



Customizing BlueField Software Deployment

Table of contents

[Changing Default Credentials for "ubuntu" User via bf.cfg](#)

[Changing UEFI Password Using bf.cfg](#)

[Changing BMC Password Using bf.cfg](#)

[Advanced Customizations During BFB Installation](#)

[bf.cfg Parameters](#)

[System Configuration Dump](#)

`bf.cfg` is an optional configuration file which may be used to customize the software deployment process on NVIDIA® BlueField® networking platforms (DPU or SuperNIC).

Note

To update the BMC components, it is required to provide the `BMC_PASSWORD` using `bf.cfg` to the BFB/ISO installation environment.

There are different ways to pass `bf.cfg` along with the BFB or ISO to customize the installation procedure:

- With BFB from the host:

```
# bfb-install -r <rshim device> -c <path to bf.cfg> -b <BFB>
```

- Using cat command:

```
# cat <BFB> <path to bf.cfg> > /dev/<rshim device>/boot
```

- By appending `bf.cfg` to the BFB and push it to RShim device on a host or BMC:

```
# cat <BFB> <path to bf.cfg> > <new BFB>
```

- In PXE environment using `bfks` parameter to provide a script that will be downloaded by the installation process and run on the Bluefield side at the beginning of installation:

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

- Or using `autoinstall.yaml`. See "[Deploying BlueField Software Using ISO with PXE](#)" for details.

Changing Default Credentials for "ubuntu" User via 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$3B0RIrfX$T1Hry93NFUJzg3Nya00rE1
```

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

```
# vim bf.cfg
ubuntu_PASSWORD=' $1$3B0RIrfX$T1Hry93NFUJzg3Nya00rE1 '
```

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

Changing UEFI Password Using `bf.cfg`

To change the UEFI password, add the current UEFI password under parameter `UEFI_PASSWORD` and define the new UEFI password under `NEW_UEFI_PASSWORD` inside the `bf.cfg` configuration file.

Changing BMC Password Using `bf.cfg`

To change the BMC root password, add the current BMC root password under parameter `BMC_PASSWORD` and define the new BMC root password under `NEW_BMC_PASSWORD` inside the `bf.cfg` configuration file.

Advanced 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 BlueField 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
=====
}

```

bf.cfg Parameters

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

```

#####
# This file contains configuration that can be pushed together to
customize
# the BFB installation. The configuration is stored as
/etc/bf.cfg which will
# be used by the 'bfcfg' or other tools as input.
#
# Uncomment the variables to customize the values as needed, Or
else the
# values will be unchanged, or the default values are used when
the variables
# are created.
# Note:
# - Try to keep at least one comment line from beginning, if
possible, when
#   pushed together with BFB;
# - Do not use spaces around the '=', it will not work for all
BFBs.
#
#####

#####
# Manufacturing information (not applicable for BlueField-1).
# !!! Note: These variables are only allowed to be programmed
once, then the
#   values will be locked. Be sure to set to correct
values when
#   customizing them.
# !!! Note: This configuration may take effect after reset.
#####
# MAC address of the OOB network interface
#MFG_OOB_MAC=00:1a:ca:11:22:33
# OPN number of this board
#MFG_OPN=MBF1M332A
# SKU ID of this board

```

```

#MFG_SKU=MBF1M332A
# Model of this board
#MFG_MODL=X
# Serial Number of this board
#MFG_SN=X
# UUID of this board
#MFG_UUID=X
# Revision
#MFG_REV=X

#####
# Manufacturing information extended (applicable for BlueField-3
only).
# !!! Note: These variables are only allowed to be programmed
once, then the
#           values will be locked. Be sure to set to correct
values when
#           customizing them.
# !!! Note: This configuration takes effect after reset.
#####
# System Manufacturer
#MFG_SYS_MFR=X
# System Product Name
#MFG_SYS_PDN=X
# System Version
#MFG_SYS_VER=X
# Baseboard Manufacturer
#MFG_BSB_MFR=X
# Baseboard Product Name
#MFG_BSB_PDN=X
# Baseboard Version
#MFG_BSB_VER=X
# Baseboard Serial Number
#MFG_BSB_SN=X

#####

