Meerkat user guide

Meerkat is **a desktop app that finds the PTZ cameras on a network and gives them the right address**. It speaks each brand's own discovery protocol, so it finds cameras on the wrong subnet as well as the right one; it changes a camera's address, mask, gateway, DNS and name where the brand allows it; and it opens each camera's built-in web page with one click. It knows Panasonic (Easy IP), Sony (IP Setup), BirdDog (its REST API), PTZOptics and other VHD cameras, any ONVIF camera and any NDI camera. For each camera it shows the login it leaves the factory with and its maker's firmware page, and it can send a firmware file you downloaded to a camera that takes one over ONVIF.

> **Before you rely on this:** every driver was written from the vendor's own published documentation and is tested end to end against a simulator built into Meerkat, which answers every protocol the way those documents say a camera does. Discovery has been run, read-only, on a real network. But **no real camera has had its address changed by Meerkat yet**, and the temporary-address commands described below have never been run on a real network interface. Try a change on an idle camera on the bench before you use it on a show. See What has been checked. **Status: beta, released at v0.2.0.**
>
> This codebase was created with Claude (Anthropic), directed and reviewed by a human author.

Installing

Download the build for your computer from the project page or from the GitHub release:

- **macOS:** the universal `.dmg` runs on Apple Silicon and Intel. Open it and drag Meerkat to Applications. The first scan makes macOS ask whether Meerkat may find devices on your local network: allow it, or every camera looks absent. If you said no, turn it back on in **System Settings → Privacy & Security → Local Network**.
- **Windows:** the `-setup.exe` installer. It is not code-signed, so SmartScreen shows *Windows protected your PC*; choose **More info → Run anyway**. Windows Firewall may ask whether Meerkat can use the network: allow it on the networks your cameras are on.
- **Linux:** the `.deb` or the `.rpm`.

Meerkat needs nothing else installed. It sends nothing anywhere except to the cameras.

The window

- **The sidebar** chooses what a scan does: which network interfaces it goes out of, any routed subnets to sweep, which brands to look for and how long to listen. It also holds the temporary addresses you have added, the vendors' recovery tools, and the **Demo cameras** switch.
- **The table** lists every camera found, one row per camera, keyed by its MAC address where the protocol gives one. The filter box above it matches brand, model, name, address and MAC.
- **The detail panel** on the right shows the selected camera: what is known about it, how it was found, its network settings and the buttons that act on it.
- **Log** at the bottom right shows what each driver sent and heard, which is the first place to look when a camera you expected is missing.

Scanning

Press **Scan**. Every brand ticked under **Brands** sends its discovery out of every interface ticked under **Interfaces**, each socket bound to its interface, and listens for the time chosen under **Listen for** (1.5 to 10 seconds; 2.5 is right for most networks). Cameras appear as they answer. A camera that answers more than one protocol (a BirdDog heard over NDI and confirmed over its API, say) is merged into one row owned by the most specific driver.

What each brand's discovery can see:

Brand	Discovery	Sees a camera on the wrong subnet?
Panasonic	Easy IP Setup broadcast, UDP 10670 → 10669	Yes
Sony	Sony IP Setup broadcast, UDP 52380	Yes
PTZOptics / VHD	the VHD upgrade tool's multicast, 239.255.255.251:8005	Yes
NDI (BirdDog and others)	mDNS `_ndi._tcp`	Yes
ONVIF	WS-Discovery, 239.255.255.250:3702	No: its answers come back by unicast

Routed subnets. A camera behind a router never hears a broadcast. Type its range under **Routed subnets**, one per line, as a CIDR block (`10.20.0.0/24`) or a range (`10.20.0.10-60`). A scan then sends each brand's discovery to that subnet's directed broadcast and sweeps it address by address: an HTTP probe for Panasonic and BirdDog, a UDP inquiry for Sony and a WS-Discovery probe for ONVIF. Cameras found that way show as **routed**.

Hover over a brand to see how its discovery works and what it cannot do. After you have changed which brands are ticked, **Back to the defaults** puts the choice back.

Reading the table

The **Reach** column says whether this computer can talk to the camera directly:

- **an interface name** (`en0` , `Ethernet`): the camera is inside that interface's subnet. Everything works.
- **routed**: it answered a probe through a router. Its web page opens and HTTP drivers can change it.
- **wrong subnet**: it was heard at layer 2 on an interface, but its address is outside that interface's subnet. Brands that address by MAC (Panasonic, Sony) can still be moved from here; anything that needs HTTP cannot, until you reach it.
- **unknown**: there is not enough to say.

A **DHCP** tag in front of an address means the camera says it took that address from a DHCP server.

Forget in the detail panel drops a camera from the list until a scan finds it again. **Clear** empties the table.

