



Deploying BlueField Software Using BFB from Host

Table of contents

Uninstall Previous Software from Host

Install RShim on Host

Ensure RShim Running on Host

Installing Ubuntu on BlueField

BFB Installation

Verify BFB is Installed

Changing Default Credentials Using bf.cfg

Password Policy

GRUB Password Protection

Firmware Upgrade

Updating NVConfig Params from Host

Customizations During BFB Installation

Default Ports and OVS Configuration

Customization of BFB Installation Using bf.cfg

bf.cfg Parameters

Default Network Interface Configuration

Ubuntu Boot Time Optimizations

DHCP Client Configuration

Ubuntu Dual Boot Support

Installing Ubuntu OS Image Using Dual Boot

Upgrading Ubuntu OS Image Using Dual Boot

Info

It is recommended to upgrade your BlueField product to the latest software and firmware versions available to benefit from new features and latest bug fixes.

Note

This procedure assumes that a BlueField DPU has already been installed in a server according to the instructions detailed in the [DPU's hardware user guide](#).

The following table lists an overview of the steps required to install Ubuntu BFB on your DPU:

Step	Procedure	Link to Section
1	Uninstall previous DOCA on host (if exists)	Uninstall Previous Software from Host
2	Install RShim on the host	Install RShim on Host
3	Verify that RShim is running on the host	Ensure RShim Running on Host
4	Change the default credentials using bf.cfg file (optional)	Changing Default Credentials Using bf.cfg
5	Install the Ubuntu BFB image	BFB Installation
6	Verify installation completed successfully	Verify BFB is Installed
7	Upgrade the firmware on your DPU	Firmware Upgrade

Uninstall Previous Software from Host

If an older DOCA software version is installed on your host, make sure to uninstall it before proceeding with the installation of the new version:

Ubuntu	<pre>host# for f in \$(dpkg --list grep doca awk '{print \$2}'); do echo \$f ; apt remove --purge \$f -y ; done host# sudo apt-get autoremove</pre>
CentOS/RHEL	<pre>host# for f in \$(rpm -qa grep -i doca) ; do yum -y remove \$f; done host# yum autoremove host# yum makecache</pre>

Install RShim on Host

Before installing the RShim driver, verify that the RShim devices, which will be probed by the driver, are listed under `lsusb` or `lspci`.

```
lspci | grep -i nox
```

Output example:

```
27:00.0 Ethernet controller: Mellanox Technologies MT42822 BlueField-2 integrated  
ConnectX-6 Dx network controller  
27:00.1 Ethernet controller: Mellanox Technologies MT42822 BlueField-2 integrated  
ConnectX-6 Dx network controller  
27:00.2 Non-Volatile memory controller: Mellanox Technologies NVMe SNAP  
Controller  
27:00.3 DMA controller: Mellanox Technologies MT42822 BlueField-2 SoC  
Management Interface // This is the RShim PF
```

RShim is compiled as part of the `doca-runtimepackage` in the `doca-host-repo-ubuntu<version>_amd64` file (.deb or .rpm).

To install `doca-runtime`:

OS	Procedure
Ubuntu/Debian	<ol style="list-style-type: none"> 1. Download the DOCA Runtime host package from the "Installation Files" section in the <i>NVIDIA DOCA Installation Guide for Linux</i>. 2. Unpack the deb repo. Run: <pre data-bbox="492 359 1463 499">host# sudo dpkg -i doca-host-repo-ubuntu<version>_amd64.deb</pre> 3. Perform apt update. Run: <pre data-bbox="492 548 1463 636">host# sudo apt-get update</pre> 4. Run apt install for DOCA runtime package. <pre data-bbox="492 684 1463 772">host# sudo apt install doca-runtime</pre>
CentOS/RHEL 7.x	<ol style="list-style-type: none"> 1. Download the DOCA runtime host package from the "Installation Files" section in the <i>NVIDIA DOCA Installation Guide for Linux</i>. 2. Unpack the RPM repo. Run: <pre data-bbox="492 961 1463 1102">host# sudo rpm -Uvh doca-host-repo-rhel<version>.x86_64.rpm</pre> 3. Enable new yum repos. Run: <pre data-bbox="492 1150 1463 1239">host# sudo yum makecache</pre> 4. Run yum install to install DOCA runtime package. <pre data-bbox="492 1287 1463 1375">host# sudo yum install doca-runtime</pre>
CentOS/RHEL 8.x or Rocky 8.6	<ol style="list-style-type: none"> 1. Download the DOCA runtime host package from the "Installation Files" section in the <i>NVIDIA DOCA Installation Guide for Linux</i>. 2. Unpack the RPM repo. Run: <pre data-bbox="492 1564 1463 1705">host# sudo rpm -Uvh doca-host-repo-rhel<version>.x86_64.rpm</pre> 3. Enable new dnf repos. Run: <pre data-bbox="492 1753 1463 1841">host# sudo dnf makecache</pre> 4. Run dnf install to install DOCA runtime.

