Enabling SR-IOV

The *Holy Grail* of performance with virtualization is for virtual machines to perform as if they were running on stand-alone physical machines. For networking this goal means reducing CPU utilization, reducing latency and jitter, and achieving native I/O throughput as if the VM was talking directly to the physical network adapter.

Single Root I/O Virtualization (SR-IOV) in Windows Server 2012 realizes this goal by enabling virtual machines to perform I/O directly to the physical network adapter, bypassing the root partition. SR-IOV is ideal for high I/O workloads that do not require port policies, QoS, or network virtualization enforced at the end host virtual switch. In Windows Server 2012, SR-IOV can be deployed in conjunction with key capabilities such as Live Migration. In addition, SR-IOV can be deployed on existing servers, because it is compatible with most current-generation 10 Gbps NICS, and several SR-IOV drivers come inbox with the operating system.

So this all sounds very interesting, but how do you enable it? Well that would be the next question that I would be asking. Assuming that you have set up the BIOS correctly, your processors support Second Level Address Translation (SLAT), and you have a SR-IOV PCIe network card in the system; the first step to have any network connectivity (whether it is to enable SR-IOV or not) is to create an external virtual switch. You could do this by using the Virtual Switch Manager in Hyper-V Manager, or you could do this in PowerShell.

Let's start by looking at the Hyper-V Manager. You open Hyper-V Manager, then click on Virtual Switch Manager on the right hand side. This will open up the Virtual Switch Manager interface. From there it is much like creating any other virtual switch with one difference:

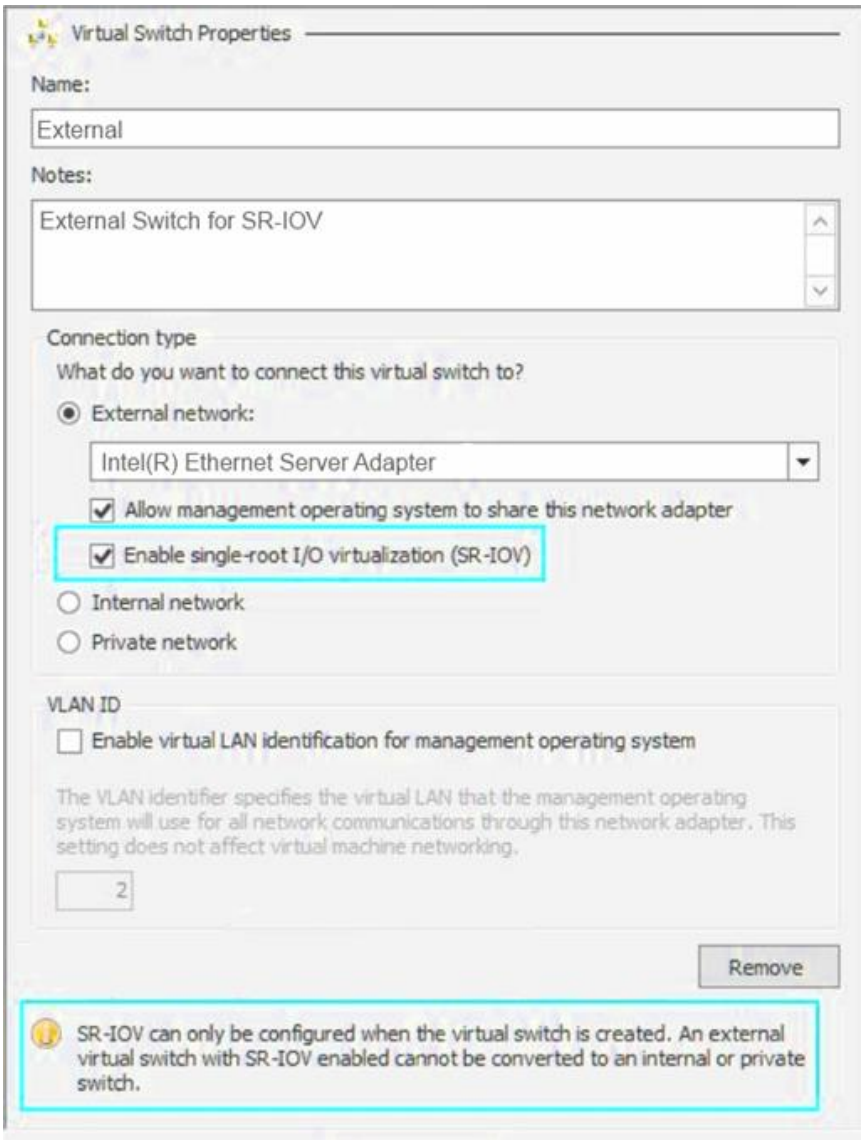

Figure 2: A screenshot showing how to enable SR-IOV for a virtual switch on the host.

