



# NVIDIA DOCA Emulated Devices

## User Guide

# Table of Contents

<b>Chapter 1. Mediated Devices</b> .....	<b>1</b>
1.1. Related Configuration.....	1
<b>Chapter 2. VirtIO-net Emulated Devices</b> .....	<b>3</b>
2.1. VirtIO-net PF Devices.....	3
2.1.1. VirtIO-net PF Device Configuration.....	3
2.1.2. Creating Hotplug VirtIO-net PF Device.....	4
<b>Chapter 3. Virtio-net SR-IOV VF Devices</b> .....	<b>5</b>
3.1. Virtio-net SR-IOV VF Device Configuration.....	5
3.2. Creating Virtio-net SR-IOV VF Devices.....	6

---

# Chapter 1. Mediated Devices

NVIDIA mediated devices deliver flexibility in allowing to create accelerated devices without SR-IOV on the BlueField® system. These mediated devices support NIC and RDMA and offer the same level of ASAP2 offloads as SR-IOV VFs. Mediated devices are supported using mlx5 sub-function acceleration technology.

Two sub-function devices are created on the BlueField device upon boot (one per port if the port is in switchdev mode) using commands from `/etc/mellanox/mlnx-sf.conf`:

```
/sbin/mlnx-sf -a create -d 0000:03:00.0 -u 61a59715-aeec-42d5-be83-f8f42ba8b049 --
mac 12:11:11:11:11:11
/sbin/mlnx-sf -a create -d 0000:03:00.1 -u 5b198182-1901-4c29-97a0-6623f3d02065 --
mac 12:11:11:11:11:12
```

The help menu for `mlnx-sf` is presented below:

```
Usage: mlnx-sf [ OPTIONS ]
OPTIONS:
-a, -action,      --action <action>          Perform action
   action:        { enable | create | configure | remove | show | set_max_mdevs |
   query_mdevs_num }
-d, -device,      --device <device>          PCI device
   <domain>:<bus>:<device>.<func> (E.g.: 0000:03:00.0)
-m, -max_mdevs,  --max_mdevs <max mdevs number> Set maximum number of MDEVs
-u, -uuid,        --uuid <uuid>             UUID to create SF with
-M, -mac,         --mac <MAC>               MAC to create SF with
-p, -permanent,  --permanent [<conf file>]    Store configuration to be used after
   reboot and/or driver restart. Default (/etc/mellanox/mlnx-sf.conf).
-V, -version,    --version                   Display script version and exit
-D, -dryrun,     --dryrun                    Display commands only
-v, -verbose,    --verbose                   Run script in verbose mode (print
   out every step of execution)
-h, -help,       --help                     Display help

"/etc/mellanox/mlnx-sf.conf" can be updated manually or using "mlnx-sf" tool with "-
p" parameter.
```

## 1.1. Related Configuration

Interface names are configured using the UDEV rule under: `/etc/udev/rules.d/82-net-setup-link.rules`.

```
SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}!="", ATTR{phys_port_name}!="",
\
    IMPORT{program}="/etc/infiniband/vf-net-link-name.sh $attr{phys_switch_id}
$attr{phys_port_name}" \
    NAME="$env{NAME}", RUN+="/sbin/ethtool -L $env{NAME} combined 4"
# MDEV network interfaces
```

```
ACTION=="add", SUBSYSTEM=="net", DEVPATH="/devices/  
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.0/61a59715-aeec-42d5-  
be83-f8f42ba8b049/net/eth[0-9]", NAME="p0m0"  
ACTION=="add", SUBSYSTEM=="net", DEVPATH="/devices/  
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.1/5b198182-1901-4c29-97a0-6623f3d0206  
net/eth[0-9]", NAME="p1m0"
```

NVMe SNAP uses p0m0 as its default interface. See `/etc/nvme_snap/sf1.conf`.

