Soterix NexaiQ Edge — User Operation Manual

Product: Soterix NexaiQ Edge appliance (VSaaS — Video Surveillance as a Service) **Vendor:** Soterix Systems · https://www.soterixsystems.com **Base OS:** Ubuntu 22.04.5 LTS (Server) · **GPU stack:** NVIDIA driver 575.51.03 + CUDA 12.9 **Image type:** Automated golden image — installs and self-provisions on its own, with a **single operator prompt** at first boot (you set the password for the built-in `soterix` login — **§6**). The OS install itself is fully unattended.

> This manual is for the technician who **deploys and operates** a NexaiQ Edge appliance from the golden ISO. It is not required reading for building the ISO from source (see **Appendix C**).

---

1. What this appliance does

You flash one USB stick from the golden ISO and boot the appliance from it. The OS install is fully unattended; the only thing you type is the **login password** once at first boot (the username is always `soterix` — **§6**). After that it runs on its own. The image:

1. Wipes and partitions the OS SSD, formats the database and video storage.
2. Installs the NVIDIA GPU driver + CUDA, Docker, and the NexaiQ VSaaS application.
3. Installs and starts the NexaiQ workload (the app registers the appliance's device UUID), then applies hardening + console branding.

When it finishes, the appliance is production-ready and the NexaiQ container is running.

---

2. Prerequisites — read this first

2.1 Hardware (the appliance)

Requirement	Detail
Architecture	x86-64 (amd64).
Firmware	**UEFI** boot mode (not legacy/CSM BIOS).
NVIDIA GPU	Required. The model is dictated by the appliance SKU — see **Appendix A**.
OS disk	One **SSD** (SATA or NVMe), **≥ 512 GB**. The installer auto-selects the **largest SSD** as the OS disk and **erases it** (rotational HDDs are never chosen). The root partition is a **fixed 300 GiB**, so a 256 GB-class drive is **too small** — use a 512 GB-class drive or larger; `/media/dbdata` (the database) takes whatever space remains after root + swap.
Video storage	One or more **rotational HDDs** (WD Purple-class). All non-OS HDDs are pooled into the video store (`/media/vdata`) at first boot — the **full pool** capacity is used.
RAM	Per appliance spec for the SKU.
Network	A wired Ethernet link with **DHCP** and **internet access** (see 2.3).

> **The OS SSD is wiped.** Installation reformats the **largest SSD** (`wipe: superblock-recursive`, `preserve: false`). It also tears down any pre-existing swap/LVM/RAID so old hardware can be re-imaged. **Back up anything you need before installing.** The video HDD pool is **always rebuilt clean** — any existing `vdata_vg` is deleted and recreated, so **previously recorded video on the HDDs is erased** on (re)install. Only the dedicated video HDDs (non-OS, not currently mounted) are touched.

2.2 Firmware / BIOS settings — mandatory

Before installing, enter the system firmware (BIOS/UEFI setup) and set:

Setting	Required value	Why
Secure Boot	**DISABLED**	The NVIDIA driver is an unsigned DKMS kernel module. With Secure Boot ON it will **never load**, leaving the appliance with no working GPU. The installer **detects Secure Boot and halts with an on-screen message** rather than ship a dead-GPU box.
Boot mode	**UEFI**	The image is a UEFI ISO.
Boot order	USB first (for install)	So the appliance boots the install USB.

> If you see a red **"SECURE BOOT IS ENABLED — INSTALLATION HALTED"** banner on the console, power off, disable Secure Boot in firmware, and reinstall.

2.3 Network

- The appliance needs **internet access during install and first boot**: the NexaiQ application (and any small base packages) are downloaded online from `applications.soterixcloud.com` and Ubuntu mirrors.
- A wired interface with **DHCP** is expected (the image auto-configures `en*`/`eth*` interfaces).
- Outbound HTTPS (443) must be permitted to Soterix and Ubuntu/NVIDIA endpoints.

2.4 Tooling you (the technician) need

Item	Purpose
USB flash drive, ≥ 16 GB	The golden ISO is ≈ 7.4 GB. Use a **USB 3.0** drive in a **USB 3.0 (blue) port** — flashing a USB-2 stick can take 30–60 min vs a few minutes on USB 3.0.
A second computer (Linux/Windows/macOS)	To download and flash the ISO.
Flashing tool	`dd` (Linux/macOS), **Rufus** or **balenaEtcher** (Windows).
A monitor + keyboard on the appliance	To watch the unattended install/provisioning and confirm completion.