OS	Procedure
	<pre>host# sudo dnf install doca-runtime</pre>

Ensure RShim Running on Host

1. Verify RShim status. Run:

```
sudo systemctl status rshim
```

Expected output:

```
active (running)
...
Probing pcie-0000:<BlueField's PCIe Bus address on host>
create rshim pcie-0000:<BlueField's PCIe Bus address on host>
rshim<N> attached
```

Where <N> denotes RShim enumeration starting with 0 (then 1, 2, etc.) for every additional DPU installed on the server.

If the text "another backend already attached" is displayed, users will not be able to use RShim on the host. Please refer to "[RShim Troubleshooting and How-Tos](#)" to troubleshoot RShim issues.

1. If the previous command displays `inactive` or another error, restart RShim service. Run:

```
sudo systemctl restart rshim
```

2. Verify RShim status again. Run:

```
sudo systemctl status rshim
```

If this command does not display "active (running)", then refer to "[RShim Troubleshooting and How-Tos](#)".

2. Display the current setting. Run:

```
# cat /dev/rshim<N>/misc | grep DEV_NAME  
DEV_NAME pcie-0000:04:00.2
```

This output indicates that the RShim service is ready to use.

Installing Ubuntu on BlueField

BFB Installation

Note

Check the BFB version installed on your BlueField-2 DPU. If the version is 1.5.0 or lower, please see Known Issue Reference #3600716 under [Known Issues](#) section.

Info

To upgrade the BMC firmware using BFB, the user must provide the current BMC credentials in the [bf.cfg](#).

Note

Upgrading the BlueField networking platform using BFB Bundle updates the NIC firmware by default. NIC firmware upgrade triggers a NIC reset flow via `mlxfwreset` in the BlueField Arm.

If this reset flow cannot complete or is not supported on your setup, bfb-install alerts about it at the end of the installation. In this case, perform a [BlueField system reboot](#) for the mlxconfig settings to take effect.

To skip NIC firmware upgrade during BFB Bundle installation , provide the parameter WITH_NIC_FW_UPDATE=no in the bf.cfg text file when running bfb-install .

A pre-built BFB of Ubuntu 22.04 with DOCA Runtime and DOCA packages installed is available on the [NVIDIA DOCA SDK developer zone](#) page.

Note

All new BlueField-2 devices and all BlueField-3 are secure boot enabled, hence all the relevant SW images (ATF/UEFI, Linux Kernel and Drivers) must be signed in order to boot. All formally published SW images are signed.

Warning

When installing the BFB bundle in NIC mode, users must perform the following:

1. Prior to installing the BFB bundle, users must unbind each NIC port, using its PCIe function address. For example:

```
host]# lspci -d 15b3:
21:00.0 Ethernet controller: Mellanox Technologies
MT43244 BlueField-3 integrated ConnectX-7 network
controller (rev 01)
```



```
21:00.1 Ethernet controller: Mellanox Technologies
MT43244 BlueField-3 integrated ConnectX-7 network
controller (rev 01)
21:00.2 DMA controller: Mellanox Technologies MT43244
BlueField-3 SoC Management Interface (rev 01)

host]# echo 0000:21:00.0 >
/sys/bus/pci/drivers/mlx5_core/unbind
host]# echo 0000:21:00.1 >
/sys/bus/pci/drivers/mlx5_core/unbind
```

If there are multiple BlueField devices to be updated in the server, repeat this step on all of them, before starting BFB bundle installations.

2. After the BFB bundle installation is done, users must perform a warm reboot on the host.

To install Ubuntu BFB, run on the host side:

```
# bfb-install -h
syntax: bfb-install --bfb | -b <BFBFILE> [--config | -c <bf.cfg>] \
[--rootfs | -f <rootfs.tar.xz>] --rshim | -r <rshimN> [--help | -h]
```

The bfb-install utility is installed by the RShim package.

This utility script pushes the BFB image and optional configuration (bf.cfg file) to the BlueField side and checks and prints the BFB installation progress. To see the BFB installation progress, please install the pv Linux tool.

Warning

BFB image installation must complete before restarting the system/BlueField. Doing so may result in the BlueField DPU not operating as expected (e.g., it may not be accessible using SSH). If this

happens, re-initiate the update process with bfb-install to recover the DPU.

