CDW Documentation

NVIDIA GPU Firmware Upgrade Runbook (Revised)

NVIDIA GPU Firmware Upgrade Runbook (Revised)

Process at a Glance

1. **Prepare** — record current versions, verify GPU health, stop all workloads
2. **Validate Package** — checksum the firmware file, preview what will change
3. **Phase 2 (in this exact order):**
    1. BMC firmware update
    2. BMC reset (mc reset cold)
    3. Motherboard tray update (SBIOS, CPLD, PSU, etc.)
    4. GPU tray update (VBIOS, NVSwitch, EROT, FPGA)
    5. Poll BackgroundCopyStatus until **Completed**
    6. Chassis power cycle (graceful, via Redfish/ipmitool)
    7. Re-check versions on staged components
    8. CommitImage on **HostBMC_0** and **HostBIOS_0**
    9. **AC power cycle** (full PDU/cord removal, 5–6 min)
    10. NIC firmware via `mlxfwmanager` on the host OS; TPM, NVMe via BMC (last — independent of the EROT/CPLD chain)
4. **Validate** — confirm new versions, GPU health, NVLink, smoke test

<note warning> Flashing the wrong firmware file or interrupting mid-flash can permanently brick a GPU. Read the full runbook before executing.

Critical ordering: BMC must be updated and reset *before* the motherboard and GPU trays so the BMC speaks the new Redfish schema used by the rest of the bundle. Skipping the AC power cycle leaves EROT/CPLD images staged but not activated. </note>

---

1. Overview

This runbook covers the end-to-end firmware upgrade process for NVIDIA DGX/HGX H100/H200 systems, GB200/B200 (Blackwell) systems, and standalone PCIe GPUs.

Scope — what this runbook covers:

- **Hopper:** DGX H100/H200, HGX H100/H200
- **Blackwell:** DGX B200, HGX B200, GB200 NVL72 compute tray (per-tray BMC), GB300
- **Standalone PCIe GPUs** via NVFlash
- Components: VBIOS, NVSwitch, EROT, FPGA, BMC, SBIOS, CPLD, PSU, NIC, TPM, NVMe, and (Blackwell only) Grace CPU FW, NVLink Switch tray FW, ConnectX-7/8 onboard NICs

Blackwell-specific notes (GB200 / B200):

- GB200 NVL72 racks are driven from the **NVIDIA Mission Control** rack-firmware tooling. Refer to the Mission Control rack-firmware document shipped with the bundle — section 7.1B below

summarises the flow but does **not** replace it.
- Bundles arrive as **two NVIDIA-released packages** plus the NVLink Switch tray package: `nvfw_BMC_<P-number>.fwpkg` and `nvfw_Compute_<P-number>.fwpkg` (Mission Control names them by NVIDIA "P-number" — e.g., **P4972** = early Blackwell baseline, **P4975** = first GA refresh, **P4978** = current — and selector JSONs are named `BMC_Full.json` and `Compute_Full.json`).
- **Ordering, simplified:** chassis BMC (BMC_Full) → **AC cycle between BMC and HMC updates** → HMC + GPU/Grace complex (Compute_Full) → NVLink Switch tray → final rack-level AC cycle for EROT/CPLD. The AC cycle in the middle is required so the new chassis BMC re-enumerates the HMC under the updated Redfish schema before Compute_Full is pushed.
- **Activation:** Mission Control uses the documented activation method `activate_fw -c` RF_AUX_PWR_CYCLE (auxiliary-rail power cycle via Redfish), which replaces the ad-hoc "`chassis power cycle`" step that Hopper uses.
- The final mandatory **rack-level AC cycle** for EROT/CPLD still applies — for GB200 NVL72 this means power-cycling the full rack PDU feeds (coordinate with DC ops).
- **nvfwupd floor:** 2.0.4 for Hopper; **2.0.5 for Blackwell** (HMC + Grace + NVLink Switch tray Redfish paths), or **2.0.9** if you intend to run parallel multi-package updates across the rack.

