Configuring NVMe Over Fabrics (NVMeoF) with RoCEv2 in Linux

Guidelines for using NVMe over Fabrics (NVMeoF) with RoCEv2 on Linux

General Guidelines and Limitations

- Cisco recommends that you check UCS Hardware and Software Compatibility specific to your Cisco IMC release to determine support for NVMeoF. NVMeoF is supported on Cisco UCS C-Series M5 and later servers.
- NVMeoF with RoCEv2 is supported only with the Cisco UCS VIC 14xx series adapters. NVMeoF is not supported on Cisco UCS VIC 12xx or 13xx series adapters.
- When creating RoCEv2 interfaces, use Cisco IMC provided Linux-NVMe-RoCE adapter policy.
- Only two RoCEv2 enabled vNICs per adapter are supported.
- Booting from an NVMeoF namespace is not supported.
- Layer 3 routing is not supported.
- RoCEv2 does not support bonding.
- Saving a crashdump to an NVMeoF namespace during a system crash is not supported.

- NVMeoF cannot be used with usNIC, VMFEX, VxLAN, VMQ, VMMQ, NVGRE, Geneve offload and DPDK features.
- Netflow monitoring is not supported on RoCEv2 interfaces.
- In the Linux-NVMe-RoCE policy, do not change values of Queue Pairs, Memory Regions, Resource Groups, and Priority settings other than to Cisco provided default values. NVMeoF functionality may not be guaranteed with different settings for Queue Pairs, Memory Regions, Resource Groups, and Priority.
- The QoSno drop class configuration must be properly configured on upstream switches such as Cisco Nexus 9000 series switches. QoS configurations vary between different upstream switches.
- Set MTU size correctly on the VLANs and QoS policy on upstream switches.
- Spanning Tree Protocol (STP) may cause temporary loss of network connectivity when a failover or failback event occurs. To prevent this issue from occurring, disable STP on uplink switches.

Interrupts

- Linux RoCEv2 interface supports only MSIx interrupt mode. Cisco recommends that you avoid changing interrupt mode when the interface is configured with RoCEv2 properties.
- The minimum interrupt count for using RoCEv2 with Linux is 8.

Downgrade Limitations

Cisco recommends that you remove the RoCEv2 configuration before downgrading to any non-supported RoCEv2 release.

Linux Requirements

Configuration and use of RoCEv2 in Linux requires the following:

- Red Hat Enterprise Linux:
  - Red Hat Enterprise Linux 7.6 with Z-Kernel 3.10.0-957.27.2
  - Redhat Enterprise Linux 7.7 with Linux Z-kernel-3.10.0-1062.9.1 and above
  - Redhat Enterprise Linux 7.8, 7.9, and 8.2

Note Cisco IMC Release 4.2(2x) or later supports Redhat Enterprise Linux 7.8, 7.9, 8.2, 8.4, and 8.5

- InfiniBand kernel API module ib_core
- Cisco IMC Release 4.1(1x) or later
- Cisco IMC Release 4.2(2x) or later
- VIC firmware - Minimum requirement is 5.1(1x) for IPv4 support and 5.1(2x) for IPv6 support

- UCS C-Series M5 servers with Cisco UCS VIC 14xx series and 15xxx series adapters
- eNIC driver version 4.0.0.6-802-21 or later provided with the 4.1(1x) release package
- enic_rdma driver version 1.0.0.6-802-21 or later provided with the 4.1(1x) release package

Note Use eNIC driver version 4.0.0.10-802.34 or later and enic_rdma driver version 1.0.0.10-802.34 or later for IPv6 support.

- A storage array that supports NVMeoF connection

Configuring RoCEv2 for NVMeoF using Cisco IMC GUI

Step 1 In the **Navigation** pane, click **Networking**.

Step 2 Expand **Networking** and click on the adapter to configure RoCEv2 vNIC.

Step 3 Select the **vNICs** tab.

Step 4 Perform one the following:

- Click **Add vNIC** to create a new vNIC and modify the properties as mentioned in next step.

  OR

- From the left pane, select an existing vNIC and modify the properties as mentioned in next step.

Step 5 Expand RoCE Properties.

Step 6 Select RoCE checkbox.

Step 7 Modify the following vNIC properties:

Property	Field	Value
Ethernet Interrupt	**Interrupt count** field	256
Ethernet Receive Queue	**Count** field	1
	Ring Size field	512
Ethernet Transmit Queue	**Count** field	1
	Ring Size field	256
Completion Queue	**Count** field	2
RoCE Properties	**Queue Pairs** field	1024
	Memory Regions field	131072
	Resource Groups field	8
	Class of Service drop-down list	5

Step 8 Click **Save Changes**.

Step 9 Select **Reboot** when prompted.

