



Deploying BlueField Software Using ISO with PXE

Table of contents

PXE Server Setup

PXE Sequence with Redfish

BlueField software (including Ubuntu OS), NIC firmware, and BMC software can be deployed using an ISO image similarly to the standard Ubuntu deployment method using ISO. The BlueField ISO image is based on the standard Ubuntu ISO image for Arm64 with an updated kernel and added DOCA packages.

PXE Server Setup

Mount the ISO:

```
$ mount bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso /mnt
$ cp /mnt/casper/vmlinuz /var/lib/tftpboot/boot/
$ cp /mnt/casper/initrd /var/lib/tftpboot/boot/
```

Example of grub.cfg:

```
menuentry "Install BF OS" {
    linux /boot/vmlinuz autoinstall fsck.mode=skip no-snapd console=hvc0 console=ttyAMA0
    earlycon=pl011,0x13010000 net.ifnames=0 biosdevname=0 iommu.passthrough=1 ip=dhcp
    url=http://<HTTP server IP>/jammy/ISO/bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso bfnet=eth0:dhcp
    bfks=http://<HTTP server IP>/jammy/ISO/bfks
    initrd /boot/initrd
}
```

The bf.cfg file can be used to customize the installation procedure. To create bf.cfg on the BlueField to be used for the installation use the bfks parameter to point to the script located on HTTP server that will create bf.cfg file:

bfks example:

```
cat > /etc/bf.cfg << 'EOF'
BMC_PASSWORD="..."
EOF
```

Standard automatic Ubuntu installation using autoinstall.yaml is also supported. See [Introduction to autoinstall - Ubuntu installation documentation](#).

Example of autoinstall.yaml that can be used to customize the installation and modify bf.cfg:

Example of a grub.cfg with autoinstall.yaml:

```
menuentry "Install BF OS" {
  linux /boot/vmlinuz autoinstall fsck.mode=skip no-snapd console=hvc0 console=ttyAMA0
  earlycon=pl011,0x13010000 net.ifnames=0 biosdevname=0 iommu.passthrough=1 ip=dhcp
  url=http://<HTTP server IP>/jammy/ISO/bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso force-
  ai=http://<HTTP server IP>/jammy/ISO/autoinstall.yaml cloud-config-url=/dev/null
  initrd /boot/initrd
}
```

Example of autoinstall.yaml:

```
version: 1

apt:
  preserve_sources_list: false
  conf: |
    Dpkg::Options {
      "--force-confdef";
      "--force-confold";
    };

storage:
  swap:
    size: 0
  grub:
    reorder_uefi: true
  config:
    - id: nvme0n1
      type: disk
      ptable: gpt
      path: /dev/nvme0n1
      name: osdisk
      wipe: superblock-recursive

    - id: nvme0n1-part1
      type: partition
```

```
device: nvme0n1
number: 1
size: 50MB
flag: boot
grub_device: true

- id: nvme0n1-part1-fs1
  type: format
  fstype: fat32
  label: efi
  volume: nvme0n1-part1

- id: nvme0n1-part2
  type: partition
  device: nvme0n1
  number: 2
  size: -1

- id: nvme0n1-part2-fs1
  type: format
  fstype: ext4
  label: root
  volume: nvme0n1-part2

- id: nvme0n1-mount
  type: mount
  path: /
  device: nvme0n1-part2-fs1
  options: defaults
  passno: 0
  fstype: auto

- id: nvme0n1-boot-mount
  type: mount
  path: /boot/efi
  device: nvme0n1-part1-fs1
  options: umask=0077
  passno: 1

reporting:
  builtin:
    type: print

# Add user-data so that subiquity doesn't complain about us not
```

```

# having a identity section
user-data:
  debug:
    verbose: true
  write_files:
    - path: /etc/iptables/rules.v4
      permissions: '0644'
      owner: 'root:root'
      content: |
        *mangle
        :PREROUTING ACCEPT [45:3582]
        :INPUT ACCEPT [45:3582]
        :FORWARD ACCEPT [0:0]
        :OUTPUT ACCEPT [36:4600]
        :POSTROUTING ACCEPT [36:4600]
        :KUBE-IPTABLES-HINT - [0:0]
        :KUBE-KUBELET-CANARY - [0:0]
        COMMIT
        *filter
        :INPUT ACCEPT [41:3374]
        :FORWARD ACCEPT [0:0]
        :OUTPUT ACCEPT [32:3672]
        :DOCKER-USER - [0:0]
        :KUBE-FIREWALL - [0:0]
        :KUBE-KUBELET-CANARY - [0:0]
        :LOGGING - [0:0]
        :POSTROUTING - [0:0]
        :PREROUTING - [0:0]
        -A INPUT -j KUBE-FIREWALL
        -A INPUT -p tcp -m tcp --dport 111 -j REJECT --reject-with icmp-port-unreachable
        -A INPUT -p udp -m udp --dport 111 -j REJECT --reject-with icmp-port-unreachable
        -A INPUT -i lo -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -d 127.0.0.0/8 -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m state --state RELATED,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp ! --dport 22 ! --tcp-flags FIN,SYN,RST,ACK SYN -m mark --mark 0xb -m state --
state NEW -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -f -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,PSH,ACK,URG -m mark --
mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -m mark --mark 0xb -m
comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m state --state INVALID -m comment --comment MD_IPTABLES -j
DROP