Changing a camera's address

Select the camera. Under **Network settings**, set the address, mask (as a prefix length: `24` is 255.255.255.0), gateway and DNS, and the name if the brand has one. **Pick a free address** fills in the next address on the subnet of the interface that heard the camera that no camera in the table is using.

Some brands need the camera's login:

- **ONVIF** needs it to read the settings as well as to change them. Fill in **User** and **Password**, then **Read its current settings**.
- **BirdDog** and **Panasonic** need it only if authentication is turned on in the camera.

Use this login under **Factory login** fills in the maker's default (see below).

Press **Apply...**. Before anything is sent, Meerkat checks the new settings: that the address is valid and not already used by a camera it can see, that the gateway is inside the subnet, that the brand supports DHCP if you asked for it, and that it can reach the camera the way this brand needs. It then shows the change for you to confirm with **Change it**. After sending, it looks for the camera at its new address; the row stays the same row, because it is keyed by MAC.

What each brand allows:

- **Panasonic:** changed by MAC, so it works on the wrong subnet. The camera only accepts a change in the first 20 minutes after it is powered on, unless it is set to allow it at any time. No

DHCP over Easy IP. Cameras of the UE160 generation with authentication on use an encrypted variant that Meerkat does not speak.

- **Sony:** changed by MAC, so it works on the wrong subnet. Some models (BRC-AM7, ILME-FR7) lock IP setup 20 minutes after power-on, and Meerkat says so when one refuses. The name is at most 8 letters, digits or spaces. No DHCP.
- **BirdDog X-series and MAX (firmware 6.x):** changed over its API, including DHCP; the camera reboots to take the new settings. The camera has to be reachable over HTTP.
- **BirdDog P-series and converters (firmware 5.5.x):** have no network API. Meerkat finds them and opens their web page, where you change the address yourself.
- **ONVIF:** changed over ONVIF with the camera's login; the camera has to be reachable.
- **PTZOptics / VHD:** discovery only for now. Use the camera's web page, or its ONVIF service if it has one.
- **NDI cameras of other brands:** found and given a web-page link; Meerkat cannot change them.

Several cameras at once. Tick their boxes, press **Address in sequence...**, and give the first address, the mask and the gateway. Meerkat gives them consecutive addresses in the order the table shows them, one at a time, checking each before the next.

Opening the web UI

Open web UI opens the camera's own page in your browser, on the port that brand uses. A camera on the wrong subnet cannot be opened until this computer can reach it: see the next section.

Factory logins

The detail panel's **Factory login** line says what the selected camera logs in with out of the box, from its maker's own documents. Open it for the details:

- **The login**, or **none** when the camera has no factory login and the account is created at the first visit to its web page.
- **Which firmware** it applies to, when the firmware decides. Both are shown.
- Whether the camera makes you **replace it at the first login**.
- **What happens after wrong passwords**, when the maker says. The Sony FR7, for one, blocks an address for 60 seconds after 8 wrong logins.
- The maker's notes, and the **sources**: each document and the place in it. A source marked **excerpt only** or **not the maker** is weaker.

When the camera needs a login for a change, **Use this login** fills in the fields. **Meerkat never tries a login by itself.** It sends one only when you press **Apply...** or **Read its current settings**.

When Meerkat does not know the camera's model, or the camera did not say, it lists the brand's logins for each model family instead. **Factory logins…** in the sidebar lists every brand, with a search by model.

Firmware updates

Meerkat does not supply or download firmware. The **Firmware update** section points you at the maker:

- **The maker's download page** for the model. Some ask you to accept a licence first.
- **The maker's steps**, and **the maker's warnings**: how long it takes, what not to unplug, the order of files, and what cannot be undone.
- **Open its firmware page** opens the update page inside the camera's own web UI, for the brands whose procedure shows its address (Canon).
- The firmware the camera reports, where it reports one. Compare it with the maker's page.

Update one camera at a time, from a computer on its own network, and never during a show. **Firmware downloads…** in the sidebar lists every brand's pages.