---

3. Download the golden ISO

> **Download link:**
> `https://applications.soterixcloud.com/vsaas/utill/NexaiQ_Golden_ISO/soterix-nexaiq-edge-golden-22.04.5.iso`

> **File:** `soterix-nexaiq-edge-golden-22.04.5.iso` (≈ 7.4 GB) **SHA256:** `624c62f28c1bec109b2a2816f9ae5b70493c05f502664d906a9aaf69d058878a` **(A `.sha256` sidecar file is published alongside the ISO.)**

Always verify the checksum before flashing — a truncated download produces a broken appliance.

```
# Linux / macOS
sha256sum soterix-nexaiq-edge-golden-22.04.5.iso          # Linux
shasum -a 256 soterix-nexaiq-edge-golden-22.04.5.iso      # macOS
```

```
# Windows (PowerShell)
Get-FileHash .\soterix-nexaiq-edge-golden-22.04.5.iso -Algorithm SHA256
```

The printed hash **must match** the published SHA256 above. If it does not, re-download.

> **Verify on the SAME computer you will flash from, right before flashing.** Copying the ISO between machines (network share, USB, cloud sync) can silently drop the tail of the file. A copy that is even 2 KB short is missing the image's **backup GPT header**, and the appliance will report **"no bootable device"** even though the original ISO is fine. So after the ISO reaches the Windows/flashing PC, re-check both:

> • **File size = exactly `7,908,767,744` bytes** (a truncated copy shows `7,908,765,696`). In Windows: right-click the file -> Properties -> "Size" (the exact byte count is in parentheses).

> • **SHA256 = the published value above.** In `cmd`: `certutil -hashfile soterix-nexaiq-edge-golden-22.04.5.iso SHA256`, or click the **checksum button in Rufus** (it computes SHA-256).

> If **Rufus pops up "Truncated ISO detected -- N KB of data is missing"**, the copy on that PC is incomplete -- **do not flash it**. Delete it and copy the ISO again (verify size + SHA), then flash.

---

4. Create the bootable USB

> Flashing **erases the entire USB drive**. Double-check the target device.

Linux / macOS (`dd`)

```
# Identify the USB device first (e.g. /dev/sdX on Linux, /dev/diskN on macOS)
lsblk                       # Linux
diskutil list               # macOS

# Write it (replace /dev/sdX with YOUR usb device — NOT a partition like sdX1)
sudo dd if=soterix-nexaiq-edge-golden-22.04.5.iso of=/dev/sdX bs=4M status=progress oflag=sync
sync
```

Windows

1. Open **Rufus** (or **balenaEtcher**).
2. Select the USB device and the `soterix-nexaiq-edge-golden-22.04.5.iso`.
3. Click **START**. Rufus detects an **ISOHybrid** image and asks how to write it — **choose "Write in DD Image mode"** (the second, non-default option), then OK and confirm the data-wipe warning.
4. Wait for the full raw write to finish (≈ 7.4 GB — slower than ISO mode; this is normal and required).
5. When it finishes, Windows may report the drive as unreadable or offer to **format it — click Cancel / No**. The stick is now a Linux GPT image; that's expected.

> **"No bootable device" on the appliance = the USB was written in the wrong Rufus mode.** This is the #1 install failure. You **must** use **DD Image mode**. Rufus's default **ISO Image mode** copies files onto a FAT32/NTFS partition — it (a) cannot hold the image's single **4.4 GB** CUDA package and (b) discards our UEFI **GPT/ESP**, so the appliance's UEFI firmware finds nothing to boot.

> **How to tell which mode Rufus is in — look at the Format Options before you click START:**

Field	Wrong (ISO mode — won't boot)	Right (DD mode — boots)
Partition scheme	editable (e.g. `MBR`)	**greyed out / ignored**
File system	editable (e.g. `NTFS`/`FAT32`)	**greyed out / ignored**
Volume label	editable	**greyed out / ignored**

> In **DD mode those fields are disabled** because Rufus byte-copies the ISO verbatim — you do **not** need to set GPT, UEFI, file system, or label. If those fields are still active, you're in ISO mode: click START and pick **DD Image mode** at the prompt. If Rufus ever skips the prompt, use **balenaEtcher**, which always does a raw write.

> **Flashing slow (~1 hour)?** That's almost always a slow USB-2 drive/port, not the tool. To speed it up: use a **USB 3.0 drive in a USB 3.0 (blue) port**; in Rufus **uncheck "Check device for bad blocks"**; and confirm you're in **DD image mode**. balenaEtcher's verification pass also doubles the time — it can be skipped. The image is ≈ 7.4 GB because it carries the NVIDIA/CUDA + Docker packages **offline** (so the appliance provisions without large first-boot downloads); on good USB 3.0 media this writes in a few minutes.