```

```

-A INPUT -p tcp -m tcp --tcp-flags RST RST -m mark --mark 0xb -m hashlimit --hashlimit-above 2/sec
--hashlimit-burst 2 --hashlimit-mode srcip --hashlimit-name hashlimit_0 --hashlimit-htable-expire 30000
-m comment --comment MD_IPTABLES -j DROP
-A INPUT -p tcp -m mark --mark 0xb -m state --state NEW -m hashlimit --hashlimit-above 50/sec --
hashlimit-burst 50 --hashlimit-mode srcip --hashlimit-name hashlimit_1 --hashlimit-htable-expire 30000
-m comment --comment MD_IPTABLES -j DROP
-A INPUT -p tcp -m mark --mark 0xb -m conntrack --ctstate NEW -m hashlimit --hashlimit-above
60/sec --hashlimit-burst 20 --hashlimit-mode srcip --hashlimit-name hashlimit_2 --hashlimit-htable-
expire 30000 -m comment --comment MD_IPTABLES -j DROP
-A INPUT -m mark --mark 0xb -m recent --rcheck --seconds 86400 --name portscan --mask
255.255.255.255 --rsource -m comment --comment MD_IPTABLES -j DROP
-A INPUT -m mark --mark 0xb -m recent --remove --name portscan --mask 255.255.255.255 --
rsource -m comment --comment MD_IPTABLES
-A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --set --
name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
-A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --
update --seconds 60 --hitcount 50 --name DEFAULT --mask 255.255.255.255 --rsource -m comment --
comment MD_IPTABLES -j DROP
-A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --set --
name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
-A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --
update --seconds 60 --hitcount 10 --name DEFAULT --mask 255.255.255.255 --rsource -m comment --
comment MD_IPTABLES -j DROP
-A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --set
--name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
-A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --
update --seconds 60 --hitcount 100 --name DEFAULT --mask 255.255.255.255 --rsource -m comment --
comment MD_IPTABLES -j DROP
-A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m
comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 179 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 68 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 122 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 6306 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED
-m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 69 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT

```

```

-A INPUT -p udp -m udp --dport 389 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 389 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 1812:1813 -m mark --mark 0xb -m conntrack --ctstate
NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 49 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 49 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m
comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --sport 53 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m
comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 500 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 4500 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED
-m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 1293 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED
-m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 1293 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 1707 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED
-m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 1707 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -i lo -p udp -m udp --dport 3786 -m conntrack --ctstate NEW,ESTABLISHED -m comment --
comment MD_IPTABLES -j ACCEPT
-A INPUT -i lo -p udp -m udp --dport 33000 -m conntrack --ctstate NEW,ESTABLISHED -m comment -
comment MD_IPTABLES -j ACCEPT
-A INPUT -p icmp -m mark --mark 0xb -m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --sport 5353 --dport 5353 -m mark --mark 0xb -m conntrack --ctstate
NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 33434:33523 -m mark --mark 0xb -m comment --comment
MD_IPTABLES -j REJECT --reject-with icmp-port-unreachable
-A INPUT -p udp -m udp --dport 123 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 514 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 67 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -
m comment --comment MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 60102 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED
-m comment --comment "MD_IPTABLES: Feature HA port" -j ACCEPT
-A INPUT -m mark --mark 0xb -m comment --comment MD_IPTABLES -j LOGGING