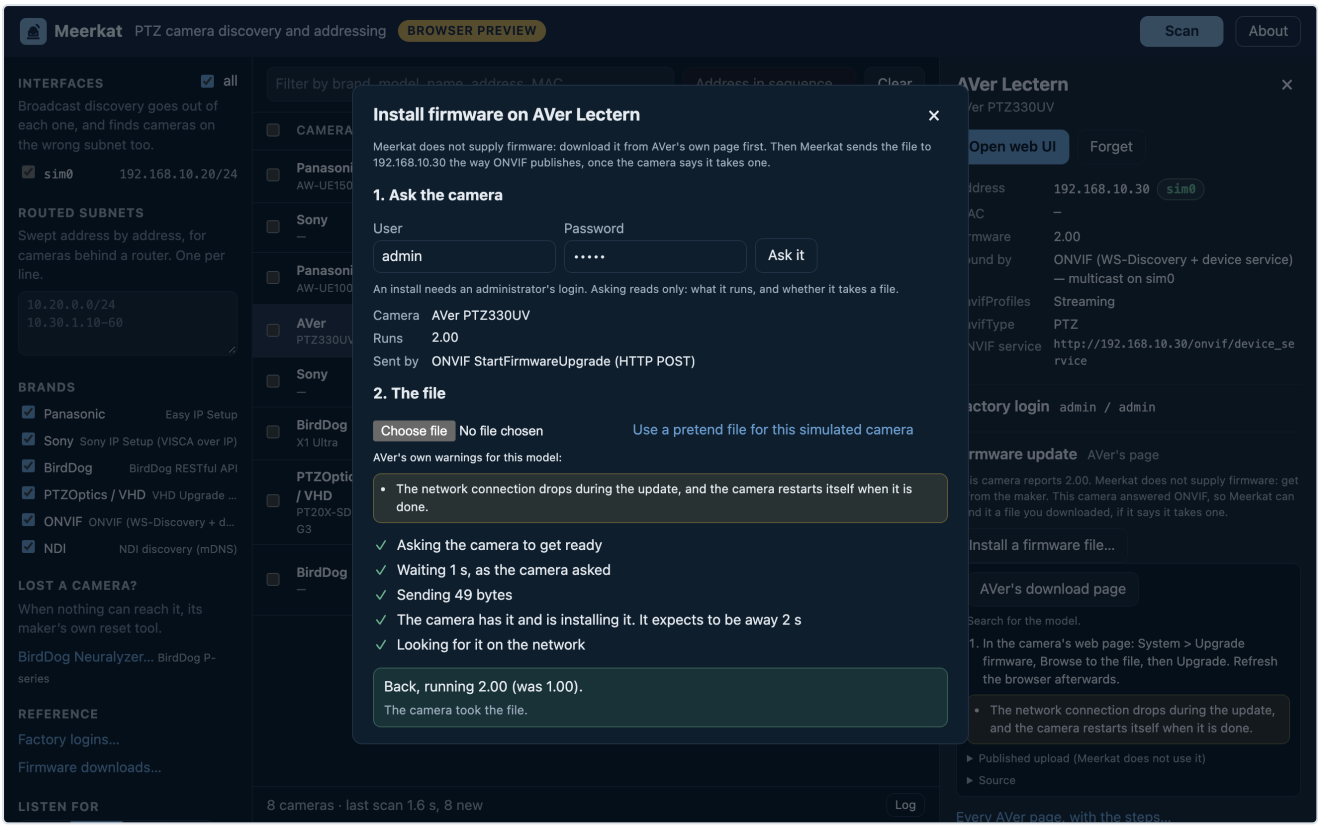
Installing a file from Meerkat

A camera found over ONVIF that says it takes a firmware file over HTTP shows **Install a firmware file…** in its firmware section. Download the file from the maker's page first; the dialog then goes in three steps.

1. **Ask the camera.** Give its administrator login and press **Ask it**. Meerkat reads what the camera is, the firmware it runs, and whether it takes a file. This only reads. A camera that does not say it takes one gets the reason, and nothing more happens.
2. **The file.** Choose the file you downloaded. Meerkat shows its size and its SHA-256: if the maker publishes a checksum, it should match. The maker's own warnings for the model are shown again here.
3. **Confirm.** Tick that the file is the maker's firmware for that model, from the maker's own page, and press **Install on …**.

Meerkat then asks the camera to get ready, waits the time the camera asks for, sends the file, and waits as long as the camera says it will be away. It looks for the camera again and reads its version:

- **Back, running 2.00 (was 1.00)**: the install took.
- **Back, but still running …**: the camera did not take the file. Check its web UI.
- **Not back after …**: it may still be installing. Leave it powered, and scan again later.
- **The camera refused the file (HTTP 415)**: the file is not one this camera takes. Nothing was installed.



Keep the camera powered and connected until it is back: a camera that loses power while it installs can need its maker's repair. Meerkat cannot undo an update. Under ONVIF the camera keeps its address and logins. The file always goes to the camera's own address, never to another host the camera names, and never over a camera on another subnet: reach it first.

Brands whose upload is not ONVIF (Sony, AVer's own upload, Panasonic) keep the maker's steps. In demo mode, the simulated AVer takes **Use a pretend file for this simulated camera**.

A camera on the wrong subnet

A camera heard on the wrong subnet gets a **Reach it from here** panel. It gives the interface that heard the camera a temporary second address in the camera's subnet, so this computer can talk to it directly: its web page opens, and a driver that works over HTTP (BirdDog, ONVIF) can change its settings. For Panasonic and Sony, which can be moved by MAC anyway, the panel starts folded away behind **Open its web UI from here too**.