```

```

# Configuration to set the platform mode (applicable for
BlueField-3 only).
# Only 'DPU' or 'NIC' are permitted values. The configuration is
applied on
# next reboot.
# !!! Note: Once the configuration is applied, a power-cycle is
required
# for this configuration to take effect.
#####
#MFG_PLAT_MODE=DPU

#####
# Control flags for platform configuration
#####

# When set to 'TRUE', the file that encapsules the configuration
# file is saved whithin a persistent storage.
# Note the file will be saved, if and only if, the configuration
# is completed successfully.
#CTL_SAVE_CONFIG_FILE=FALSE

# Reset Lanugage.
# The EFI variables 'Lang*', 'PlatformLang*' are deleted.
#CTL_RESET_LANG=TRUE

# Reset all SYS_* attributes listed below.
#CTL_RESET_SYS=FALSE

# Delete all BOOT* configurations.
#CTL_DELETE_ALL_BOOT=FALSE

# Reset all misc configurations.
#CTL_RESET_MISC=FALSE

# Delete all UEFI secure boot keys
#CTL_DELETE_ALL_UEFI_SECURE_BOOT_KEYS=TRUE

```

```

# Update Secure boot state, either 'Enabled' or 'Disabled'.
# It is relevant when the platform is in user mode, i.e.,
# PK key installed.
#CTL_UEFI_SECURE_BOOT_STATE=DISABLED

# Delete UEFI password settings.
#CTL_DELETE_UEFI_PASSWORD=TRUE

#####
# 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 (only for BF1 BlueWhale board).
#SYS_ENABLE_2ND_EMMC=FALSE

# Enable eMMC boot partition protection.
#SYS_BOOT_PROTECT=FALSE

# Enable SPCR table in ACPI.
#SYS_ENABLE_SPCR=FALSE

# Select SPCR port when SYS_ENABLE_SPCR=TRUE.
# Either 0 (Port 0) or 1 (Port 1)
#SYS_SPCR_PORT=0

```

```
# Disable PCIe in ACPI.
#SYS_DISABLE_PCIE=FALSE

# Enable OP-TEE in ACPI (obsolete).
#SYS_ENABLE_OPTEE=FALSE

# Disable the TM Fifo ACPI
#SYS_DISABLE_TMFF=FALSE

# Disable the 'force_pxe' retry behavior.
# By default, PXE boot will keep retrying all the PXE devices.
# If disabled, it'll try PXE interfaces one round then continue
the normal
# booting sequence.
#SYS_DISABLE_FORCE_PXE_RETRY=FALSE

# Disable BMC Field Mode.
#SYS_ENABLE_BMC_FIELD_MODE=FALSE

# Threshold for the correctable errors to be observed
# before reporting it to the OS.
# Supported values are 0-4294967294
#SYS_CE_THRESHOLD=5000

# Disable OS error handling via HEST
#SYS_DISABLE_HEST=FALSE

# L3 Cache partition level.
# 0: 00.0%
# 1: 12.5%
# 2: 25.0%
# 3: 37.5%
# 4: 50.0%
# 5: 62.5%
# 6: 75.0%
# 7: 87.5%
```

```

#SYS_L3_CACHE_PARTITION_LEVEL=0

# Enable I2C3 in ACPI.
#SYS_ENABLE_I2C3=FALSE

# When set to 'TRUE' UEFI will keep retrying all the bootable
# devices as long as network boot devices are present.
#SYS_ENABLE_FORCE_BOOT_RETRY=FALSE

# Enable OEM MFG configuration.
#SYS_ENABLE_OEM_MFG_CONFIG=FALSE

# Disable I2C1 in ACPI.
#SYS_DISABLE_I2C1=FALSE

# Enable UEFI wait for BMC
#SYS_ENABLE_BMC_WAIT=FALSE

# Disable the auto-refresh of UEFI boot options.
#SYS_DISABLE_AUTO_BOOT_REFRESH=FALSE

# Enable BMC network configuration menu entry.
#SYS_DISPLAY_BMC_NET_CONFIG=FALSE

# Enable Redfish Feature.
#SYS_ENABLE_REDFISH=TRUE

# Enable RTCSync.
#SYS_RTCSYNC=FALSE

#####
# Configuration to enable UEFI Secure Boot with NVIDIA default
settings on
# The signed EFI Capsule
'/lib/firmware/mellanox/boot/capsule/EnrollKeysCap'