Scope — what this runbook does NOT cover:

- DGX A100 (use the Firmware Update Container — separate runbook)
- Standalone Mellanox/ConnectX adapter firmware in non-DGX hosts (use `mlxfwmanager` separately)
- Quantum/Spectrum InfiniBand or Ethernet switch firmware
- Storage controller / RAID HBA firmware

---

2. Background

Components Updated

Component	What It Controls	Activation Trigger
BMC	Out-of-band server management	BMC reset (mc reset cold)
SBIOS / UEFI	Motherboard initialisation	Chassis power cycle + CommitImage
CPLD	Board-level glue logic, power sequencing	**AC power cycle (mandatory)**
VBIOS	GPU boot, clocks, power limits, PCIe config	Chassis power cycle
NVSwitch FW	GPU-to-GPU interconnect (NVLink fabric)	Chassis power cycle
EROT	Hardware root of trust / secure boot	**AC power cycle (mandatory)**
FPGA	Power management and control plane	Chassis power cycle
PSU	Power supply unit firmware	Live, one at a time — redundant PSU carries load. Refuses to flash if redundancy is unhealthy.
NIC / ConnectX	Onboard NIC firmware (incl. ConnectX-7/8 on Blackwell)	Chassis power cycle (run AFTER the AC cycle in this runbook)
TPM	Trusted Platform Module	Chassis power cycle (run AFTER the AC cycle in this runbook)

Component	What It Controls	Activation Trigger
NVMe	Onboard storage firmware	Chassis power cycle (run AFTER the AC cycle in this runbook)

<note important> EROT and CPLD images are staged into a secondary flash region by the update, but only activate on a full AC power removal — a warm reboot or chassis power cycle is not sufficient. </note>

Driver Compatibility

The minimum driver depends on the **firmware package version**, not the GPU architecture alone. Refer to the **per-package release notes** shipped with the .fwpkg you intend to install. Do not rely on generic architecture → driver tables; they go stale quickly as new packages add features and ABI requirements.

NVSwitch Chip vs NVLink Switch Tray — Do Not Confuse

- **NVSwitch chip** — silicon on the GPU baseboard. Inventory path: `/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_NVSwitch_<n>`. Updated via the compute tray BMC as part of the GPU tray package.
- **NVLink Switch tray** — separate rack-level switch chassis in GB200 NVL72 racks. Inventory path: `/redfish/v1/UpdateService/FirmwareInventory/NVSwitch_Tray_*`. Updated via the **NVLink Switch tray BMC**, not the compute tray BMC, with its own .fwpkg.

Selecting the wrong target will return a Redfish 404 and silently skip the component.

Time Budget (typical per-node)

Phase	Estimated Time
Pre-upgrade preparation (Phase 1)	20 min
BMC update + reset + verify (2.1–2.2)	10 min
Motherboard tray update (2.3)	15 min
GPU tray update (2.4)	25 min
BackgroundCopyStatus poll (2.5)	10–20 min
Chassis power cycle + recheck (2.6–2.7)	10 min
CommitImage (2.8)	2 min
AC power cycle (2.9, includes 6m drain)	20 min
NIC / TPM / NVMe + chassis cycle (2.10)	15 min
Validation (Phase 3)	15 min
Total — Hopper DGX/HGX	**~2h 30m**
GB200 NVL72 rack (add HMC, NVLink Switch tray, rack PDU cycle 10m+15m boot)	**add 1h 15m**

Schedule a **3-hour window for Hopper**, **4-hour window for Blackwell B200**, **5-hour window for GB200 NVL72**.

---

3. Tools

Tool Overview

Tool	Purpose	Where It Runs
nvfwupd	Full firmware bundle update for DGX/HGX via BMC Redfish API	Management host (remote)
NVFlash	VBIOS-only flash for standalone PCIe GPUs	GPU host (root required)
nvidia-smi	Query GPU health, firmware version, running processes	GPU host
nvsm	DGX system-wide health check	GPU host
ipmitool	Remote power cycle and BMC reset	Management host
curl	Redfish polling (BackgroundCopyStatus, CommitImage)	Management host

nvfwupd Version Requirement

- **Hopper (H100/H200) minimum: nvfwupd 2.0.4** — earlier versions do not support CommitImage or the inventory paths used below.
- **Blackwell (B200 / GB200 / GB300) minimum: nvfwupd 2.0.5** — required for HMC, Grace CPU, and NVLink Switch tray Redfish paths. Use **2.0.9** if you plan to push multiple packages in parallel from the rack management host (BMC_Full + Compute_Full).
- Confirm with: `nvfwupd –version`
- Download from the NVIDIA Enterprise Support Portal.

