> $locationpath = ($ppsrch[50] | get-pnpdeviceproperty DEVPKEY_Device_LocationPaths).data[0]
PS C:\Users\Administrator> $locationpath
PCIROOT(0)#PCI(0200)#PCI(0000)
PS C:\Users\Administrator>  remove-vmassignabledevice -locationpath $locationpath -vmname TGPU
PS C:\Users\Administrator>
```

Figure 17   Remove-VMAssignableDevice output

6. You can now confirm that the GPU has been removed from the VM by checking Device Manager in the VM, Figure 18


TGPU on TESTGPU - Virtual Machine Connection
File Action Media Clipboard View Help
Device Manager
File Action View Help

- TGpu
  - Computer
  - Disk drives
  - Display adapters
    - Microsoft Hyper-V Video
  - DVD/CD-ROM drives
  - Human Interface Devices
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Print queues
  - Processors
  - Storage controllers
  - System devices

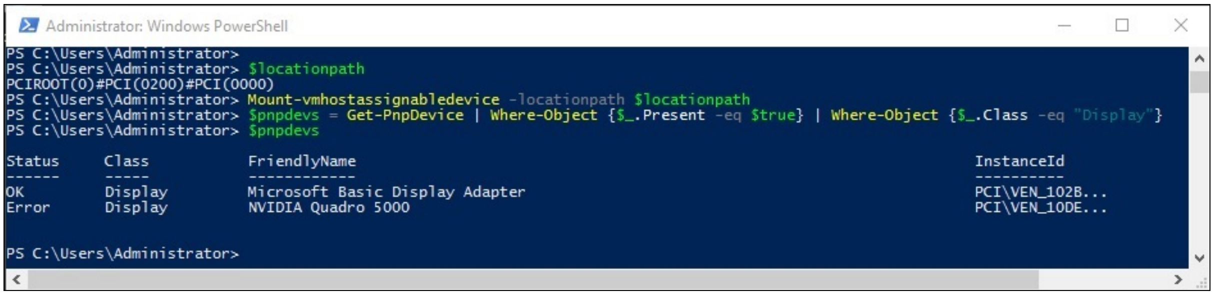
Figure 18   The GPU is now removed as a device from Device Manager in the VM

7. On the host, mount the device again, using the **Mount-VmHostAssignableDevice** command as shown in Example 8

Example 8 Mount-VmHostAssignableDevice command

```
Mount-VmHostAssignableDevice -locationpath $locationpath

$pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object
{$_.Class -eq "Display"}

$pnpdevs
```

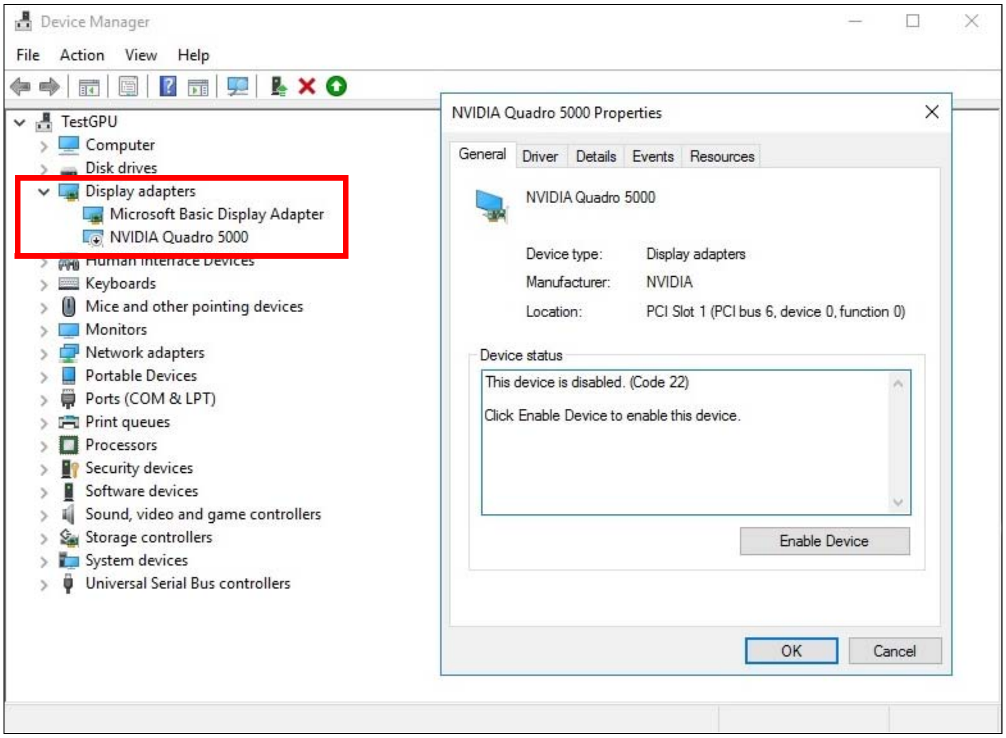
The output is shown in Figure 19.