The following is an output example of Ubuntu 20.04 installation with the bfb-install script assuming pv has been installed.

```
# bfb-install --bfb <BlueField-BSP>.bfb --config bf.cfg --rshim rshim0 Pushing bfb +
cfg
1.46GiB 0:01:11 [20.9MiB/s] [ <=> ]
Collecting BlueField booting status. Press Ctrl+C to stop...
INFO[PSC]: PSC BL1 START
INFO[BL2]: start
INFO[BL2]: boot mode (rshim)
INFO[BL2]: VDDQ: 1120 mV
INFO[BL2]: DDR POST passed
INFO[BL2]: UEFI loaded
INFO[BL31]: start
INFO[BL31]: lifecycle Production
INFO[BL31]: MB8: VDD adjustment complete
INFO[BL31]: VDD: 743 mV
INFO[BL31]: power capping disabled
INFO[BL31]: runtime
INFO[UEFI]: eMMC init
INFO[UEFI]: eMMC probed
INFO[UEFI]: UPVS valid
INFO[UEFI]: PMI: updates started
INFO[UEFI]: PMI: total updates: 1
INFO[UEFI]: PMI: updates completed, status 0
INFO[UEFI]: PCIe enum start
INFO[UEFI]: PCIe enum end
INFO[UEFI]: UEFI Secure Boot (disabled)
INFO[UEFI]: exit Boot Service
INFO[MISC]: : Found bf.cfg
INFO[MISC]: : Ubuntu installation started
INFO[MISC]: bfb_pre_install
```

```
INFO[MISC]: Installing OS image
INFO[MISC]: : Changing the default password for user ubuntu
INFO[MISC]: : Running bfb_modify_os from bf.cfg
INFO[MISC]: : Ubuntu installation finished
```

Verify BFB is Installed

After installation of the Ubuntu OS is complete, the following note appears in `/dev/rshim0/misc` on first boot:

```
...
INFO[MISC]: Linux up
INFO[MISC]: DPU is ready
```

"DPU is ready" indicates that all the relevant services are up and users can login the system.

After the installation of the Ubuntu 20.04 BFB, the configuration detailed in the following sections is generated.

Note

Make sure all the services (including cloud-init) are started on BlueField and to perform a graceful shutdown before power cycling the host server.

BlueField OS image version is stored under `/etc/mlnx-release` in the BlueField:

```
# cat /etc/mlnx-release
bf-bundle-2.7.0-<version>_ubuntu-22.04_prod
```

Changing Default Credentials Using bf.cfg

Info

For a comprehensive list of the supported parameters to customize bf.cfg during BFB installation, refer to section "[bf.cfg Parameters](#)".

Ubuntu users are prompted to change the default password (ubuntu) for the default user (ubuntu) upon first login. Logging in will not be possible even if the login prompt appears until all services are up ("DPU is ready" message appears in /dev/rshim0/misc).

Note

Attempting to log in before all services are up prints the following message: Permission denied, please try again.

Alternatively, Ubuntu users can provide a unique password that will be applied at the end of the BFB installation. This password must be defined in a bf.cfg configuration file. To set the password for the ubuntu user:

1. Create password hash. Run:

```
# openssl passwd -1  
Password:  
Verifying - Password:
```

```
$1$3B0RlrfX$TIHry93NFUJzg3Nya00rE1
```

2. Add the password hash in quotes to the bf.cfg file:

```
# vim bf.cfg
ubuntu_PASSWORD='$1$3B0RlrfX$TIHry93NFUJzg3Nya00rE1'
```

The bf.cfg file is used with the bfb-install script in the steps that follow.

Password Policy

The following table provides the password policy parameters.

Config File Path	Parameter	Value	Description
/etc/security/pwquality.conf	minlen	12	Minimum password length
/etc/pam.d/common-password	remember	3	The number of previous passwords which cannot be reused
/etc/security/faillock.conf	silent	Uncommented	Prevents printing informative messages to the user
	deny	10	The number of authentication attempts permitted before the user is locked out
	unlock_time	600	The duration, in seconds, of the lockout period

Info

Each of these parameters is configurable in its respective config file indicated in the "Config File Path" column.

Info

Please refer to the "[Default Passwords and Policies](#)" section for more password policy information.

GRUB Password Protection

GRUB menu entries are protected by a username and password to prevent unwanted changes to the default boot options or parameters.

The default credentials are as follows:

Username	admin
Password	BlueField

The password can be changed during BFB installation by providing a new `grub_admin_PASSWORD` parameter in `bf.cfg`:

```
# vim bf.cfg
grub_admin_PASSWORD='
grub.pbkdf2.sha512.10000.5EB1FF92FDD89BDAF3395174282C77430656A6DBEC1F92
```

To get a new encrypted password value use the command `grub-mkpasswd-pbkdf2`.

After the installation, the password can be updated by editing the file `/etc/grub.d/40_custom` and then running the command `update-grub` which updates the file `/boot/grub/grub.cfg`.

Firmware Upgrade

To upgrade firmware:

1. Access the BlueField using one of the available interfaces (RShim console, BMC console, SSH via `oob_net0` or `tmfifo_net0` interfaces).
2. Upgrade the firmware on the DPU. Run:

```
sudo /opt/mellanox/mlnx-fw-updater/mlnx_fw_updater.pl --force-fw-update
```

Example output:

```
Device #1:
-----

Device Type: BlueField-2
[...]
Versions: Current Available
FW <Old_FW> <New_FW>
```

Note

Important! To apply NVConfig changes, stop here and follow the steps in section "Updating NVConfig Params". In this case, the following step #3 is redundant.