Key Commands Reference

```
# --- nvfwupd ---
nvfwupd --version                                          # MUST be >=
2.0.4

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version -p
<pkg.fwpkg>

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> update_fw -p <pkg.fwpkg>
-y
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> update_fw -p <pkg.fwpkg>
-y -s selector.json

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> force_update enable
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> force_update status
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> force_update disable

# Blackwell / Mission Control activation (replaces ipmitool mc reset cold)
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> servertype=GB200 \
  activate_fw -c RF_AUX_PWR_CYCLE

# --- mlxfwmanager (host OS, Hopper NIC updates) ---
sudo mlxfwmanager --query
```

```
sudo mlxfwmanager -u -i fw-ConnectX7-rel-<version>.bin --yes

# --- NVFlash ---
sudo nvflash --list
sudo nvflash --version
sudo nvflash --index=0 --save backup_$(date +%Y%m%d).rom
sudo nvflash --index=0 new_firmware.rom

# --- nvidia-smi ---
nvidia-smi -L
nvidia-smi -q
nvidia-smi -q | grep "VBIOS Version"
nvidia-smi --query-compute-apps=pid,name,used_memory --format=csv
nvidia-smi nvlink -s -i 0

# --- ipmitool ---
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> chassis power cycle
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> mc reset cold
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> mc info
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> sel list

# --- nvsm ---
nvsm show health
```

---

4. Prerequisites

<note important> Complete every item below before starting. Do not proceed if any item cannot be confirmed. </note>

Access & Credentials

- [ ] BMC IP address, username, and password (administrator privileges)
- [ ] SSH access to the target server
- [ ] **Physical or remote-hands access for the AC power cycle step**
- [ ] Access to NVIDIA Enterprise Support Portal

Tooling

- [ ] **nvfwupd ≥ 2.0.4** installed on management host
- [ ] ipmitool, curl, jq available on management host

Firmware Package

- [ ] Correct .fwpkg for your specific platform (GPU tray + motherboard tray packages)
- [ ] SHA256 checksum verified against release notes

- [ ] Per-package release notes reviewed for driver minimums and known issues
- [ ] **Previous .fwpkg saved for rollback**

BMC Version Hop (pre-1.1.3 systems)

- [ ] If current BMC firmware is **v1.0.0 or v1.1.1**, you **must** stage to **v1.1.3** first, reset BMC, and verify, before attempting **v24.09.1** (or any later bundle). The full hop is **v1.0.0 / v1.1.1 → v1.1.3 → v24.09.1**. Skipping the intermediate hop requires a **factory reset** of the BMC to recover.
- [ ] Confirm current BMC version (pin to the BMC Redfish path, not a substring match):

```
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version \
  | grep -E '(^|\s)HostBMC_0\b'
```

Environment

- [ ] All GPU workloads stopped
- [ ] `nvidia-smi` is not running (passive queries block VBIOS updates)
- [ ] Maintenance window: **3h Hopper / 4h B200 / 5h GB200 NVL72** (see Time Budget in Section 2)
- [ ] VBIOS backed up via NVFlash (standalone PCIe only)
- [ ] **MIG disabled** on all GPUs: `sudo nvidia-smi -mig 0` (some packages refuse to flash with MIG instances active)
- [ ] **nvidia-fabricmanager stopped**: `sudo systemctl stop nvidia-fabricmanager` (DGX/HGX only; prevents dirty shutdown during chassis cycle)
- [ ] **Confidential Computing (CC) mode disabled** if previously enabled — some .fwpkg versions refuse VBIOS flash while CC is on. Check the package release notes.
- [ ] **PSU redundancy healthy**: `nvsm show health | grep -i psu` (all PSUs OK; PSU update will refuse if redundancy is degraded)
- [ ] **Driver minimum verified against this package's release notes**: `nvidia-smi –query-gpu=driver_version –format=csv,noheader` ≥ the minimum listed in the .fwpkg release notes
- [ ] **Pre-upgrade SEL captured**: `ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> sel list > preupgrade_sel_$(date +%Y%m%d).log`

System State

- [ ] Management host can ping the BMC IP
- [ ] Server on stable power
- [ ] No other firmware operations running on the same server
- [ ] `nvsm show health` clean

Rollback Readiness

- [ ] Current firmware versions recorded in change ticket
- [ ] Previous firmware .fwpkg saved

- [ ] Rollback procedure reviewed (Section 9)