```



```

-A FORWARD -j DOCKER-USER
-A OUTPUT -o oob_net0 -m comment --comment MD_IPTABLES -j ACCEPT
-A DOCKER-USER -j RETURN
-A LOGGING -m mark --mark 0xb -m comment --comment MD_IPTABLES -j NFLOG --nflog-prefix
"IPTables-Dropped: " --nflog-group 3
-A LOGGING -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
-A PREROUTING -i oob_net0 -m comment --comment MD_IPTABLES -j MARK --set-xmark
0xb/0xffffffff
-A PREROUTING -p tcp -m tcpmss ! --mss 536:65535 -m tcp ! --dport 22 -m mark --mark 0xb -m
contrack --ctstate NEW -m comment --comment MD_IPTABLES -j DROP
COMMIT
*nat
:PREROUTING ACCEPT [1:320]
:INPUT ACCEPT [1:320]
:OUTPUT ACCEPT [8:556]
:POSTROUTING ACCEPT [8:556]
:KUBE-KUBELET-CANARY - [0:0]
:KUBE-MARK-DROP - [0:0]
:KUBE-MARK-MASQ - [0:0]
:KUBE-POSTROUTING - [0:0]
-A POSTROUTING -m comment --comment "kubernetes postrouting rules" -j KUBE-POSTROUTING
-A KUBE-MARK-DROP -j MARK --set-xmark 0x8000/0x8000
-A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
-A KUBE-POSTROUTING -m mark ! --mark 0x4000/0x4000 -j RETURN
-A KUBE-POSTROUTING -j MARK --set-xmark 0x4000/0x0
-A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic requiring SNAT" -j
MASQUERADE --random-fully
COMMIT
users:
- name: ubuntu
lock_passwd: False
groups: adm, audio, cdrom, dialout, dip, floppy, lxd, netdev, plugdev, sudo, video
sudo: ALL=(ALL) NOPASSWD:ALL
shell: /bin/bash
plain_text_passwd: 'ubuntu'
chpasswd:
list: |
ubuntu:ubuntu
expire: True
no_ssh_fingerprints: true
runcmd:
- [ /usr/sbin/netfilter-persistent, start ]
- [ /opt/mellanox/doca/services/telemetry/import_doca_telemetry.sh ]
- [ /usr/bin/bfrshlog, "INFO: DPU is ready" ]

```

late-commands:

```
# write release file
- |
cat << EOF > /target/etc/bf-release
BF_NAME="Mellanox Bluefield"
BF_PRETTY_NAME="Mellanox Bluefield"
BF_SWBUILD_TIMESTAMP="2024-06-12-12-47-25"
BF_SWBUILD_VERSION="2.7.0085-1"
BF_COMMIT_ID="7fce146"
BF_PLATFORM="BlueField SoC"
BF_SERIAL_NUMBER="1332723060006"
EOF

# mount cdrom
- mkdir -p /target/tmp/cdrom
- mount --bind /cdrom /target/tmp/cdrom || true
- |
cat << EOF > /target/etc/apt/sources.list
deb [check-date=no] file:///tmp/cdrom/ jammy main restricted
EOF

# avoid running flash kernel after install kernel
- mkdir -p /target/run/systemd
- echo docker > /target/run/systemd/container

# Install packages
- curtin in-target -- apt update -y
- curtin in-target -- apt remove -y --purge `dpkg --get-selections | grep openipmi | awk '{print $2}'`
- curtin in-target -- /bin/bash -c "DEBIAN_FRONTEND=noninteractive RUN_FW_UPDATER=no apt-get
install --no-install-recommends -y acpid bc binutils bridge-utils build-essential cracklib-runtime dc
docker.io flash-kernel i2c-tools ifenslave iperf3 iptables-persistent iputils-arping iputils-ping iputils-
tracpath kexec-tools libpam-pwquality libssl-dev lldpad lm-sensors net-tools network-manager nfs-
common nvme-cli openssh-server python3.10 python3-pyinotify python3-pip rasdaemon rsync
sbsigntool shim-signed tcpdump watchdog doca-runtime doca-devel containerd kubelet runc nv-
common-apis nvidia-repo-keys linux-bluefield-modules-bluefield linux-image-5.15.0-1042-bluefield"

# rewrite sources
- |
cat << EOF > /target/etc/apt/sources.list
deb http://ports.ubuntu.com/ubuntu-ports/ jammy main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-updates main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-security main restricted universe multiverse
EOF
```

```
# Allow cloud-init to configure networking
- find /target/etc/cloud/cloud.cfg.d/ -type f ! -name README ! -name 05_logging.cfg ! -name 90_dpkg.cfg
-delete || true;
- curtin in-target -- cloud-init clean
```

```
# Post-installation steps
```

```
# Create bf.cfg
```

```
- |
```

```
cat << EOF > /target/etc/bf.cfg
```

```
# 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"
```

```
#####
```

```
# 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 BlueField 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: no)
WITH_NIC_FW_UPDATE="yes"
EOF

# Run post-installation script to update ATF/UEFI, NIC firmware and BMC components
- curtin in-target -- /bin/bash -c "device=/dev/nvme0n1 /usr/local/sbin/bfiso-post-install.sh || true"

- curtin in-target -- systemctl disable snapd