Enabling an SRIOV BIOS Policy

Use these steps to configure the server with RoCEv2 vNIC to enable the SRIOV BIOS policy before enabling the IOMMU driver in the Linux kernel.

Step 1 In the **Navigation** pane, click **Compute**.

Step 2 Expand **BIOS** > **Configure BIOS** > **I/O**.

Step 3 Select **Intel VT for direct IO** to **Enabled**.

Step 4 Click **Save**.

Step 5 Reboot the host for the changes to take effect.

Configuring RoCEv2 for NVMeoF on the Host System

Before you begin

Configure the server with RoCEv2 vNIC and the SRIOV-enabled BIOS policy.

Step 1 Open the `/etc/default/grub` file for editing.

Step 2 Add `intel_iommu=on` at the end of the line in `GRUB_CMDLINE_LINUX` as shown in the following example:

```
sample /etc/default/grub configuration file after adding intel_iommu=on:
# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap biosdevname=1 rhgb quiet
 intel_iommu=on"
GRUB_DISABLE_RECOVERY="true"
```

Step 3 Save the file.

Step 4 Run the following command to generate a new `grub.cfg` file:

- For Legacy boot:

  ```
  # grub2-mkconfig -o /boot/grub2/grub.cfg
  ```

- For UEFI boot:

  ```
  # grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
  ```

Step 5 Reboot the server for the changes to take effect after enabling IOMMU.

Step 6 Use the following to check the output file and verify that the server is booted with the `intel_iommu=on` option:

```
cat /proc/cmdline | grep iommu
```

Note its inclusion at the end of the output.

Example:

```
[root@localhost basic-setup]# cat /proc/cmdline | grep iommu
BOOT_IMAGE=/vmlinuz-3.10.0-957.27.2.el7.x86_64 root=/dev/mapper/rhel-root ro crashkernel=auto
rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet intel_iommu=on LANG=en_US.UTF-8
```

What to do next

Download the enic and enic_rdma drivers.

Installing Cisco enic and enic_rdma Drivers

The enic_rdma driver requires enic driver. When installing enic and enic_rdma drivers, download and use the matched set of enic and enic_rdma drivers from here. Do not attempt to use the binary enic_rdma driver downloaded from cisco.com with an inbox enic driver.

Step 1 Run the following command to tnstall the enic and enic_rdma rpm packages:

```
# rpm -ivh kmod-enic-<version>.x86_64.rpm kmod-enic_rdma-<version>.x86_64.rpm
```

The enic_rdma driver is now installed but not loaded in the running kernel.

During enic_rdma installation, enic_rdmalibnvdimm module might fail to install on RHEL 7.7 because the nvdimm-security.conf dracut module requires spaces in the add_drivers value. For the workaround, follow the instructions from the following links:

- https://access.redhat.com/solutions/4386041
- https://bugzilla.redhat.com/show_bug.cgi?id=1740383

Step 2 Reboot the server to load enic_rdma driver into the running kernel.

Step 3 Run the following command to verify the installation of enic_rdma driver and RoCEv2 interface:

```
# dmesg | grep enic_rdma
[    4.025979] enic_rdma: Cisco VIC Ethernet NIC RDMA Driver, ver 1.0.0.6-802.21 init
[    4.052792] enic 0000:62:00.1 eth1: enic_rdma: IPv4 RoCEv2 enabled
[    4.081032] enic 0000:62:00.2 eth2: enic_rdma: IPv4 RoCEv2 enabled
```

Step 4 Run the following command to load the nvme-rdma kernel module:

```
# modprobe nvme-rdma
```

After the server reboots, nvme-rdma kernel module is unloaded. To load nvme-rdma kernel module on every server reboot, create nvme_rdma.conf file using:

```
# echo nvme_rdma > /etc/modules-load.d/nvme_rdma.conf
```

Note For more information about enic_rdma after installation, use the **rpm -q -l kmod-enic_rdma** command to extract the README file.

What to do next

Discover targets and connect to NVMe namespaces. If your system needs multipath access to the storage, see Setting Up Device Mapper Multipath, on page 7.

Discovering the NVMe Target

Use this procedure to discover the NVMe target and connect NVMe namespaces.

Before you begin

- Ensure that you have **nvme-cli** version 1.6 or later.
- Configure the IP address on the RoCEv2 interface and make sure the interface can ping the target IP.

Step 1 Perform the following to create an nvme folder in `/etc`, and then manually generate hostnqn.

```
# mkdir /etc/nvme
# nvme gen-hostnqn > /etc/nvme/hostnqn
```

Step 2 Perform the following to create a `settos.sh` file and run the script to set priority flow control (PFC) in IB frames.