---

5. Best Practices

#	Practice	Why It Matters
1	Always run `show_version` first	Confirms exactly what will change
2	Back up VBIOS before flashing	No backup = no rollback path
3	Stop all GPU processes including `nvidia-smi`	Active processes block or corrupt VBIOS updates
4	Never interrupt mid-flash	Closing terminal or losing power bricks the GPU
5	Verify the package checksum	A corrupted .fwpkg silently fails
6	Wait for BackgroundCopyStatus: Completed	Rebooting early means firmware never activates
7	**Always AC-cycle after CommitImage**	EROT and CPLD only activate on full AC removal
8	Use a wired management network	VPN or WiFi drops leave flash incomplete
9	Disable ForceUpdate immediately after downgrade	Leaving it enabled is a security risk
10	Update BMC first, everything else after	Subsequent components depend on the new BMC Redfish schema
11	Validate before restoring workloads	Confirms upgrade succeeded
12	Document pre and post versions	Required for change management

---

6. Phase 1 — Pre-Upgrade Preparation

Step 1.1 — Record Current Firmware Version

```
# Pin the full pre-upgrade output to a file for the change ticket
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version \
  | tee preupgrade_versions_$(date +%Y%m%d_%H%M).txt

# Standalone PCIe GPU
nvidia-smi -q | grep "VBIOS Version" | tee preupgrade_vbios_$(date
+%Y%m%d).txt
```

Copy full output to your change ticket and fill in:

Component	Pre-Upgrade Version	Target Version
BMC		
SBIOS		
CPLD		
VBIOS		
NVSwitch FW		
EROT FW		

Component	Pre-Upgrade Version	Target Version
FPGA FW		
PSU		
NIC		
TPM		
NVMe		

Step 1.2 — Verify GPU Health

```
nvidia-smi -L
nvsm show health
```

Step 1.3 — Stop All GPU Workloads

```
kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-data
nvidia-smi --query-compute-apps=pid,name,used_memory --format=csv
# Output must be empty
```

Step 1.3b — Disable MIG and Stop Fabric Manager

```
# Disable MIG on all GPUs (no-op if already disabled)
sudo nvidia-smi -mig 0

# Stop fabric manager (DGX/HGX/Blackwell) — prevents a dirty shutdown
# when the chassis is power-cycled in Phase 2.
sudo systemctl stop nvidia-fabricmanager
systemctl is-active nvidia-fabricmanager   # must report "inactive"

# Confidential Computing — disable if currently enabled and the package
# release notes require it. (Skip if your environment never uses CC mode.)
# nvidia-smi conf-compute -srs 0
```

Step 1.4 — Backup VBIOS (Standalone PCIe Only)

```
sudo nvflash --index=0 --save gpu0_backup_$(date +%Y%m%d).rom
```

Step 1.5 — Verify Package Integrity

```
sha256sum nvfw_DGX-HGX-H100-H200x8_<version>.fwpkg
```

Step 1.6 — Preview What Will Change

```
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version -p
```

```
<pkg.fwpkg>
```

---

7. Phase 2 — Firmware Upgrade Execution

<note warning> **Strict ordering — do not deviate.** BMC first, then motherboard tray, then GPU tray, then poll, then chassis power cycle, then CommitImage, then AC cycle, then NIC/TPM/NVMe. </note>

7.1 DGX / HGX — nvfwupd

Step 2.1 — Update BMC

Use the documented one-shot BMC selector. Do not hand-enumerate child paths — let nvfwupd resolve them from the package.

```
cat > selector_bmc.json <<'EOF'
{
  "Targets": [
    "/redfish/v1/UpdateService/FirmwareInventory/HostBMC_0"
  ]
}
EOF

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> \
  update_fw -p nvfw_DGX_<version>.fwpkg -y -s selector_bmc.json
```

Step 2.2 — Reset the BMC

```
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> mc reset cold
sleep 180
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> mc info

# Pin the recheck to the BMC Redfish path (not a substring match on "bmc",
# which also matches HGX_FW_BMC_* aggregate entries on some packages).
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version \
  | grep -E '(^|\s)HostBMC_0\b'
```

Confirm the BMC reports the new version before continuing.

