Dark Ambient Radio — System Documentation

Purpose of this document. A complete, implementation-level description of the broadcast chain running on host `macdeb`, written so that it can be turned into Ansible roles without further archaeology. Every path, file mode, unit name and non-obvious workaround is recorded, including the reasons behind decisions that would otherwise look arbitrary.

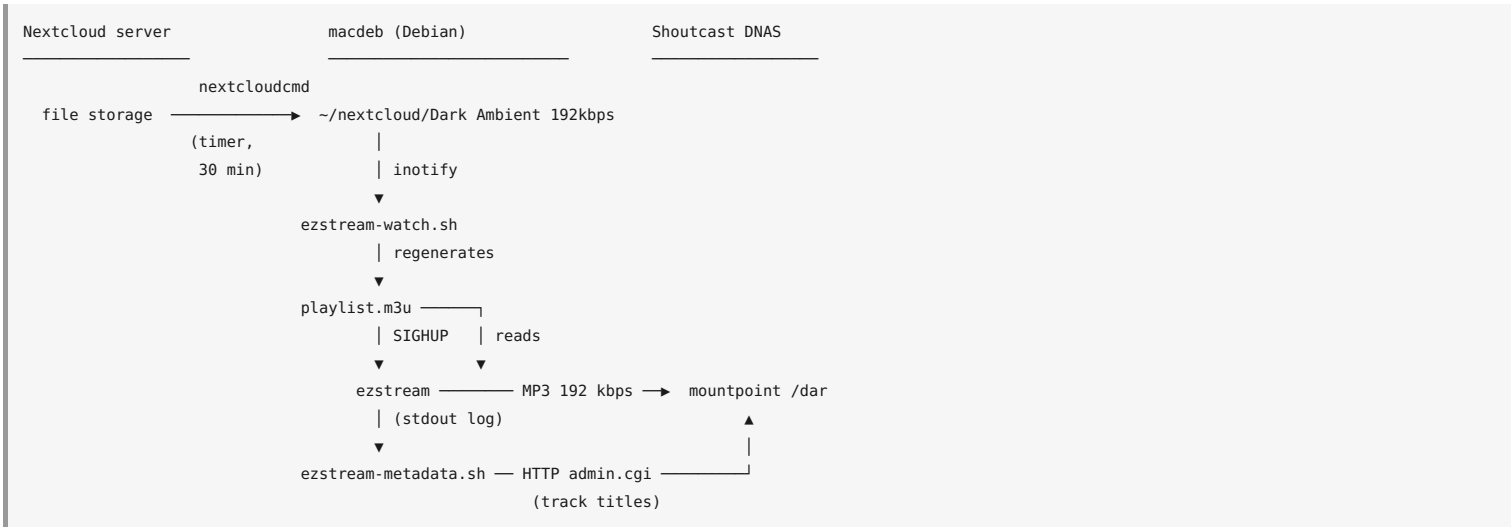
Status: production. Streaming target migrated from a local Shoutcast DNAS test server to a hosted provider (`s3.viastreaming.net:8835`).

The complete source of every script referenced below is reproduced in the appendices (§A–§E), so this document is self-contained: it can be handed to someone with no access to the host and still be sufficient to rebuild the system.

---

1. Overview

A directory tree of MP3 files is synchronised from Nextcloud to a Linux host. A watcher regenerates an M3U playlist whenever the tree changes and signals the streaming daemon to reload it. A second helper reads the daemon's log and pushes the current track title to the streaming server, because the daemon cannot do this itself against a Shoutcast server.



1.1 Design constraints discovered during implementation

These are the load-bearing facts. An automation that ignores them will produce a silent or title-less stream.

1. **Shoutcast DNAS does not accept an Icecast source client.** DNAS speaks the SHOUTcast v1 and v2 source protocols only. Setting ezstream's `<protocol>HTTP</protocol>` yields `connect: error -4: Socket error` and the process exits. `ICY` is mandatory.
2. **ezstream does not send metadata over ICY to DNAS 2.6.** Verified by packet capture: no `admin.cgi?mode=updinfo` request is ever emitted on any port, while a manual `curl` to the same endpoint succeeds. Hence the metadata bridge in §6.
3. **The metadata bridge parses ezstream's log**, so ezstream *must* run with `-vv`. Removing the verbosity flag silently breaks track titles.
4. **Stream format must match the source material exactly.** The stream is passthrough (no re-encoding). Files deviating from 192 kbps / 44100 Hz / stereo can cause DNAS to drop the source. Library normalisation is a prerequisite, not an optimisation (§8).
5. **`systemd --user` requires a login session or lingering.** Several operational commands fail with `Failed to connect to bus: No medium found` without it (§9.2).