3. Perform a [BlueField system reboot](#) for the upgrade to take effect.

Updating NVConfig Params from Host

1. Optional. To reset the BlueField NIC firmware configuration (aka Nvconfig params) to their factory default values, run the following from the BlueField ARM OS or from the host OS:

```
# sudo mlxconfig -d /dev/mst/<MST device> -y reset

Reset configuration for device /dev/mst/<MST device>? (y/n) [n] : y
Applying... Done!
-l- Please reboot machine to load new configurations.
```

Note

For now, please ignore tool's instruction to reboot

Note

To learn what MST device the BlueField DPU has on your setup, run:

```
mst start
mst status
```


Example output taken on a multiple DPU host:

```
// The MST device corresponds with PCI Bus address.

MST modules:
-----
MST PCI module is not loaded
MST PCI configuration module loaded

MST devices:
-----
/dev/mst/mt41692_pciconf0 - PCI configuration cycles
access.
domain:bus:dev.fn=0000:03:00.0 addr.reg=88 data.reg=92
cr_bar.gw_offset=-1
Chip revision is: 01
/dev/mst/mt41692_pciconf1 - PCI configuration cycles
access.
domain:bus:dev.fn=0000:83:00.0 addr.reg=88 data.reg=92
cr_bar.gw_offset=-1
Chip revision is: 01
/dev/mst/mt41686_pciconf0 - PCI configuration cycles
access.
domain:bus:dev.fn=0000:a3:00.0 addr.reg=88 data.reg=92
cr_bar.gw_offset=-1
Chip revision is: 01
```

The MST device IDs for the BlueField-2 and BlueField-3 DPUs in this example are `/dev/mst/mt41686_pciconf0` and `/dev/mst/mt41692_pciconf0` respectively.

2. (Optional) Enable NVMe emulation. Run:

```
sudo mlxconfig -d <MST device> -y s NVME_EMULATION_ENABLE=1
```

3. Skip this step if your BlueField DPU is Ethernet only. Please refer to section "Supported Platforms and Interoperability" under the Release Notes to learn your DPU type.

If you have a VPI DPU, the default link type of the ports will be configured to IB. If you want to change the link type to Ethernet, please run the following configuration:

```
sudo mlxconfig -d <MST device> -y s LINK_TYPE_P1=2 LINK_TYPE_P2=2
```

4. Perform a [BlueField system-level reset](#) for the new settings to take effect.

Customizations During BFB Installation

Using special purpose configuration parameters in the bf.cfg file, the BlueField's boot options and OS can be further customized. For a full list of the supported parameters to customize your DPU system during BFB installation, refer to section "[bf.cfg Parameters](#)". In addition, the bf.cfg file offers further control on customization of BlueField OS installation and software configuration through scripting.

Add any of the following functions to the bf.cfg file for them to be called by the install.sh script embedded in the BFB:

- `bfb_modify_os` – called after the file system is extracted on the target partitions. It can be used to modify files or create new files on the target file system mounted under `/mnt`. So the file path should look as follows: `/mnt/<expected_path_on_target_OS>`. This can be used to run a specific tool from the target OS (remember to add `/mnt` to the path for the tool).
- `bfb_pre_install` – called before eMMC/SSD partitions format and OS filesystem is extracted
- `bfb_post_install` – called as a last step before reboot. All eMMC/SSD partitions are unmounted at this stage.

For example, the bf.cfg script below disables OVS bridge creation upon boot:

```

# cat /root/bf.cfg

bfb_modify_os()
{
log ===== bfb_modify_os =====
log "Disable OVS bridges creation upon boot"
sed -i -r -e 's/(CREATE_OVS_BRIDGES=).*\1"no"/ /mnt/etc/mellanox/mlnx-ovs.conf
}

bfb_pre_install()
{
log ===== bfb_pre_install =====
}

bfb_post_install()
{
log ===== bfb_post_install =====
}

```

Note

After modifying files on the BlueField, run the command `sync` to flush file system buffers to eMMC/SSD flash memory to avoid data loss during reboot or power cycle.