Step 2.3 — Update Motherboard Tray (SBIOS, CPLD, PSU)

The motherboard tray uses the **empty selector** ({}), which tells nvfwupd to apply every motherboard-side component the package declares. Do not hand-enumerate `HostBIOS_0` /

CPLDMB_0 / PSU_* — the path names vary by SKU and package, and a typo silently skips the component.

```
cat > selector_mb.json <<'EOF'
{}
EOF

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> \
  update_fw -p nvfw_DGX_<version>.fwpkg -y -s selector_mb.json
```

Cross-check beforehand with `show_version -p <pkg.fwpkg>` to confirm which components the package will touch.

Step 2.4 — Update GPU Tray (VBIOS, NVSwitch, EROT, FPGA)

The GPU tray uses the **HGX_0 one-shot selector**, which targets the entire GPU baseboard inventory in a single call. The BMC fans out internally to all GPUs, NVSwitch chips, EROT, and FPGA — do not enumerate per-GPU paths.

```
cat > selector_gpu.json <<'EOF'
{
  "Targets": [
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_0"
  ]
}
EOF

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> \
  update_fw -p nvfw_DGX-HGX-H100-H200x8_<version>.fwpkg -y -s
selector_gpu.json
```

Verify the exact child paths your platform exposes with `nvfwupd … show_version` on a real DGX H100/H200 before relying on inventory names elsewhere in this runbook — they can differ between H100 and H200 chassis SKUs.

Step 2.5 — Wait for Background Copy to Complete

```
curl -k -u <USER>:<PASS> \
  https://<BMC_IP>/redfish/v1/Chassis/HGX_ERoT_BMC_0
```

Poll every 60 seconds until `BackgroundCopyStatus` reports `Completed`. If `Failed` — do **not** proceed; see Troubleshooting.

Step 2.6 — Chassis Power Cycle

```
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> chassis power cycle
```

Wait 5–10 minutes for the server to fully boot.

Step 2.7 — Recheck Versions

```
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version
```

Confirm staged versions are visible on BMC, SBIOS, GPU tray components.

Step 2.8 — CommitImage (BMC and SBIOS)

CommitImage promotes the staged image to the active slot. **Required for BMC and SBIOS.**

```
# Commit BMC
curl -k -u <USER>:<PASS> -X POST \
  -H "Content-Type: application/json" \
  -d '{"Targets":["/redfish/v1/UpdateService/FirmwareInventory/HostBMC_0"]}'
\
https://<BMC_IP>/redfish/v1/UpdateService/Actions/Oem/NvidiaUpdateService.Co
mmitImage

# Commit SBIOS
curl -k -u <USER>:<PASS> -X POST \
  -H "Content-Type: application/json" \
  -d
'{"Targets":["/redfish/v1/UpdateService/FirmwareInventory/HostBIOS_0"]}' \
https://<BMC_IP>/redfish/v1/UpdateService/Actions/Oem/NvidiaUpdateService.Co
mmitImage
```

Step 2.9 — AC Power Cycle (MANDATORY)

<note warning> This step is **non-negotiable** for EROT and CPLD activation. A chassis power cycle is **not** sufficient. </note>

1. Power off the host via OS shutdown or `ipmitool chassis power off`.
2. **Physically remove all AC power cords** (or open the PDU breakers feeding the chassis).
3. **Wait 5–6 minutes** to allow standby rails to fully discharge.
4. Reconnect AC power and power the server back on.
5. Wait 5–10 minutes for full boot.

Step 2.10 — Update NIC (mlxfwmanager), TPM, NVMe

These components are independent of the GPU/EROT activation chain and are updated last.

Hopper (DGX/HGX H100/H200) — NICs via host-OS mlxfwmanager, not Redfish. On Hopper the ConnectX adapters are not exposed under the HGX BMC's firmware inventory; the BMC will return 404 / “no such target” for HGX_FW_NIC_* paths. Update them from the host OS instead:

```
# Inventory and current versions on the host OS
sudo mlxfwmanager --query

# Apply the bundled MFT firmware image (matches your ConnectX SKU)
sudo mlxfwmanager -u -i fw-ConnectX7-rel-<version>.bin --yes

# Re-query to confirm the new FW Version is present and "Status: Done"
sudo mlxfwmanager --query
```

A subsequent chassis power cycle (or PCIe FLR via `mlxfwreset`) is required to activate the new NIC firmware — do this **after** the AC cycle in Step 2.9.