---

5. Install on the appliance

1. Insert the USB stick into the appliance. Connect **monitor, keyboard, and Ethernet**.
2. Power on and enter the **one-time boot menu** (commonly **F12 / F11 / F9 / Esc** — varies by vendor). Select the **USB device (UEFI)**.
3. The branded **Soterix NexaiQ Edge** boot menu appears and auto-starts the install.
4. **No prompts during the OS install — it is fully unattended.** The installer wipes the OS SSD and lays down the OS (login user `soterix`, hostname `nexaiq-edge`, both fixed), then **reboots** when the OS install completes. You set the login password at first boot, not here (**§6**). - The kernel (Ubuntu HWE **6.x**) is installed during this phase so the appliance boots straight into the GPU-capable kernel.

> If installation halts with the **Secure Boot** banner, go back to **§2.2**, disable Secure Boot, and reinstall.

Remove the USB stick after the first reboot (or set internal disk first in boot order) so it boots from the SSD.

---

6. First boot — automatic provisioning

On the first boot from the SSD, the appliance runs the **first-boot orchestrator** (`nexaiq-firstboot.service`) on the console. Watch the monitor — progress streams live to the screen.

> **Secure Boot check (first boot).** The very first thing first boot does is re-check Secure Boot. If it is **enabled**, the console prints `SECURE BOOT IS ENABLED — PROVISIONING HALTED` and stops **before** the password prompt — disable Secure Boot in firmware (**§2.2**) and power on again; provisioning resumes on its own.

> **Set the login password (the one prompt — required).** Before provisioning starts, the console asks you to **set the password for the built-in `soterix` login**. Type it, retype to confirm, and press Enter. **This step is mandatory — provisioning will not continue until you enter it** (the prompt blocks here, and it re-prompts on the next boot if the screen is closed before you type it). This is the appliance login for **console + SSH** — record it where your commissioning process requires; there are **no factory-default credentials**, and until you set it the account is locked. (The username is always `soterix`; you are not asked for it.)

After the password is set, SKU detection and network config run (best-effort), then the **fixed provisioning sequence**:

1. **Kernel update** — ensures the HWE 6.x kernel (reboots once to apply if it had to upgrade).
2. **Docker** — installs the container runtime.
3. **NVIDIA driver + CUDA** — installs and brings up the GPU (may reboot once).
4. **LVM group creation** — pools the surveillance HDDs into the `vdata_vg` volume group.
5. **Mount `/media/dbdata` + `/media/vdata`** — both must be mounted before the app (the install stops here with guidance if they aren't).
6. **NexaiQ application** — downloads and starts the VSaaS workload; **the app generates the appliance's device UUID** and shows it on the console (record it where your process requires).

Security hardening + console branding run last (best-effort).

Provisioning is complete when the orchestrator writes `/etc/soterix/.firstboot_done` and the service disables itself. The NexaiQ container is then running.

> **Record the Device ID.** At the very end of provisioning the appliance prints the **NexaiQ Device ID** on the monitor in **large green text** (read from the app's `vtpl_cnf/vsaas_cloud_config.cnf` → `VTPL_VSAAS_UNIQUE_ID`). **Write it down** — it's needed for activation/support. If you miss it, it's saved on the appliance at **`/etc/soterix/device-uuid.txt`** and shown in the login banner (MOTD).

> This takes several minutes (driver build + container/app download). In some hardware cases the NVIDIA stage may reboot once and resume automatically — this is normal; let it finish.

---

7. Log in

Log in on the **console** or over **SSH** with the username **`soterix`** and the **password you set at first boot** (**§6**) — there are no factory-default credentials.

Field	Value
Username	**`soterix`** (fixed)
Password	the password you set at first boot
Hostname	**`nexaiq-edge`** (fixed)
SSH	Enabled, password login allowed (`ssh soterix@<appliance-ip>`).

> **Security:** the password is operator-set at first boot, so set a strong one at commissioning. You can change it later on the box with `passwd`. (An additional SSH public key can be baked in at build time via `SOTERIX_SSH_AUTHORIZED_KEY` — installed into `soterix`'s `~/.ssh`; see **Appendix C**.)

> **Lost the credentials?** There is no recovery account or default password — re-image the appliance (**§4–§5**) and set them again.

Find the appliance IP from the console login banner, your DHCP server, or `ip -br a` on the console.