Default Ports and OVS Configuration

The `/sbin/mlnx_bf_configure` script runs automatically with `ib_umad` kernel module loaded (see `/etc/modprobe.d/mlnx-bf.conf`) and performs the following configurations:

1. Ports are configured with switchdev mode and software steering.
2. RDMA device isolation in network namespace is enabled.
3. Two scalable function (SF) interfaces are created (one per port) if BlueField is configured with Embedded CPU mode (default):

```
# mlnx-sf -a show

SF Index: pci/0000:03:00.0/229408
Parent PCI dev: 0000:03:00.0
Representor netdev: en3f0pf0sf0
Function HWADDR: 02:a9:49:7e:34:29
Function trust: off
Function roce: true
Function eswitch: NA
Auxiliary device: mlx5_core.sf.2
netdev: enp3s0f0s0
RDMA dev: mlx5_2

SF Index: pci/0000:03:00.1/294944
Parent PCI dev: 0000:03:00.1
Representor netdev: en3f1pf1sf0
Function HWADDR: 02:53:8f:2c:8a:76
Function trust: off
Function roce: true
Function eswitch: NA
Auxiliary device: mlx5_core.sf.3
netdev: enp3s0f1s0
RDMA dev: mlx5_3
```

The parameters for these SFs are defined in configuration file `/etc/mellanox/mlnx-sf.conf`.

```
/sbin/mlnx-sf --action create --device 0000:03:00.0 --sfnum 0 --hwaddr
02:61:f6:21:32:8c
/sbin/mlnx-sf --action create --device 0000:03:00.1 --sfnum 0 --hwaddr
02:30:13:6a:2d:2c
```

Note

To avoid repeating a MAC address in the your network, the SF MAC address is set randomly upon BFB installation. You may choose to configure a different MAC address that better suit your network needs.

4. Two OVS bridges are created:

```
# ovs-vsctl show
f08652a8-92bf-4000-ba0b-7996c772aff6
Bridge ovsbr2
Port ovsbr2
Interface ovsbr2
type: internal
Port p1
Interface p1
Port en3f1pf1sf0
Interface en3f1pf1sf0
Port pf1hpf
Interface pf1hpf
Bridge ovsbr1
Port p0
Interface p0
Port pf0hpf
Interface pf0hpf
Port ovsbr1
Interface ovsbr1
type: internal
Port en3f0pf0sf0
Interface en3f0pf0sf0
ovs_version: "2.14.1"
```

The parameters for these bridges are defined in configuration file `/etc/mellanox/mlnx-ovs.conf`:

```
CREATE_OVS_BRIDGES="yes"
OVS_BRIDGE1="ovsbr1"
OVS_BRIDGE1_PORTS="p0 pf0hpf en3f0pf0sf0"
OVS_BRIDGE2="ovsbr2"
OVS_BRIDGE2_PORTS="p1 pf1hpf en3f1pf1sf0"
OVS_HW_OFFLOAD="yes"
OVS_START_TIMEOUT=30
```

Note

If failures occur in `/sbin/mlnx_bf_configure` or configuration changes happen (e.g. switching to separated host mode) OVS bridges are not created even if `CREATE_OVS_BRIDGES="yes"`.

5. OVS HW offload is configured.

Customization of BFB Installation Using `bf.cfg`

The BFB installation process as well as the content and configuration of the target OS can be customized during BFB installation process using the `bf.cfg` file. The `bf.cfg` file is passed to the DPU via RShim or using [PXE](#) configuration and is sourced by BFB's installation script at the beginning of the BFB installation process.

Info

Information is available under "[bf.cfg Parameters](#)".

A number of helper functions are available in the BFB's install.sh script to enable customization.


- `bfb_modify_os` – the shell function is called after the file system is extracted on the target partitions. It can be used to modify files or create new files on the target file system mounted under `/mnt`. So the file path should look something like the following: `/mnt/<expected_path_on_target_OS>`. This can be used to run a specific tool from the target OS (remember to add `/mnt` to the path for the tool).
- `bfb_pre_install` – the shell function is called before the partitions format and OS filesystem is extracted.
- `bfb_post_install` – the shell function is called as a last step before reboot. All partitions are unmounted at this stage.

The BFB installation process includes the following tasks:

1. Installing target OS if `UPDATE_DPU_OS="yes"` (default)

1. Creating and formatting partitions on the SSD (default) or EMMC drive.
2. Extracting target OS file system from the tarball file coming with the BFB.
3. Configuring target OS depending on the underlying hardware and provided configuration.
4. Building initramfs for the target OS to make sure all the requirements for boot drivers are included.

2. Updating ATF and UEFI if `UPDATE_ATF_UEFI="yes"` (default).

 **Info**

This is relevant for PXE installation only as ATF and UEFI are updated automatically via RShim.

3. Updating BMC components:

Info

Requires BMC username and password to be provided.

1. Bringing up VLAN 4040 network interface on top of oob_net0. VLAN 4040 is configured with static IP 192.168.240.2/29. The timeout for bringing up the connection with the DPU's BMC VLAN 4040 interface (192.168.240.1) is set to BMC_IP_TIMEOUT (default is 600 seconds).
2. Updating BMC firmware if a different version is available and UPDATE_BMC_FW="yes" (default). The timeout for the BMC firmware update task is BMC_TASK_TIMEOUT (default is 1800 seconds).
3. Updating CEC firmware if a different version is available and UPDATE_CEC_FW="yes" (default).
4. Updating the DPU golden image if a different version is available and UPDATE_DPU_GOLDEN_IMAGE="yes" (default).
5. Updating the NIC firmware golden image if a different version is available and UPDATE_NIC_FW_GOLDEN_IMAGE="yes" (default).
6. Rebooting BMC if its firmware was updated and BMC_REBOOT="yes" (disabled by default).