Note:
At the bottom on the window you can see the SR-IOV warning. Once a switch is created you cannot add this option again. If you wish to add SR-IOV later you will have to delete the switch and recreate it.

As I suggested above you can also create the Virtual Switch in PowerShell. Using PowerShell's extensions for Hyper-V you run the command New-VMSwitch. This command does require a parameter to specify the physical network that you wish to use. And Microsoft thought of that as well and we have a command for that. You can run Get-NetAdapter to list them out.

So here you can see that I have listed out the network adapters in PS below:

```
PS C:\Users\administrator.KHNPIV> Get-NetAdapter

Name                      InterfaceDescription                    ifIndex Status       MacAddress             LinkSpeed
----                      --------------------                    ------- ------       ----------             ---------
Private                   Broadcom NetXtreme Gigabit Ethernet #2       14 Up           00-1C-23-E1-71-FA         1 Gbps
HyperV                    Broadcom BCM5709C NetXtreme II Gi...#29      15 Up           00-10-18-5B-05-1E         1 Gbps
Public                    Broadcom NetXtreme Gigabit Ethernet          13 Up           00-1C-23-E1-71-FB         1 Gbps
SR-IOV                    Broadcom BCM57712 NetXtreme...#26            12 Up           00-10-18-5B-05-1C         1 Gbps
```

Figure 3: A screenshot showing how to use Windows PowerShell to list the network adapters.

So now that we have the network adapters name we can use the New-VMSwitch command:

```
PS C:\Users\administrator.KHNPIV> New-VMSwitch SR-IOV -netadaptername SR-IOV -EnableIov $true

Name            SwitchType NetAdapterInterfaceDescription
----            ---------- ------------------------------
Sr-IOV          External   Broadcom BCM57712 NetXtreme...#26
```

Figure 4: A screenshot showing the New-VMSwitch cmdlet.

So now with the Get-VMSwitch command we can see the properties that were exposed on the VMNetworkadapter object:

```
PS C:\> get-vmswitch | fl *iov*


IovEnabled                : True
IovVirtualFunctionCount   : 32
IovVirtualFunctionsInUse  : 1
IovQueuePairCount         : 63
IovQueuePairsInUse        : 1
IovSupport                : True
IovSupportReasons         : {OK}
```

Figure 5: A screenshot showing output from running the Get-VMSwitch cmdlet.

I know…that's cool but what does it mean? Well this would be a great time to take a look at the output in greater details.

At the top we have IovEnabled. This is true only when the virtual switch is created in SR-IOV mode…and well false in any other configurations. The rest require a bit more explanation:

- **IovVirtualFunctionCount**: This is the number of VFs that are currently available for use by guest operating systems. Keep in mind this is a hardware setting on the physical network adapter and may vary by vendor. Also note that each software based NIC can be backed by a VF. Also keep in mind that each VM can have up to eight software based NIC's.
- **IovVirtualFunctionsInUse**: This is the current number of VFs in use by guest operating systems. In the screen shot above you see this as listed with 1. This is because I have one VM that is running one NIC in SR-IOV mode.
- **IovQueuePairCount**: This is the number of pairs that are available as hardware resources on the physical NIC. Again this may vary from hardware vendor to hardware vendor. In most cases there will be as many pairs available as there are VFs. Depending on the vendor additional functionality might be included in the VFs, for instance a hardware vendor my support RSS in a guest operating system that is backed by a VF and

more than one queued pair may be required for this. Again this is all based on hardware so any questions about this should be directed to the vendor.

- **IovQueuePairsInUse**: This is the number of queued pair that are currently assigned to VFs and assigned to a guest operating system.

For the last two, I will address these together as they are similar.

IovSupport and IovSupportReasons are numeric code and descriptions regarding the status of the physical network adapter. I will address this more in the troubleshooting section.

Enabling the guest OS

OK now we have the switch created, is that it? And the correct answer would be not yet. We now have to enable the guest operating system. So with that in mind we open up the settings for the guest and under the sub-node for the network adapter you will see hardware acceleration. You guessed it…that is where we would enable SR-IOV for the guest:

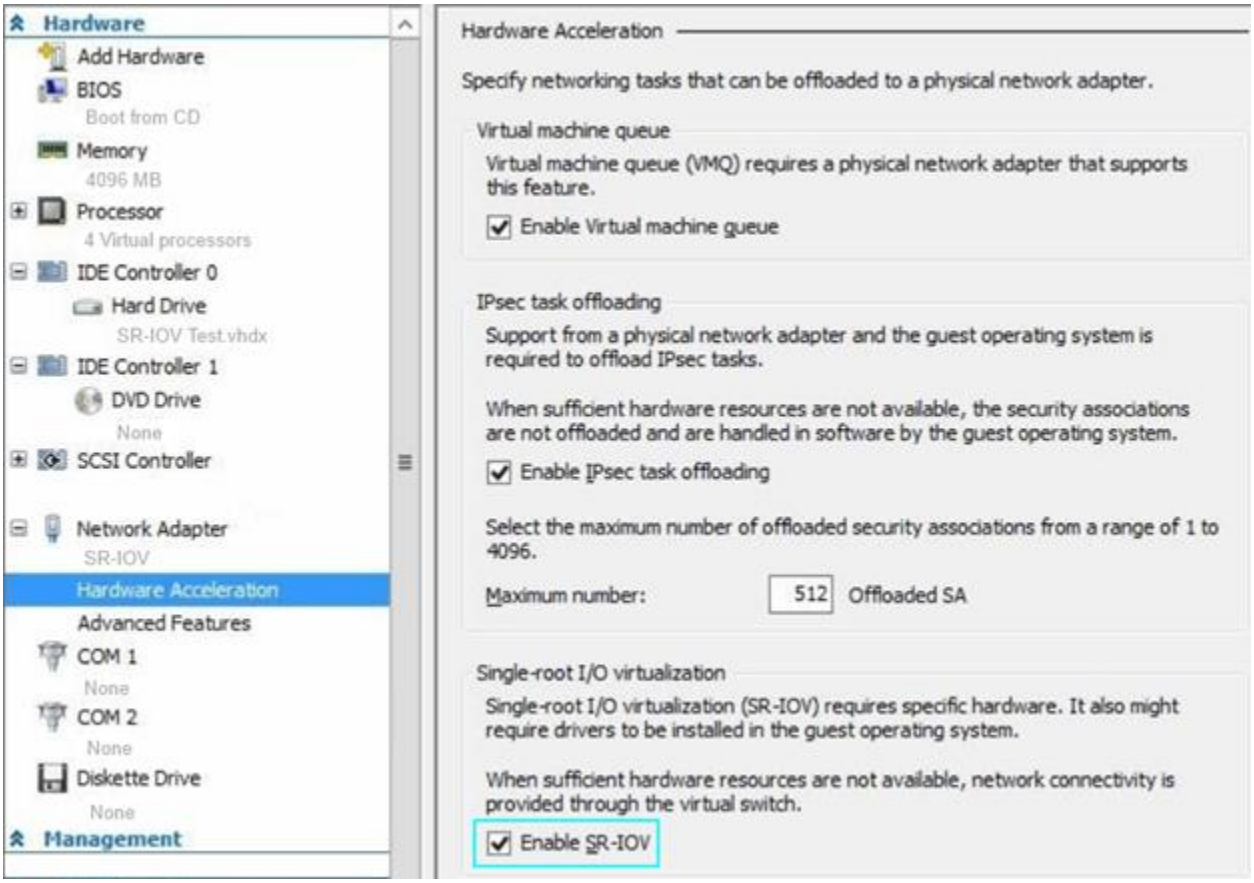

Figure 6: A screenshot showing how to enable SR-IOV for the guest.

Note:
Placing a check mark in the "Enable SR-IOV" box sets IovWeight setting to some number greater than 0.

And now let's take a look at PowerShell to see how we can set this in PowerShell.

So with the command Set-VMNetworkAdapter we can set this setting:

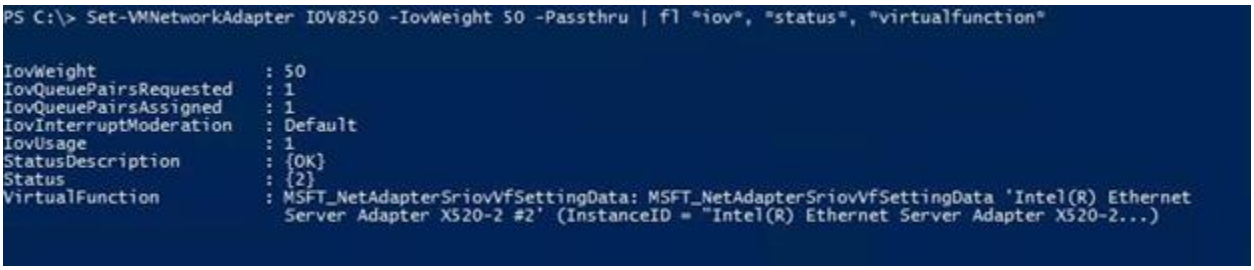

Figure 7: A screenshot showing the Set-VMNetworkAdapter cmdlet.

Here you can see IovWeight set to 50. This should be explained some, but the concept here is very simple. So you might be familiar with VMQWeight in Windows Server 2008 R2, and well the IovWeight functionality operates the same in Windows Server 2012. This setting expresses the desire for a hardware offload, but it's not a guarantee. So any number greater than 0 turn this setting on…so in short 1-100 is on, and 0 is off.