---

# Chapter 2. VirtIO-net Emulated Devices



**Note:** This feature is supported at beta level. Please contact NVIDIA Support for enablement.

This feature enables users to create VirtIO-net emulated PCIe devices in the system where the NVIDIA® BlueField®-2 DPU is connected. This is done by the virtio-net-controller software module present in the DPU. Virtio-net emulated devices allow users to hot plug up to 127 virtio-net PCIe PF Ethernet NIC devices and virtio-net PCI VF Ethernet NIC devices in the host system where the DPU is plugged in.

DPU software also enables users to create virtio block PCI PF and SR-IOV PCI VF devices. This is covered in the *NVIDIA Mellanox NVMe SNAP and virtio-blk SNAP Documentation*.

## 2.1. VirtIO-net PF Devices

This section covers managing virtio-net PCI PF devices using virtio-net-controller.

### 2.1.1. VirtIO-net PF Device Configuration

1. Run the following command on the DPU:

```
$ mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```

2. Cold reboot the host system.

3. Apply the following configuration on the DPU in three steps:

```
$ mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1
```

```
$ mlxconfig -d /dev/mst/mt41686_pciconf0 s \  
PCI_SWITCH_EMULATION_ENABLE=1 \  
PCI_SWITCH_EMULATION_NUM_PORT=16 \  
VIRTIO_NET_EMULATION_ENABLE=1 \  
VIRTIO_NET_EMULATION_NUM_VF=0 \  
VIRTIO_NET_EMULATION_NUM_PF=0 \  
VIRTIO_NET_EMULATION_NUM_MSIX=16 \  
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \  
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \  
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \  
ECPF_ESWITCH_MANAGER=1 \  
ECPF_PAGE_SUPPLIER=1 \  
SRIOV_EN=0 \  
PF_SF_BAR_SIZE=10 \  
PF_TOTAL_SF=64
```

```
$ mlxconfig -d /dev/mst/mt41686_pciconf0.1 s \  
PF_SF_BAR_SIZE=10 \  

```

```
PF_TOTAL_SF=64
```

4. Cold reboot the host system a second time.

## 2.1.2. Creating Hotplug VirtIO-net PF Device

VirtIO emulated network PCIe devices are created and destroyed using virtio-net-controller application console. When this application is terminated, all created VirtIO-net emulated devices are hot unplugged.

1. Start virtio-net controller on the DPU. Run:

```
$ virtio_net_controller -i mlx5_0
```

2. Create a hotplug virtio-net device. Run:

```
$ create virtio-net dev 0x0 0C:C4:7A:FF:22:93 1500 3 1024
```

This creates one hotplug virtio-net device with MAC address 0C:C4:7A:FF:22:93, MTU 1500, and 3 VirtIO queues with a depth of 1024 entries. This device is uniquely identified by its index. This index is used to query and update device attributes. This device is uniquely identified by its index. This index is used to query and update device attributes.

3. Set the link state up of the newly create device. Run:

```
$ change virtio-net dev 0 link 1
```

4. Change the SF netdevice's queue parameters. The SF's netdevice is printed by the dump command of the virtio-net-controller. It is also printed during the create command. Run:

```
$ ethtool -L <sf_netdev> combined 1
$ ethtool -G <sf_netdev> rx 16
$ ethtool -G <sf_netdev> tx 128
```

5. Bring up the representor port of the device. Run:

```
$ ip link set dev pf0sf1 up
$ ovs-vsctl add-port <bridge> pf0sf1
```

Once steps 1-5 are completed, virtio-net device should be available in the host system.

6. To show all the device configurations of virtio-net device that you created, run:

```
$ show virtio-net dev 0
```

7. To list all the virtio-net devices, run:

```
$ list virtio-net devices
```

8. Once usage is complete, to hot-unplug a VirtIO net device, run:

```
$ destroy virtio-net dev 0
```

---

# Chapter 3. Virtio-net SR-IOV VF Devices

This section covers managing virtio-net PCI SR-IOV VF devices using virtio-net-controller.

## 3.1. Virtio-net SR-IOV VF Device Configuration



**Note:** Virtio-net SR-IOV VF is only supported with statically configured PF, hot-plugged PF is not currently supported.

1. On the x86 host, blacklist kernel modules virtio\_pci and virtio\_net by adding them to `/etc/modprobe.d/blacklist.conf`:

```
blacklist virtio_pci
blacklist virtio_net
```
2. On the x86 host, enable SR-IOV. Please refer to [MLNX\\_OFED documentation](#) under Features Overview and Configuration > Virtualization > Single Root IO Virtualization (SR-IOV) > Setting Up SR-IOV for instructions on how to do that. Make sure the parameters `intel_iommu=on iommu=pt pci=realloc` exist in `grub.conf` file.
3. Reboot x86 host, make sure those modules are not loaded.