Note

BMC reboot is required to apply the new BMC firmware version, but BMC reboot resets the BMC console which is used to monitor the BFB installation process. This is why BMC reboot is disabled by default and should be done after the BFB installation process if using the BMC console.

4. NIC firmware update if WITH_NIC_FW_UPDATE="yes" (default).
5. Reboot.

A complete installation log becomes available on the target file system after the installation process is finished (e.g., /root/Ubuntu.installation.log).

bf.cfg Parameters

The following is a comprehensive list of the supported parameters to customize the bf.cfg file for BFB installation:

```
#####  
# Configuration which can also be set in  
# UEFI->Device Manager->System Configuration  
#####  
# Enable SMMU in ACPI.  
#SYS_ENABLE_SMMU = TRUE  
  
# Enable I2C0 in ACPI.  
#SYS_ENABLE_I2C0 = FALSE  
  
# Disable SPMI in ACPI.  
#SYS_DISABLE_SPMI = FALSE  
  
# Enable the second eMMC card which is only available on the BlueField Reference  
Platform.  
#SYS_ENABLE_2ND_EMMC = FALSE  
  
# Enable eMMC boot partition protection.  
#SYS_BOOT_PROTECT = FALSE  
  
# Enable SPCR table in ACPI.  
#SYS_ENABLE_SPCR = FALSE  
  
# Disable PCIe in ACPI.  
#SYS_DISABLE_PCIE = FALSE  
  
# Enable OP-TEE in ACPI.  
#SYS_ENABLE_OPTEE = FALSE
```

```

#####
# Boot Order configuration
# Each entry BOOT<N> could have the following format:
# PXE:
# BOOT<N> = NET-<NIC_P0 | NIC_P1 | OOB | RSHIM>-<IPV4 | IPV6>
# PXE over VLAN (vlan-id in decimal):
# BOOT<N> = NET-<NIC_P0 | NIC_P1 | OOB | RSHIM>[.<vlan-id>]-<IPV4 | IPV6>
# UEFI Shell:
# BOOT<N> = UEFI_SHELL
# DISK: boot entries created during OS installation.
# BOOT<N> = DISK
#####
# This example configures PXE boot over the 2nd ConnectX port.
# If fails, it continues to boot from disk with boot entries created during OS
# installation.
#BOOT0 = NET-NIC_P1-IPV4
#BOOT1 = DISK

# UPDATE_ATF_UEFI - Updated ATF/UEFI (Default: yes)
# Relevant for PXE installation only as while using RSHIM interface ATF/UEFI
# will always be updated using capsule method
UPDATE_ATF_UEFI="yes"

# UPDATE_DPU_OS - Update/Install DPU Operating System (Default: yes)
UPDATE_DPU_OS="yes"

# grub_admin_PASSWORD - Hashed password to be set for the "admin" user to
enter Grub menu
# Relevant for Ubuntu BFB only. (Default: is not set)
# E.g.:
grub_admin_PASSWORD='grub.pbkdf2.sha512.10000.5EB1FF92FDD89BDAF339517428
grub_admin_PASSWORD='grub.pbkdf2.sha512.10000.<hashed password>'

# ubuntu_PASSWORD - Hashed password to be set for "ubuntu" user during BFB
installation process.

```

```

# Relevant for Ubuntu BFB only. (Default: is not set)
ubuntu_PASSWORD=<hashed password>

#####
# BMC Component Update
#####
# BMC_USER - User name to be used to access BMC (Default: root)
BMC_USER="root"

# BMC_PASSWORD - Password used by the BMC user to access BMC (Default: None)
BMC_PASSWORD=""

# BMC_IP_TIMEOUT - Maximum time in seconds to wait for the connection to the
# BMC to be established (Default: 600)
BMC_IP_TIMEOUT=600

# BMC_TASK_TIMEOUT - Maximum time in seconds to wait for BMC task (BMC/CEC
# Firmware update) to complete (Default: 1800)
BMC_TASK_TIMEOUT=1800

# UPDATE_BMC_FW - Update BMC firmware (Default: yes)
UPDATE_BMC_FW="yes"

# BMC_REBOOT - Reboot BMC after BMC firmware update to apply the new version
# (Default: no). Note that the BMC reboot will reset the BMC console.
BMC_REBOOT="no"

# UPDATE_CEC_FW - Update CEC firmware (Default: yes)
UPDATE_CEC_FW="yes"

# UPDATE_DPU_GOLDEN_IMAGE - Update DPU Golden Image (Default: yes)
UPDATE_DPU_GOLDEN_IMAGE="yes"

# UPDATE_NIC_FW_GOLDEN_IMAGE- Update NIC firmware Golden Image (Default:
yes)
UPDATE_NIC_FW_GOLDEN_IMAGE="yes"