```
PS C:\Users\Administrator>
PS C:\Users\Administrator> $locationpath
PCIROOT(0)#PCI(0200)#PCI(0000)
PS C:\Users\Administrator> Mount-vmhostassignabledevice -locationpath $locationpath
PS C:\Users\Administrator> $pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object {$_.Class -eq "Display"}
PS C:\Users\Administrator> $pnpdevs

Status     Class          FriendlyName                                                      InstanceId
------     -----          ------------                                                      ----------
OK         Display        Microsoft Basic Display Adapter                                   PCI\VEN_102B...
Error      Display        NVIDIA Quadro 5000                                                PCI\VEN_10DE...

PS C:\Users\Administrator>
```

Figure 19 Output of the Mount-VmHostAssignableDevice command

8. The mount can be verified by checking Device Manager on the host (Figure 20). Once again, the device is visible in the Display adapters section (albeit disabled at present).



NVIDIA Quadro 5000

Device type: Display adapters

Manufacturer: NVIDIA

Location: PCI Slot 1 (PCI bus 6, device 0, function 0)

Device status

This device is disabled. (Code 22)

Click Enable Device to enable this device.

Figure 20 Device Manager shows the GPU is mounted as a device on the host

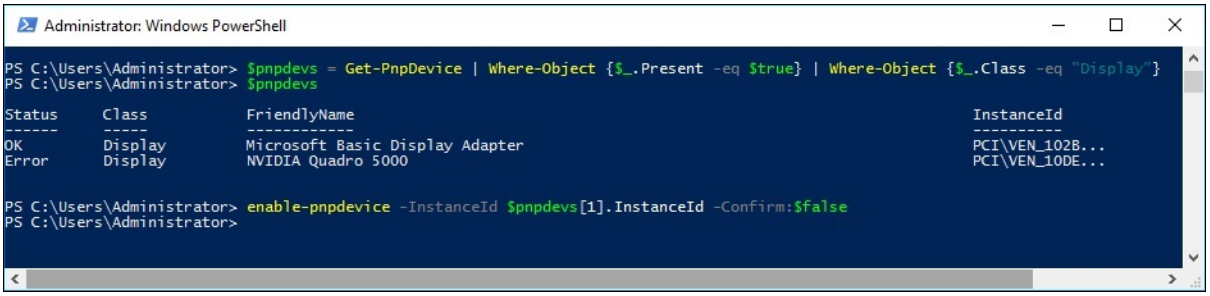
9. Enable the device using the **Enable-PnpDevice** command. Similar to the Disable-PnpDevice command, the array index into the $pnpdevs variable will be 1,

because the NVIDIA GPU device is the second entry in the list (a zero-based list). The command is as follows:

Example 9 Enable-PnpDevice command

```
$pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object
{$_.Class -eq "Display"}