---

8. Verify the deployment

After `/etc/soterix/.firstboot_done` exists, run the bundled validator **as root on the appliance** to confirm production state:

```
sudo bash /usr/local/sbin/validate-on-target.sh
```

It checks and prints **PASS** / **FAIL** / **WARN** for:

- First-boot completion marker present.
- Partition layout, labels (`SOTERIX-ROOT`, `SOTERIX-DBDATA`, `SOTERIX-ESP`, `SOTERIX-VDATA`) and mounts (`/media/dbdata`, `/media/vdata`).
- LVM video pool on the HDDs.
- **NVIDIA** driver reports **575.51.03** (`nvidia-smi`).
- **Docker** installed, service enabled, daemon running.
- NexaiQ container present.

Exit code 0 / "required checks PASSED" means the appliance is good. GPU/app checks soft-fail (WARN) on a GPU-less test VM.

Quick manual spot-checks:

```
nvidia-smi                  # GPU + driver 575.51.03
docker ps                   # nexaiq container running
df -h /media/dbdata /media/vdata
```

---

9. Troubleshooting

Symptom	Likely cause	Action
"No bootable device" / USB not in UEFI boot menu	USB flashed with **Rufus ISO mode** (file-copy) — breaks our UEFI GPT/ESP and can't hold the 4.4 GB CUDA file; **or** the ISO copy on the flashing PC was **truncated** (missing backup GPT)	Re-flash the same ISO in **Rufus → "Write in DD Image mode"** (or balenaEtcher). First confirm the ISO copy is intact (size + SHA256, **§3**). The ISO itself is fine; the write method or a short copy was the problem.
Rufus: "Truncated ISO detected — N KB of data is missing"	The ISO copy on this PC is incomplete — the transfer (network share / USB / sync) dropped the tail of the file	**Do not flash it.** Delete the copy, copy the ISO again, and verify **size = `7,908,767,744` bytes** and the **SHA256** match (**§3**) before flashing.
Red **"SECURE BOOT ENABLED — HALTED"** banner at install	Secure Boot is ON	Power off → firmware setup → **disable Secure Boot** → reinstall.
`nvidia-smi` fails / "no devices" after provisioning	Secure Boot still ON (module won't load), or driver build issue	Confirm `mokutil --sb-state` says **disabled**. If enabled, disable in firmware. Then clear the NVIDIA stage markers and let it re-run (below).

Appliance reboots repeatedly during NVIDIA stage	Driver module can't load (usually Secure Boot)	The stage has a bounded reboot guard and will stop with diagnostics. Check `mokutil --sb-state`; review the NVIDIA log (below).
Boots into the wrong kernel (5.x)	Rare; kernel didn't install at OS-install time	Confirm `uname -r` is **6.x**. If not: `sudo apt-get install -y linux-generic-hwe-22.04 && sudo update-grub && sudo reboot`.
No `/media/vdata`	No HDDs detected, or all disks busy	Provisioning refuses to wipe mounted/OS disks. Check `lsblk`; ensure video HDDs are attached and not the OS disk.
App not running	App download/start hiccup at first boot	Re-run provisioning (below); check Docker is up (`docker ps`).
`SKU=UNKNOWN` in `/etc/soterix/sku.env`	GPU and/or WD Purple HDD count match no listed SKU	Verify the correct GPU (`lspci | grep -i nvidia`) and that all surveillance HDDs are detected (`lsblk`). To force it, write the SKU into `/etc/soterix/sku` and re-run the SKU stage. See **Appendix A**.

Re-run a stuck stage (markers live under `/etc/soterix/`; deleting one forces that stage to re-run):

```
# Example: force the NVIDIA stage to re-run after fixing Secure Boot
sudo rm -f /etc/soterix/.stage_nvidia_done /etc/soterix/.nvidia_reboot_count /var/run/vsaas_needs_rerun_aft
sudo systemctl start nexaiq-firstboot.service     # or reboot
```

Where the logs are:

Log	Path
First-boot orchestrator / scaffold	`/var/log/installation_logs/`
NVIDIA / Docker installers	`/usr/local/sbin/installers/logs/deploylogs/*.log`
Storage summary	`/var/log/installation_logs/storage-summary.txt`
First-boot service journal	`journalctl -u nexaiq-firstboot.service`

Appendix A — SKU model & GPU mapping

Appliance SKUs follow the pattern **`PVA-<channels>-<retentionDays>NR`** (e.g. `PVA-50-30NR` = 50 channels, 30-day retention). The SKU is auto-detected against the authoritative **Soterix Edgebox Configuration (SKU-wise)** table, in this order: DMI product name (must be a listed SKU) → `/etc/soterix/sku` override → **hardware fingerprint** (NVIDIA GPU + WD Purple HDD count). **If the hardware matches no listed SKU, it is recorded as `UNKNOWN`** — there is no approximate guessing.