```

```

# pre_bmc_components_update - Shell function called by BFB's install.sh before
# updating BMC components (no communication to the BMC is established at this
# point)

# post_bmc_components_update - Shell function called by BFB's install.sh after
# updating BMC components

#####
# NIC Firmware update
#####
# WITH_NIC_FW_UPDATE - Update NIC Firmware (Default: yes)
WITH_NIC_FW_UPDATE="yes"

#####
# Other misc configuration
#####

# MAC address of the rshim network interface (tmfifo_net0).
#NET_RSHIM_MAC = 00:1a:ca:ff:ff:01

# DHCP class identifier for PXE (arbitrary string up to 32 characters)
#PXE_DHCP_CLASS_ID = NVIDIA/BF/PXE

# Create dual boot partition scheme (Ubuntu only)
# DUAL_BOOT=yes

# Upgrade NIC firmware
# WITH_NIC_FW_UPDATE=yes

# Target storage device for the DPU OS (Default SSD: /dev/nvme0n1)
device=/dev/nvme0n1

# bfb_modify_os - SHELL function called after the file system is extracted on the
target partitions.

```

```
# It can be used to modify files or create new files on the target file system mounted
under
# /mnt. So the file path should look as follows:
/mnt/<expected_path_on_target_OS>. This
# can be used to run a specific tool from the target OS (remember to add /mnt to
the path for
# the tool).

# bfb_pre_install – SHELL function called before partitions format
# and OS filesystem is extracted

# bfb_post_install – SHELL function called as a last step before reboot.
# All partitions are unmounted at this stage.
```

Default Network Interface Configuration

Network interfaces are configured using the netplan utility:

```
# cat /etc/netplan/50-cloud-init.yaml
# This file is generated from information provided by the datasource. Changes
# to it will not persist across an instance reboot. To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
  ethernets:
    tmfifo_net0:
      addresses:
        - 192.168.100.2/30
      dhcp4: false
      nameservers:
        addresses:
          - 192.168.100.1
```

```
routes:
- metric: 1025
to: 0.0.0.0/0
via: 192.168.100.1
oob_net0:
dhcp4: true
renderer: NetworkManager
version: 2

# cat /etc/netplan/60-mlnx.yaml
network:
ethernets:
enp3s0f0s0:
dhcp4: 'true'
enp3s0f1s0:
dhcp4: 'true'
renderer: networkd
version: 2
```

BlueField DPUs also have a local IPv6 (LLv6) derived from the MAC address via the STD stack mechanism. For a default MAC, 00:1A:CA:FF:FF:01, the LLv6 address would be fe80::21a:caff:feff:ff01.

For multi-device support, the LLv6 address works with SSH for any number of DPUs in the same host by including the interface name in the SSH command:

```
host]# systemctl restart rshim
// wait 10 seconds
host]# ssh -6 ubuntu@fe80::21a:caff:feff:ff01%tmfifo_net<n>
```

Note

If tmfifo_net<n> on the host does not have an LLv6 address, restart the RShim driver:

```
systemctl restart rshim
```

Ubuntu Boot Time Optimizations

To improve the boot time, the following optimizations were made to Ubuntu OS image:

```
# cat /etc/systemd/system/systemd-networkd-wait-online.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/bin/nm-online -s -q --timeout=5

# cat /etc/systemd/system/NetworkManager-wait-online.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/lib/systemd/systemd-networkd-wait-online --timeout=5

# cat /etc/systemd/system/networking.service.d/override.conf
[Service]
TimeoutStartSec=5
ExecStop=
ExecStop=/sbin/ifdown -a --read-environment --exclude=lo --force --ignore-errors
```

This configuration may affect network interface configuration if DHCP is used. If a network device fails to get configuration from the DHCP server, then the timeout value in the two files above must be increased.

Grub Configuration:

Setting the Grub timeout at 2 seconds with `GRUB_TIMEOUT=2` under `/etc/default/grub`. In conjunction with the `GRUB_TIMEOUT_STYLE=countdown` parameter, Grub will show the countdown of 2 seconds in the console before booting Ubuntu. Please note that, with this short timeout, the standard Grub method for entering the Grub menu (i.e., SHIFT or Esc) does not work. Function key F4 can be used to enter the Grub menu.

System Services:

docker.service is disabled in the default Ubuntu OS image as it dramatically affects boot time.

The kexec utility can be used to reduce the reboot time. Script `/usr/sbin/kexec_reboot` is included in the default Ubuntu 20.04 OS image to run corresponding kexec commands.

```
# kexec_reboot
```

DHCP Client Configuration

```
/etc/dhcp/dhclient.conf:  
send vendor-class-identifier "NVIDIA/BF/DP";  
interface "oob_net0" {  
    send vendor-class-identifier "NVIDIA/BF/OOB";  
}
```

Ubuntu Dual Boot Support

BlueField DPU may be installed with support for dual boot. That is, two identical images of the BlueField OS may be installed using BFB.