```

PXE Sequence with Redfish

HTTP boot configuration can be done using the BlueField BMC's Redfish interface.

ISO upgrade via Redfish to set UEFI HTTPs/PXE boot by setting UEFI first boot source.

To set the UEFI first boot source using Redfish:

1. Follow the instructions under section "[Deploying BlueField Software Using BFB with PXE](#)".
2. Check the current boot override settings by performing a GET on the ComputerSystem schema over 1GbE to the BlueField BMC. Look for the "Boot" property.

```
curl -k -X GET -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/ |  
python3 -m json.tool  
{  
...  
"Boot": {  
    "BootNext": "",  
    "BootOrderPropertySelection": "BootOrder",  
    "BootSourceOverrideEnabled": "Disabled",  
    "BootSourceOverrideMode": "UEFI",  
    "BootSourceOverrideTarget": "None",  
    "UefiTargetBootSourceOverride": "None",  
    ....  
},  
....  
"BootSourceOverrideEnabled@Redfish.AllowableValues": [  
    "Once",  
    "Continuous",  
    "Disabled"  
],  
"BootSourceOverrideTarget@Redfish.AllowableValues": [  
    "None",  
    "Pxe",  
    "UefiHttp",  
    "UefiShell",  
    "UefiTarget",  
    "UefiBootNext"  
],  
....  
}
```

Info

Boot override enables overriding the first boot source, either once or continuously.

3. The example output above shows the `BootSourceOverrideEnabled` property is Disabled and `BootSourceOverrideTarget` is None. The `BootSourceOverrideMode` property should always

be set to UEFI. Allowable values of `BootSourceOverrideEnabled` and `BootSourceOverrideTarget` are defined in the metadata (`BootSourceOverrideEnabled@Redfish.AllowableValues` and `BootSourceOverrideTarget@Redfish.AllowableValues` respectively).

4. If `BootSourceOverrideEnabled` is set to `Once`, then boot override is disabled after the first boot, and any related properties are reset to their former values to avoid repetition. If it is set to `Continuous`, then on every reboot the BlueField keeps performing boot override (`HTTPBoot`).
5. To perform boot override, perform a `PATCH` to pending settings URI over the 1GbE to the BlueField BMC.

```
curl -k -X PATCH -d '{"Boot": {"BootSourceOverrideEnabled": "Once",
"BootSourceOverrideMode": "UEFI", "BootSourceOverrideTarget": "UefiHttp",
"HttpBootUri": "http://<HTTP-Server-Ip>/Image.iso"}}' -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/Settings | python3 -m json.tool
```


For example:

```
curl -k -X GET -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/ |
python3 -m json.tool
{
...
"Boot": {
  "BootNext": "",
  "BootOrderPropertySelection": "BootOrder",
  "BootSourceOverrideEnabled": "Once",
  "BootSourceOverrideMode": "UEFI",
  "BootSourceOverrideTarget": "UefiHttp",
  "UefiTargetBootSourceOverride": "None",
  ....
},
....
}
```

6. After performing the above `PATCH` successfully, reboot the BlueField using the Redfish Manager schema over the 1GbE to the BlueField BMC:

```
curl -k -u root:<password> -H "Content-Type: application/json" -X POST https://<BF-BMC-IP>/redfish/v1/Systems/Bluefield/Actions/ComputerSystem.Reset -d '{"ResetType" : "GracefulRestart"}
```

7. Once UEFI has completed, check whether the settings are applied by performing a GET on ComputerSystem schema over the 1GbE OOB to the BlueField BMC.

 **Note**

The HttpBootUri property is parsed by the Redfish server and the URI is presented to the BlueField as part of DHCP lease when the BlueField performs the HTTP boot.

© Copyright 2024, NVIDIA. PDF Generated on 08/20/2024