$pnpdevs

Enable-PnpDevice -InstanceID $pnpdevs[1].InstanceID -Confirm:$false
```

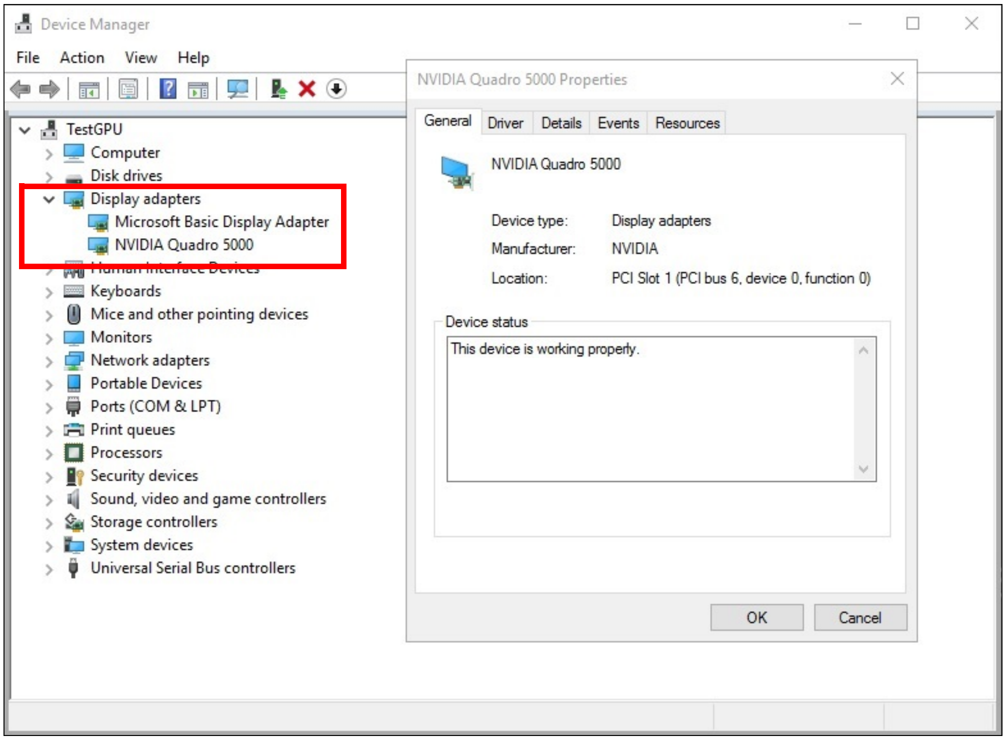
The output is shown in Figure 21.



```
PS C:\Users\Administrator> $pnpdevs = Get-PnpDevice | Where-Object {$_.Present -eq $true} | Where-Object {$_.Class -eq "Display"}
PS C:\Users\Administrator> $pnpdevs

Status     Class          FriendlyName                                                        InstanceId
------     -----          ------------                                                        ----------
OK         Display        Microsoft Basic Display Adapter                                     PCI\VEN_102B...
Error      Display        NVIDIA Quadro 5000                                                  PCI\VEN_10DE...

PS C:\Users\Administrator> enable-pnpdevice -InstanceId $pnpdevs[1].InstanceId -Confirm:$false
PS C:\Users\Administrator>
```

Figure 21 Output of the Enable-PnpDevice command

10. The device is now restored as mounted and enabled on the host system, as shown by the entry in Device Manager, in Figure 22. The device driver is automatically loaded.



NVIDIA Quadro 5000

Device type: Display adapters

Manufacturer: NVIDIA

Location: PCI Slot 1 (PCI bus 6, device 0, function 0)

Device status

This device is working properly.

Figure 22 GPU is mounted and activated and the device driver loaded

Summary

Discrete Device Assignment is one of the new features in Windows Server 2016. It makes it possible to make NVMe and GPU devices available for exclusive use by a virtual machine. VMs can own and use the physical device directly and users can get better performance: faster speed in NVMe file operations and more resources for computing and graphics processing with GPUs.

For more information about Discrete Device Assignment, see the following articles:

- Discrete Device Assignment — Description and background

  https://blogs.technet.microsoft.com/virtualization/2015/11/19/discrete-device-assignment-description-and-background/

- Discrete Device Assignment — Machines and devices

  https://blogs.technet.microsoft.com/virtualization/2015/11/20/discrete-device-assignment-machines-and-devices/

Authors

This paper was produced by the following team of specialists:

Liu Ping is a Windows Engineer with the Lenovo Enterprise Business Group in Beijing, China. She has eight years of experience with diagnostics development, seven years of experience with BIOS development, and two years of experience with Windows debugging.

David Tanaka is a UEFI BIOS developer and Windows Engineer with Lenovo, based in Kirkland, WA (USA). He has over nine years of experience with UEFI and BIOS development and over 18 years of experience with Windows Development/Engineering.

Thanks to the following people for their contributions to this project:

- Yang Yao, Lenovo Information Development
- David Watts, Lenovo Press
- Mark T. Chapman, Lenovo

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service.

Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
1009 Think Place - Building One
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary.

Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This document was created or updated on April 18, 2017.

Send us your comments via the **Rate & Provide Feedback** form found at
http://lenovopress.com/lp0088

Trademarks

Lenovo, the Lenovo logo, and For Those Who Do are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. These and other Lenovo trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by Lenovo at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of Lenovo trademarks is available on the Web at http://www.lenovo.com/legal/copytrade.html.

The following terms are trademarks of Lenovo in the United States, other countries, or both:

Lenovo®
Lenovo(logo)®
ThinkServer®

The following terms are trademarks of other companies:

Intel, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Hyper-V, Windows, Windows PowerShell, Windows Server, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.