Note To avoid failure of sending NVMeoF traffic, you must create and run this script after every server reboot.

```
# cat settos.sh
#!/bin/bash
for f in `ls /sys/class/infiniband`;
do
        echo "setting TOS for IB interface:" $f
        mkdir -p /sys/kernel/config/rdma_cm/$f/ports/1
        echo 186 > /sys/kernel/config/rdma_cm/$f/ports/1/default_roce_tos
done
```

Step 3 Run the following command to discover the NVMe target:

```
nvme discover --transport=rdma --traddr=<IP address of transport target port>
```

Example:

To discover the target at 50.2.85.200:

```
# nvme discover --transport=rdma --traddr=50.2.85.200

Discovery Log Number of Records 1, Generation counter 2
=====Discovery Log Entry 0======
trtype:  rdma
adrfam:  ipv4
subtype: nvme subsystem
treq:    not required
portid:  3
trsvcid: 4420
subnqn:  nqn.2010-06.com.purestorage:flasharray.9a703295ee2954e
```

```
traddr:  50.2.85.200
rdma_prtype: roce-v2
rdma_qptype: connected
rdma_cms:    rdma-cm
rdma_pkey: 0x0000
```

Note To connect to the discovered NVMe target using IPv6, put the IPv6 target address next to the **traddr** option.

Step 4 Run the following command to connect to the discovered NVMe target:

```
nvme connect --transport=rdma --traddr=<IP address of transport target port>> -n <subnqn value from
 nvme discover>
```

Example:

To discover the target at 50.2.85.200 and the subnqn value found above:

```
# nvme connect --transport=rdma --traddr=50.2.85.200 -n
nqn.2010-06.com.purestorage:flasharray.9a703295ee2954e
```

Step 5 Use the **nvme list** command to verify the mapped namespaces:

```
# nvme list
Node             SN                   Model                    Namespace Usage
 Format           FW Rev
---------------- -------------------- ------------------------- --------- -----------------------
 ---------------- --------
/dev/nvme0n1     09A703295EE2954E     Pure Storage FlashArray     72656     4.29  GB /   4.29  GB
  512   B +  0 B   99.9.9
/dev/nvme0n2     09A703295EE2954E     Pure Storage FlashArray     72657     5.37  GB /   5.37  GB
  512   B +  0 B   99.9.9
```

Setting Up Device Mapper Multipath

If your system is configured with Device Mapper Multipathing (DM Multipath), use this procedure to set up device mapper multipath.

Step 1 Install the `device-mapper-multipath` package.

Step 2 Perform the following to enable and start **multipathd**:

```
# mpathconf --enable --with_multipathd y
```

Step 3 Edit the **etc/multipath.conf** file to use the following values:

```
defaults {
        polling_interval        10
        path_selector           "queue-length 0"
        path_grouping_policy    multibus
        fast_io_fail_tmo        10
        no_path_retry           0
        features                0
        dev_loss_tmo            60
        user_friendly_names     yes
}
```

Step 4 Perform the following to flush with the updated multipath device maps:

```
# multipath -F
```

Step 5 Perform the following to restart multipath service:

```
# systemctl restart multipathd.service
```

Step 6 Perform the following to rescan multipath devices:

```
# multipath -v2
```

Step 7 Perform the following to check the multipath status:

```
# multipath -ll
```

Deleting RoCEv2 Interface Using Cisco IMC CLI

SUMMARY STEPS

1. server # **scope chassis**
2. server/chassis # **scope adapter** *index_number*
3. server/chassis/adapter # **scope host-eth-if** *vNIC_name*
4. server/chassis/adapter/host-eth-if # **set rocev2 disabled**
5. server/chassis/adapter/host-eth-if *# **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	server # **scope chassis**	Enters the chassis command mode.
Step 2	server/chassis # **scope adapter** *index_number*	Enters the command mode for the adapter card at the PCI slot number specified by *index_number.* **Note** Ensure that the server is powered on before you attempt to view or change adapter settings. To view the *index* of the adapters configured on you server, use the **`show adapter`** command.
Step 3	server/chassis/adapter # **scope host-eth-if** *vNIC_name*	Enters the command mode for the vNIC specified by *vNIC_name*.
Step 4	server/chassis/adapter/host-eth-if # **set rocev2 disabled**	Disables RoCE properties on the vNIC.
Step 5	server/chassis/adapter/host-eth-if *# **commit**	Commits the transaction to the system configuration. **Note** The changes take effect when the server is rebooted.

Example

```
server# scope chassis
server/chassis # scope adapter 1
server/chassis/adapter # scope host-eth-if vNIC_Test
server/chassis/adapter/host-eth-if # set rocev2 disabled
server/chassis/adapter/host-eth-if *# commit
```