Meerkat suggests an address near the top of the camera's subnet that no camera in the table is using, and you can change it. It refuses one that overlaps a subnet the computer is already on. Press **Add temporary address**:

- **macOS** asks for an administrator password, and runs `ifconfig <interface> alias`.
- **Windows** asks for permission to make changes, and adds the address with `New-NetIPAddress`, marked so it is never used as a source address for anything else.
- **Linux** asks through `pkexec`, and runs `ip addr add`.

The addresses in use are listed under **Temporary addresses** in the sidebar, each with a **Remove** button. Meerkat removes them all when it quits (asking for the password once more), and a restart removes them anyway: none of these commands survive one. Pick an address that nothing else on the camera's network uses. **Demo cameras** cannot be switched on while a temporary address exists: remove it first.

Lost a BirdDog?

A BirdDog P-series camera given an address nothing can reach can be put back with BirdDog's own **Neuralyzer**, which resets the camera's network settings to the factory defaults. BirdDog has not published how Neuralyzer talks to the camera, so Meerkat cannot do the reset itself. **Lost a camera?** in the sidebar, or the link on a BirdDog that cannot be changed, opens the steps:

1. Download Neuralyzer from BirdDog and run it on a computer on the same switch as the camera. BirdDog says the camera needs current firmware.
2. Use it to reset the camera's network settings.
3. Come back and press **Watch for it**.

Watch for it scans every few seconds for up to five minutes, with every default brand and a direct probe of BirdDog's fallback address, and lists every BirdDog that turns up new or at a new address. Out of the box a BirdDog uses DHCP and falls back to 192.168.100.100/24 when no DHCP server answers, so a camera that comes back there is usually on another subnet: select it, and use **Reach it from here** to open its web page.

Demo cameras

Tick **Demo cameras** (or start Meerkat with `MEERKAT_DEMO=1`, or `--demo`) to work against eight simulated cameras instead of the network: two Panasonic, two Sony, two BirdDog, one ONVIF and one VHD, on a pretend interface called `sim0`, some on the wrong subnet, one locked and one outside its 20-minute window. Every protocol is answered on loopback and **nothing is sent to the network**: in demo mode every socket is bound to 127.0.0.1 and every destination is rewritten to the simulator or to a closed port. It is the place to learn the app, and to try a change before making it for real.

The command line

The same engine runs without the window, from a checkout of the source:

```
cd src-tauri
cargo run -p meerkat-cli -- scan                        # every interface, every brand
cargo run -p meerkat-cli -- scan --range 10.20.0.0/24  # plus a routed subnet
cargo run -p meerkat-cli -- set <camera> --ip 192.168.10.21 --prefix 24 --gw 192.168.10.1
# dry run
cargo run -p meerkat-cli -- set <camera> --ip 192.168.10.21 --prefix 24 --gw 192.168.10.1 --
yes  # do it
cargo run -p meerkat-cli -- web <camera>
cargo run -p meerkat-cli -- install <camera> fw.bin --user admin --pass …         # dry run
cargo run -p meerkat-cli -- install <camera> fw.bin --user admin --pass … --yes   # send it
cargo run -p meerkat-cli -- scan --demo --json         # against the simulator
```

`<camera>` is a MAC address, an IP address or a row id from the scan. `set` prints the current and the new settings and does nothing without `--yes`; `install` without a file only asks the camera what it runs, and with one does nothing without `--yes`.

What has been checked

- **Every driver against the simulator**, end to end: discovery, reading the settings, changing them, and finding the camera again at its new address. The simulator answers each protocol as the vendor's documentation describes it.
- **Discovery on a real network**, read-only.
- **Temporary addresses in demo mode**, including a BirdDog reached on its fallback address and re-addressed over its API.
- **The command layer of the desktop app**, on Tauri's mock runtime.
- **Firmware install over ONVIF against the simulator**: the login, the camera's answer, a file for another model refused, a wrong login refused, and the camera back on the new version.

Not yet:

- **Changing the address of a real camera**, of any brand. The first real Panasonic, Sony and BirdDog on the bench are the next test, each read first and then re-addressed.
- **Sending firmware to a real camera.**
- **The temporary-address commands on a real interface**, on any of the three systems. Only the exact commands are checked, in the tests.
- **Neuralyzer** alongside Meerkat.
- **The Windows and Linux builds** against a camera.

Running it from source

```
npm install
npm run app            # the desktop app, built from source
npm run app:demo       # the same against the simulator
npm test               # the address helpers
cd src-tauri && cargo test --workspace   # drivers, engine, simulator end to end, command
layer
```

The source is on GitHub, under the MIT licence. Adding a brand is one file: see `docs/drivers.md` in the repository.

NDI® is a registered trademark of Vizrt NDI AB. See ndi.video.