SKU	Channels	Retention	Primary GPU	HDDs	Total	Usable
PVA-25-30NR	25	30 d	RTX A1000	2 × 14 TB	28 TB	~25.2 TB
PVA-25-60NR	25	60 d	RTX A1000	3 × 18 TB	54 TB	~48.6 TB

PVA-25-90NR	25	90 d	RTX 2000 Ada	4 × 22 TB	88 TB	~79.2 TB
PVA-50-30NR	50	30 d	RTX 2000 Ada	3 × 18 TB	54 TB	~48.6 TB
PVA-50-60NR	50	60 d	RTX 2000 Ada	5 × 22 TB	110 TB	~99 TB
PVA-50-90NR	50	90 d	RTX 2000 Ada	7 × 24 TB	168 TB	~151.2 TB
PVA-75-30NR	75	30 d	RTX 4000 Ada	4 × 22 TB	88 TB	~79.2 TB
PVA-75-60NR	75	60 d	RTX 4000 Ada	7 × 24 TB	168 TB	~151.2 TB
PVA-75-90NR	75	90 d	RTX 4000 Ada	10 × 24 TB	240 TB	~216 TB
PVA-100-30NR	100	30 d	RTX PRO 4500	5 × 22 TB	110 TB	~99 TB
PVA-100-60NR	100	60 d	RTX PRO 4500	9 × 24 TB	216 TB	~194.4 TB
PVA-100-90NR	100	90 d	RTX PRO 4500	13 × 24 TB	312 TB	~280.8 TB

All units use the **i7-14700** CPU and a **512 GB** OS SSD. Detected SKU details are written to `/etc/soterix/sku.env` (`SKU=`, `CHANNELS=`, `RETENTION_DAYS=`, `PRIMARY_GPU=`, `HDD_COUNT=`, `USABLE_TB=`). A mismatch reads `SKU=UNKNOWN`.

> **If the SKU shows `UNKNOWN`:** the GPU and/or the number of WD Purple HDDs don't match any listed configuration. Verify the correct GPU is installed and all surveillance HDDs are seated/detected (`lspci | grep -i nvidia`, `lsblk`). You can also force a value by writing the SKU into `/etc/soterix/sku` and re-running the SKU stage.

Appendix B — Storage layout

OS SSD (created at install, GPT):

Partition	Size	Label	Mount
EFI System	512 MiB	`SOTERIX-ESP`	`/boot/efi`
Root (ext4)	300 GiB	`SOTERIX-ROOT`	`/`
Swap	4 GiB	—	swap
Database (ext4)	remainder	`SOTERIX-DBDATA`	`/media/dbdata`

> Root is **300 GiB** (the NexaiQ container images need the headroom); this requires a **≥ 512 GB-class OS SSD**. `/media/dbdata` then takes whatever NVMe space remains after root (e.g. ~172 GiB on a 477 GB SSD).

Video HDD pool (created at first boot): all non-OS rotational disks → LVM `vdata_vg` → `vdata` (ext4, **full pool** — `lvcreate 95%VG`) → label `SOTERIX-VDATA` → **`/media/vdata`** (mounted by UUID, `nofail` so a degraded

pool never blocks boot).

Appendix C — Building the ISO from source (engineering only)

The golden ISO is built from the standalone scripts in this repository on an Ubuntu 22.04 amd64 build host (root, with internet; a container runtime — Docker/Podman — is recommended so the offline package closure is complete).

```
# Turnkey offline build (recommended)
sudo ./build-offline-iso.sh

# Direct build with env overrides
OUTPUT_ISO=/tmp/edge.iso ./build-golden-image.sh
OFFLINE_BUNDLE=0 sudo ./build-golden-image.sh        # smaller online-install ISO
SOTERIX_SSH_AUTHORIZED_KEY="$(cat key.pub)" ./build-golden-image.sh   # bake an extra SSH key
```

The build prints the output ISO path and **SHA256** — publish both at the download link in **§3**. See `CLAUDE.md` in the repo for the full build architecture.

> **Install speed / region note.** To shorten the OS-install phase, the autoinstall skips the install-time security-update download (the appliance updates via the app's OTA) and installs the base tools from the on-ISO offline repo instead of the network mirror. For the few packages that do come from the network, the installer uses **geoip** to select the Ubuntu mirror **nearest to the appliance's location** automatically (no region is hard-coded), so the same ISO installs quickly wherever it is deployed.

Soterix Systems — Intelligent Security, Unmatched Efficiency · https://www.soterixsystems.com