```

```

# is used to enroll NVIDIA default certificate files and reset
UEFI password
# to default.
# !!! Note: This configuration takes effect after reset.
#####
#UEFI_SECURE_BOOT=TRUE

#####
# Partition configuration
# Multiple devices can be added here as DISK<M>_NAME.
# Each device can have multiple partitions identified by
"DISK<M>_PART<N>_xxx"
# '<N>' is the partition ID, which represents partition
"DISK<M>_NAME"p<N>.
# For example, assuming DISK0_NAME is /dev/mmcblk0. N=1 means
partition
# /dev/mmcblk0p1, N=8 means partition /dev/mmcblk0p8, etc.
#
# Each device has the following definitions. Optional attributes
are marked
# as (optional) below.
#   DISK<M>_NAME: device path, such as /dev/mmcblk0
#   DISK<M>_PART<N>_SIZE: partition <N> size in MB
#
#                               Value 0 means 'max remaining space',
which is only
#
#                               allowed on one partition.
#   DISK<M>_PART<N>_TYPE: partition <N> type, which could be
#
#                               EFI, Linux or UUID defined in
#
https://en.wikipedia.org/wiki/GUID\_Partition\_Table
#   DISK<M>_PART<N>_MOUNT: mount path
#   DISK<M>_PART<N>_PERSIST: CREATE | KEEP (optional)
#
#                               CREATE: create or overwrite this
partition
#
#                               KEEP:   keep this partition, or
create if not exist

```

```

#                                     Only one partition could be marked
as 'persist'.
#####
#DISK0_NAME=/dev/mmcblk0
#DISK0_PART1_SIZE=150
#DISK0_PART1_TYPE=EFI
#DISK0_PART1_MOUNT=/boot/efi
#DISK0_PART2_SIZE=0
#DISK0_PART2_TYPE=Linux
#DISK0_PART2_MOUNT=/
#DISK0_PART8_PERSIST=CREATE
#DISK0_PART8_SIZE=250
#DISK0_PART8_TYPE=Linux

#####
# Boot Order configuration
# Each entry BOOT<N> could have the following format:
# PXE or HTTP:
#   BOOT<N>=NET-<NIC_P0 | NIC_P1 | OOB | RSHIM>-<IPV4 | IPV6>[-
HTTP]
# PXE over VLAN (vlan-id in decimal):
#   BOOT<N>=NET-<NIC_P0 | NIC_P1 | OOB | RSHIM>[.<vlan-id>]-<IPV4
| IPV6>[-HTTP]
# UEFI Shell:
#   BOOT<N>=UEFI_SHELL
# DISK: boot entries created during OS installation.
#   BOOT<N>=DISK
#
# Additional BOOT<N>_* parameters may be added to complement the
# Boot Order configuration. These optional field parameters are:
#   BOOT<N>_DESC      : boot description string (single/double
quoted).
#       BOOT2_DESC='ubuntu'
#   BOOT<N>_DEVPATH  : boot device path string (single/double
quoted).
#       Full device path, applicable to any boot option.