The following is a proposed SSD partitioning layout for 119.24 GB SSD:

```
Device Start End Sectors Size Type  
/dev/nvme0n1p1 2048 104447 102400 50M EFI System  
/dev/nvme0n1p2 104448 114550086 114445639 54.6G Linux filesystem  
/dev/nvme0n1p3 114550087 114652486 102400 50M EFI System  
/dev/nvme0n1p4 114652487 229098125 114445639 54.6G Linux filesystem  
/dev/nvme0n1p5 229098126 250069645 20971520 10G Linux filesystem
```

Where:

- `/dev/nvme0n1p1` – boot EFI partition for the first OS image

- /dev/nvme0n1p2 – root FS partition for the first OS image
- /dev/nvme0n1p3 – boot EFI partition for the second OS image
- /dev/nvme0n1p4 – root FS partition for the second OS image
- /dev/nvme0n1p5 – common partition for both OS images

For example, the following is a proposed eMMC partitioning layout for a 64GB eMMC:

```
Device Start End Sectors Size Type
/dev/mmcblk0p1 2048 104447 102400 50M EFI System
/dev/mmcblk0p2 104448 50660334 50555887 24.1G Linux filesystem
/dev/mmcblk0p3 50660335 50762734 102400 50M EFI System
/dev/mmcblk0p4 50762735 101318621 50555887 24.1G Linux filesystem
/dev/mmcblk0p5 101318622 122290141 20971520 10G Linux filesystem
```

Where:

- /dev/mmcblk0p1 – boot EFI partition for the first OS image
- /dev/mmcblk0p2 – root FS partition for the first OS image
- /dev/mmcblk0p3 – boot EFI partition for the second OS image
- /dev/mmcblk0p4 – root FS partition for the second OS image
- /dev/mmcblk0p5 – common partition for both OS images

Note

The common partition can be used to store BFB files that will be used for OS image update on the non-active OS partition.

Installing Ubuntu OS Image Using Dual Boot

Note

For software upgrade procedure, please refer to section "[Upgrading Ubuntu OS Image Using Dual Boot](#)".

Add the values below to the `bf.cfg` configuration file (see section "[bf.cfg Parameters](#)" for more information).

```
DUAL_BOOT=yes
```

If EMMC size is ≤ 16 GB, dual boot support is disabled by default, but it can be forced by setting the following parameter in `bf.cfg`:

```
FORCE_DUAL_BOOT=yes
```

To modify the default size of the `/common` partition, add the following parameter:

```
COMMON_SIZE_SECTORS=<number-of-sectors>
```

The number of sectors is the size in bytes divided by the block size (512). For example, for 10GB, the `COMMON_SIZE_SECTORS=$((10*2**30/512))`.

After assigning size for the `/common` partition, what remains is divided equally between the two OS images.

```
# bfb-install --bfb <BFB> --config bf.cfg --rshim rshim0
```

This will result in the Ubuntu OS image to be installed twice on the BlueField DPU.

Note

For comprehensive list of the supported parameters to customize `bf.cfg` during BFB installation, refer to section "[bf.cfg Parameters](#)".

Upgrading Ubuntu OS Image Using Dual Boot

1. Download the new BFB to the BlueField DPU into the `/common` partition. Use `bfb_tool.py` script to install the new BFB on the inactive BlueField DPU partition:

```
/opt/mellanox/mlnx_snap/exec_files/bfb_tool.py --op fw_activate_bfb --bfb  
<BFB>
```

2. Reset BlueField DPU to load the new OS image:

```
/sbin/shutdown -r 0
```

BlueField DPU will now boot into the new OS image.

Use `efibootmgr` utility to manage the boot order if necessary.

- Change the boot order with:

```
# efibootmgr -o
```


- Remove stale boot entries with:

```
# efibootmgr -b <E> -B
```

Where `<E>` is the last character of the boot entry (i.e., `Boot000<E>`). You can find that by running:

```
# efibootmgr
```

```
BootCurrent: 0040
Timeout: 3 seconds
BootOrder: 0040,0000,0001,0002,0003
Boot0000* NET-NIC_P0-IPV4
Boot0001* NET-NIC_P0-IPV6
Boot0002* NET-NIC_P1-IPV4
Boot0003* NET-NIC_P1-IPV6
Boot0040* focal0
...2
```

 **Note**

Modifying the boot order with `efibootmgr -o` does not remove unused boot options. For example, changing a boot order from 0001,0002,0003 to just 0001 does not actually remove 0002 and 0003. 0002 and 0003 need to be explicitly removed using `efibootmgr -B`.

© Copyright 2024, NVIDIA. PDF Generated on 06/07/2024