TPM and NVMe remain BMC-driven on Hopper. Confirm the exact inventory paths on your chassis with `nvfwupd … show_version` before flashing — names like `TPM_0` / `NVMe_0` (without the `HGX_FW_` prefix) appear on some packages.

```
# After confirming the actual inventory names on your chassis:
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> \
  show_version | grep -Ei '(tpm|nvme)'

# Then build a selector_tpm_nvme.json with the paths returned above and:
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> \
  update_fw -p nvfw_DGX_<version>.fwpkg -y -s selector_tpm_nvme.json
```

Blackwell (B200 / GB200) — also update onboard ConnectX-7/8 NICs:

```
cat > selector_periph_bw.json <<'EOF'
{
  "Targets": [
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_ConnectX_0",
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_ConnectX_1",
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_ConnectX_2",
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_ConnectX_3",
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_TPM_0",
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_FW_NVMe_0"
  ]
}
EOF

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> servertype=GB200 \
  update_fw -p nvfw_GB200_<version>.fwpkg -y -s selector_periph_bw.json
```

Then host reboot:

```
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> chassis power cycle
```

7.1B Blackwell — GB200 / B200 (Mission Control flow)

<note important> This subsection summarises the **NVIDIA Mission Control rack-firmware**

document that ships with the Blackwell bundle. Cross-check every selector path and package P-number against the Mission Control doc for your bundle version before executing. The component path names below are the documented Mission Control names — they are not the same as the Hopper HGX_FW_* paths and must not be cross-pollinated. </note>

Blackwell delivers two top-level packages plus the NVLink Switch tray package:

Package name	Selector JSON	P-number examples
`nvfw_BMC_<P-number>.fwpkg`	`BMC_Full.json`	P4972 (baseline), P4975 (GA refresh), P4978 (current)
`nvfw_Compute_<P-number>.fwpkg`	`Compute_Full.json`	matched P-number to BMC package
`nvfw_NVLSwitch_<P-number>.fwpkg`	`NVLSwitch_Full.json`	(rack-level NVLink Switch tray)

Ordering (per Mission Control):

1. **B1.** Push `BMC_Full` to the **chassis BMC**.
2. **B2.** `activate_fw -c RF_AUX_PWR_CYCLE` on the chassis BMC.
3. **B3. AC power cycle between BMC and HMC updates** (rack PDU cycle for NVL72; chassis cord pull for standalone B200). Wait 5 min for standby drain. This is required so the new chassis BMC re-enumerates the HMC under the updated Redfish schema before Compute_Full is pushed.
4. **B4.** Push `Compute_Full` (HMC + Grace CPU FW + GPU/NVSwitch/EROT/FPGA on the compute tray).
5. **B5.** `activate_fw -c RF_AUX_PWR_CYCLE` for the Compute_Full payload.
6. **B6.** Push the NVLink Switch tray package from the rack-level management host against the NVLink Switch tray BMC (not the compute tray BMC).
7. **B7.** Final **rack-level AC cycle** (Step 2.9 / 2.9b) to activate EROT and CPLD on every tray.

Step B1 — Push BMC_Full to the chassis BMC

The Mission Control selector JSON is a one-shot; do not enumerate child components.

```
cat > BMC_Full.json <<'EOF'
{
  "Targets": [
    "/redfish/v1/UpdateService/FirmwareInventory/HostBMC_0"
  ]
}
EOF

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> servertype=GB200 \
  update_fw -p nvfw_BMC_P4978.fwpkg -y -s BMC_Full.json
```

Step B2 — Activate the chassis BMC payload

Blackwell uses the documented Mission Control activation verb, not `ipmitool mc reset cold`:

```
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> servertype=GB200 \
```

```
  activate_fw -c RF_AUX_PWR_CYCLE
```

Step B3 — AC cycle between BMC and HMC updates

This is **not** the final EROT/CPLD AC cycle — it sits between the two halves of the bundle and is mandatory.

1. Standalone B200: chassis cord pull, 5 min drain.
2. GB200 NVL72: rack-level PDU cycle (both A-side and B-side feeds), 10 min drain, 15 min boot.
3. Confirm the chassis BMC comes back on the new firmware before proceeding: `nvfwupd … show_version | grep -E '(^|\s)HostBMC_0\b'`.