```
# lsmod | grep -i virtio
```



**Note:** Please make sure those modules are blacklisted before proceeding. If those modules are built into kernel image, they are not blacklistable. In that case, users must have a way to start the `virtio_net_controller` from Arm using SSH or serial/oob interfaces. This can be achieved by connecting the RShim/serial cable to another host, or setting up the OOB interface.

4. Run the following command on the DPU:

```
mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```
5. Cold reboot the host system.
6. Apply the following configuration on the DPU in three steps.
  - a). 

```
$ mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1
```
  - b). 

```
$ mlxconfig -d /dev/mst/mt41686_pciconf0 s \
PCI_SWITCH_EMULATION_ENABLE=1 \
```

```

PCI_SWITCH_EMULATION_NUM_PORT=16 \
VIRTIO_NET_EMULATION_ENABLE=1 \
VIRTIO_NET_EMULATION_NUM_VF=125 \
VIRTIO_NET_EMULATION_NUM_PF=4 \
VIRTIO_NET_EMULATION_NUM_MSIX=4 \
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \
ECPF_ESWITCH_MANAGER=1 \
ECPF_PAGE_SUPPLIER=1 \
SRIOV_EN=1 \
PF_SF_BAR_SIZE=8 \
PF_TOTAL_SF=508

```

```
c). $ mlxconfig -d /dev/mst/mt41686_pciconf0.1 s PF_TOTAL_SF=170 PF_SF_BAR_SIZE=8
```

7. Cold reboot the host system.

## 3.2. Creating Virtio-net SR-IOV VF Devices

The virtio-net-controller application console must be kept alive to maintain the functionality of the static PF and its VFs.

1. Start virtio-net controller on the DPU. Run:

```
# virtio_net_controller -i mlx5_0
```

2. On the x86 host, make sure the static virtio network device presents.

```
# lspci | grep -i virtio
85:00.3 Network controller: Red Hat, Inc. Virtio network device
```

3. On x86 host, run:

```
# modprobe -v virtio_pci
# modprobe -v virtio_net
```

The net device should be created:

```
# ethtool -i p7p3
driver: virtio_net
version: 1.0.0
firmware-version:
expansion-rom-version:
bus-info: 0000:85:00.3
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

4. To create SR-IOV VF devices on the x86 host, run:

```
# echo 2 > /sys/bus/pci/drivers/virtio-pci/0000\:85\:00.3/sriov_numvfs
```

2 VFs should be created from x86 host:

```
# lspci | grep -i virt
85:00.3 Network controller: Red Hat, Inc. Virtio network device
85:04.5 Network controller: Red Hat, Inc. Virtio network device
85:04.6 Network controller: Red Hat, Inc. Virtio network device
```

5. From the DPU virtio-net controller, run:

```
virtio-net-controller> list virtio-net devices
PF VF BDF      SF-Rep  Fw Open Stat Features      Queues  Msix  MAC              MTU
0   0 85:0.3  pf0sf1  Y   Y  0x0f 0x2300470028  8       0    0cc47aff2292  1500
    0 85:4.5  pf0sf2  Y   Y  0x0f 0x300470028  8       0    c6237b326745  1500
    1 85:4.6  pf0sf3  Y   Y  0x0f 0x300470028  8       0    724833666998  1500
```

Bring up corresponding SF and add to OVS bridge:



```
# ip link set dev pf0sf2 up  
# ovs-vsctl add-port <bridge> pf0sf2
```

Now the VF should be functional.

## Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation nor any of its direct or indirect subsidiaries (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

## Trademarks

NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## Copyright

© 2021 NVIDIA Corporation. All rights reserved.