```

```

#
BOOT0_DEVPATH='PciRoot(0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)
#   File path, applicable to DISK boot options.
#       BOOT2_DEVPATH='\\EFI\\ubuntu\\shimaa64.efi'
#   BOOT<N>_ARGS      : boot arguments string (single/double
quoted).
#       Boot arguments are for future use.
#####
# In the example below, BOOT0 is use for PXE boot over the 2nd
ConnectX port.
# BOOT1 is for HTTP boot over the 2nd ConnectX port. If both
BOOT0 and BOOT1
# fail, it continues to boot from disk with boot entries created
during OS
# installation.
#BOOT0=NET-NIC_P1-IPV4
#BOOT1=NET-NIC_P1-IPV4-HTTP
#BOOT2=DISK

#####
# Other misc configuration
#####
# MAC address of the rshim network interface.
#NET_RSHIM_MAC=00:1a:ca:ff:ff:01

# DHCP class identifier for PXE (arbitrary string up to 32
chracters)
# If FACTORY_DEFAULT_DHCP_BEHAVIOR is TRUE (or not specified),
default value
# 'NVIDIA/BF/PXE' will be configured during BFB installation. Or
else it'll
# take the value configured in 'PXE_DHCP_CLASS_ID'.
#FACTORY_DEFAULT_DHCP_BEHAVIOR=FALSE
#PXE_DHCP_CLASS_ID=NVIDIA/BF/PXE

# DHCP IPv6 DUID configuration for network boot.

```

```

# Permitted values are 'UUID' or 'LLT'.
#NET_DHCP_IPV6_DUID=UUID

# Version of the BF Bundle. The version can be obtained through
# 'bfver' command line.
#BF_BUNDLE_VERSION=bf-bundle-2.7.0-27_24.04_ubuntu-22.04_prod

#####
# BlueField Arm platform firmware Update
#####
# 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"

#####
# BlueField Arm OS Update
#####
# UPDATE_DPU_OS - Update/Install BlueField Operating System
(Default: yes)
#UPDATE_DPU_OS="yes"

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

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

# On some operating systems, the partition UUID may change after
the first
# boot following installation. To mitigate this, enable
FSTAB_USE_DEV_NAME=yes
# to use device names instead of UUIDs. (Default: no)
#FSTAB_USE_DEV_NAME=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.sha512.10000.5EB1FF92FDD89BDAF3395
#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)
# Use openssl to set the password for the ubuntu user:
#
# $ openssl passwd -1
# Password:
# Verifying - Password:
# $1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1
#
#ubuntu_PASSWORD='$1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1'

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

# Force NIC firmware update even if the currently installed
version matches the provided
# version. (Default: no)
#FORCE_NIC_FW_UPDATE=no

# Reset NIC firmware after flashing to apply the new firmware
version. (Default: yes)
#NIC_FW_RESET="yes"

```

```

# Perform a level-3 NIC firmware reset if a level-0 reset was
attempted and failed. (Default: no)
#FORCE_NIC_FW_RESET="no"

#####
# 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=""

# NEW_BMC_PASSWORD - can be used to change BMC_PASSWORD to the
new one (Default: None)
# Note: current BMC_PASSWORD is required
#NEW_BMC_PASSWORD=<new 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"

# UPDATE_CERTIFICATES - Update Nvidia certificates (Default: yes)
# PEM certificates uploaded to the BMC only once.
#UPDATE_CERTIFICATES="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

# Clear the RSHIM log buffer on the DPU BMC before starting the
BMC component update
# process to ensure all messages are captured in the log.
(Default: yes)
#RESET_BMC_RSHIM_LOG="yes"

#####
# Hook function to customize the BFB installation
#####

```

```
# bfb_modify_os()
# - SHELL function called after file the 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 EMMC partitions format
#   and OS filesystem is extracted.

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

System Configuration Dump

The `bfcfg` script included with the BlueField Arm OS can be used to dump system configuration information modified with `bf.cfg` or through the UEFI menu. The `-d` parameter can be used to enable dump mode and the `-l` parameter used to set the dump level (how much configuration information is logged).

To perform a full system configuration dump, run:

```
# bfcfg -d -l 2
```

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 (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes 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 NONINFRINGEMENT, 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 and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2026, NVIDIA. PDF Generated on 02/28/2026