Step B4 — Push Compute_Full

`Compute_Full` bundles HMC, Grace CPU FW, GPU/NVSwitch/EROT/FPGA for the compute tray.

```
cat > Compute_Full.json <<'EOF'
{
  "Targets": [
    "/redfish/v1/UpdateService/FirmwareInventory/HGX_0"
  ]
}
EOF

nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> servertype=GB200 \
  update_fw -p nvfw_Compute_P4978.fwpkg -y -s Compute_Full.json
```

Step B5 — Activate Compute_Full

```
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> servertype=GB200 \
  activate_fw -c RF_AUX_PWR_CYCLE
```

Step B6 — NVLink Switch tray (GB200 NVL72)

Run from the rack-level management host against the NVLink Switch tray BMC. Confirm the exact tray-side selector path names from the Mission Control NVLink-Switch addendum for your bundle — they have changed between P-numbers.

```
nvfwupd -t ip=<NVL_SW_BMC_IP> user=<USER> password=<PASS>
servertype=NVL_SWITCH \
  update_fw -p nvfw_NVLSwitch_P4978.fwpkg -y -s NVLSwitch_Full.json

nvfwupd -t ip=<NVL_SW_BMC_IP> user=<USER> password=<PASS>
servertype=NVL_SWITCH \
  activate_fw -c RF_AUX_PWR_CYCLE
```

Step B7 — Final rack-level AC cycle (GB200 NVL72)

For GB200 NVL72 the final AC step is a **rack-level PDU cycle**, not a single chassis cord pull. Coordinate with DC operations:

1. Schedule a full rack power-down window.
2. Open both A-side and B-side PDU feeds (the rack is fed redundantly — pulling only one side leaves standby rails energised).
3. Wait **10 minutes** (longer than the 5–6 min for standalone DGX due to BBU/standby capacitance).
4. Restore power; allow 15 minutes for the full rack to come up before validation.

7.2 Standalone PCIe GPU — NVFlash

Step 2.11 — Unload the NVIDIA Driver

```
sudo rmmod nvidia_uvm nvidia_drm nvidia_modeset nvidia
lsmod | grep nvidia
```

Step 2.12 — Flash the VBIOS

```
sudo nvflash --index=0 new_vbios_firmware.rom
```

Step 2.13 — Reboot

```
sudo reboot
```

---

8. Phase 3 — Validation

Step 3.1 — Confirm New Firmware Version

```
nvfwupd -t ip=<BMC_IP> user=<USER> password=<PASS> show_version
nvidia-smi -q | grep "VBIOS Version"
```

Every component must report the new version. Pay particular attention to **EROT** and **CPLD** — if either still reports the old version, the AC power cycle was not effective; repeat Step 2.9.

Step 3.2 — Verify GPU Detection

```
nvidia-smi -L
```

Step 3.3 — Run Health Check

```
nvidia-smi -q
nvsm show health
```

Step 3.4 — Check NVLink

```
nvidia-smi nvlink -s -i 0
```

Step 3.5 — Smoke Test

```
python3 -c "import torch; print(torch.cuda.is_available());
print(torch.cuda.get_device_name(0))"
```

Step 3.6 — Record Final State

Component	Pre-Upgrade	Post-Upgrade	Status
BMC			Pass / Fail
SBIOS			Pass / Fail
CPLD			Pass / Fail
VBIOS			Pass / Fail
NVSwitch FW			Pass / Fail
EROT FW			Pass / Fail
FPGA FW			Pass / Fail
PSU			Pass / Fail
NIC			Pass / Fail
TPM			Pass / Fail
NVMe			Pass / Fail

Step 3.7 — Capture Post-Upgrade SEL and Restart Services

```
# Diff against the pre-upgrade SEL captured in Section 4
ipmitool -I lanplus -H <BMC_IP> -U <USER> -P <PASS> sel list \
  > postupgrade_sel_$(date +%Y%m%d).log
diff preupgrade_sel_*.log postupgrade_sel_*.log

# Restart fabric manager and confirm healthy
sudo systemctl start nvidia-fabricmanager
sudo systemctl status nvidia-fabricmanager   # must be active (running)

# Re-enable MIG only if your workload requires it
# sudo nvidia-smi -mig 1
```

Investigate any new SEL entries (assertion events, thermal trips, fan failures) before restoring workloads.

Step 3.8 — Restore Workloads

```
kubectl uncordon <node-name>
```

Monitor for 30–60 minutes.

---

9. Rollback / Downgrade

Same flow as the forward upgrade, but with the previous .fwpkg and `force_update enable` set first. Disable `force_update` immediately after. EROT/CPLD downgrades also require the AC cycle.

---

10. Troubleshooting

Symptom	Cause	Action
`Invalid firmware file`	Wrong .fwpkg	Re-check `show_version -p` against the component
`servertype not recognised` / Redfish 404	nvfwupd defaulting to wrong server type	Add `servertype=DGX` (or HGX) to the `-t` target string
VBIOS update blocked — GPU activity	Process still using the GPU	Kill PIDs from `nvidia-smi –query-compute-apps`; drain K8s node
BackgroundCopyStatus: Failed	Wrong firmware file or transient BMC error	Do **not** reboot. Retry. Contact NVIDIA Support
EROT/CPLD version unchanged after reboot	AC power cycle skipped or too short	Repeat Step 2.9 — full AC removal, 5–6 minute wait
Server does not boot after AC cycle	Failed flash	BMC serial console + `ipmitool sel list`; contact NVIDIA Support
GPU missing after upgrade	PCIe or driver issue	`lspci`, `dmesg` for Xid; reload driver
NVLink ports Inactive	NVSwitch FW or fabric manager	Cold reboot; restart `nvidia-fabricmanager`
Firmware unchanged after reboot	Rebooted before background copy finished	Re-run, wait for `Completed`
`Downgrade not allowed`	ForceUpdate flag not enabled	See Section 9
BMC unreachable after a version-hop skip	Skipped the v1.1.3 intermediate hop (v1.0.0/v1.1.1 → v24.09.1 direct)	Factory-reset the BMC, then re-stage v1.0.0/v1.1.1 → v1.1.3 → v24.09.1; see “BMC Recovery” below
PSU update refuses to start	PSU redundancy degraded	`nvsm show health | grep -i psu`; replace failed PSU before retrying
Fabric Manager fails to start post-upgrade	NVSwitch FW / driver mismatch	Confirm driver ≥ release notes min; `journalctl -u nvidia-fabricmanager`; reload `nvidia` module
New SEL entries after upgrade	Thermal/fan/PSU event during reboot	Investigate each entry; do not restore workloads until cleared

BMC Recovery (Unreachable BMC)

If the BMC is unreachable after an update (no Redfish, no IPMI, no SSH), **do not attempt field recovery from a USB image**. The BMC recovery image, layout, and procedure are NVIDIA-internal and SKU-specific; running the wrong image can permanently brick the chassis BMC and is not covered by warranty.

Instead:

1. **Open an NVIDIA Enterprise Support case** with severity matching your maintenance window.
2. **Capture the BMC serial console** at **115200 8N1** (rear-panel serial port or serial-over-USB on the management cable) from the moment the BMC fails through any attempted recovery. Attach the full log to the support case — NVIDIA will require it before authorising any recovery image.
3. Provide: chassis SKU, current BMC version (last known good), target BMC version, the package P-number / file name attempted, and confirmation of whether the v1.0.0 / v1.1.1 → v1.1.3 → v24.09.1 hop was followed.
4. Do not pull the chassis from the rack or attempt a factory reset until NVIDIA confirms the recovery path — some failures are recoverable from the host side without touching the BMC.

---

11. Risk Register

Risk	Likelihood	Impact	Mitigation
Wrong firmware file flashed	Medium	Critical	`show_version` first; checksum
Power loss during flash	Low	Critical	Stable power; avoid power maintenance windows
Workload active during upgrade	Medium	High	Verify `nvidia-smi` apps query empty
Driver / firmware mismatch	Medium	High	Check per-package release notes
Network loss to BMC mid-update	Low	High	Wired network only
Previous .fwpkg unavailable	Medium	High	Save before every upgrade
ForceUpdate left enabled	Low	Medium	Disable + verify after downgrade
AC cycle skipped — EROT/CPLD not active	High	High	AC step is mandatory in Phase 2
BMC version-hop skipped (v1.0.0/v1.1.1 → v24.09.1 direct)	Medium	Critical	Stage v1.0.0/v1.1.1 → v1.1.3 → v24.09.1; direct skip requires a BMC factory reset to recover
Reboot before background copy completes	Medium	High	Poll Redfish until `Completed`

---

12. References

- NVIDIA DGX H100/H200 Firmware Update Guide
- Per-package release notes — **always consult for driver minimums and known issues**
- NVIDIA Enterprise Support Portal