---

2. Host and account layout

Item	Value
Host	`macdeb`, Debian
Service account	`radio`
Music root	`/home/radio/nextcloud/Dark Ambient 192kbps`
ezstream prefix	`/usr/local/bin/ezstream` (built from source, v1.0.1)
ezstream working dir	`/home/radio/ezstream-1.0.1`
Helper scripts	`/home/radio/bin/`
Secrets	`/home/radio/.config/*.env`, `/home/radio/.netrc` (mode 0600)
Library size	~4,550 tracks at time of writing

All services run as **user units** under `radio`, not system units. This keeps paths, `$HOME` and file ownership consistent and avoids `User=`/`Group=` plumbing. Lingering must be enabled so the units start at boot without a login:

```
loginctl enable-linger radio
```

---

3. Software inventory

Component	Version / source	Role
ezstream	1.0.1, built from source	Source client, reads playlist, streams MP3
libshout	bundled with ezstream build	Transport; ICY protocol
Shoutcast DNAS	2.6.1.777 (test), hosted (production)	Streaming server
nextcloudcmd	Debian package `nextcloud-desktop-cmd`	One-shot Nextcloud sync
inotify-tools	Debian package	Filesystem watching
python3-mutagen	Debian package	ID3 tag reading for metadata bridge
curl	Debian package	Metadata HTTP calls
ffmpeg	on a Windows workstation	Library normalisation (§8)

4. ezstream

4.1 Configuration — `/home/radio/ezstream-1.0.1/dar.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<ezstream>
    <servers>
        <server>
            <name>default</name>
            <protocol>ICY</protocol>          <!-- mandatory, see §1.1(1) -->
            <hostname>s3.viastreaming.net</hostname>
            <port>8835</port>
            <password>SOURCE_PASSWORD</password>
            <reconnect_attempts>5</reconnect_attempts>
            <tls>None</tls>
        </server>
    </servers>

    <streams>
        <stream>
            <name>default</name>
            <mountpoint>/dar</mountpoint>
            <format>MP3</format>
            <stream_name>Dark Ambient Radio</stream_name>
            <stream_url>https://www.darkambientradio.de</stream_url>
            <stream_genre>Ambient</stream_genre>
            <stream_description>Muzak for the final elevation</stream_description>
            <stream_bitrate>192</stream_bitrate>
            <stream_channels>2</stream_channels>
            <stream_samplerate>44100</stream_samplerate>
        </stream>
    </streams>

    <intakes>
        <intake>
            <name>default</name>
            <type>playlist</type>
            <filename>/home/radio/ezstream-1.0.1/playlist.m3u</filename>
            <shuffle>Yes</shuffle>
            <stream_once>No</stream_once>
        </intake>
    </intakes>

    <metadata>
        <normalize_strings>Yes</normalize_strings>
        <no_updates>No</no_updates>
    </metadata>
</ezstream>
```

Config pitfalls fixed during setup:

- A `<filename>` element inside `<stream>` is 0.x syntax and is not valid in 1.0.x. It was present in the inherited config and pointed at a non-existent path. Removed.
- The intake `<filename>` must be **absolute**. Relative paths resolve against the process working directory, which under systemd is not the user's home.
- The file should not be world-readable — it contains the source password. ezstream warns `dar.xml: world readable` at startup. Target mode `0640`, owner `radio:radio`.

4.2 Signal behaviour

Signal	Effect
`SIGHUP`	Re-reads the playlist **after the current track ends**. With `<shuffle>Yes</shuffle>` the list is reshuffled and restarted; playback position is not preserved. With shuffle off, ezstream tries to locate the last played file in the new list and continue after it.
`SIGUSR1`	Skips the current track. Useful for testing without waiting out a 15-minute drone piece.

4.3 Unit — `~/.config/systemd/user/ezstream.service`

```
[Unit]
Description=ezstream - Dark Ambient Radio
After=network-online.target

[Service]
Type=simple
WorkingDirectory=/home/radio/ezstream-1.0.1
ExecStart=/usr/local/bin/ezstream -c /home/radio/ezstream-1.0.1/dar.xml -p /home/radio/ezstream-1.0.1/ezstream.pid -vv
ExecReload=/bin/kill -HUP $MAINPID
Restart=always
RestartSec=5

[Install]
WantedBy=default.target
```

`-vv` is **required** — it is the data source for the metadata bridge (§6). Changing `ExecStart` needs a full `restart`; `reload` only sends SIGHUP and the process keeps its old arguments.

---

5. Playlist generation and hot reload

5.1 `mkplaylist.sh`

Scans the music root and writes an M3U file with absolute paths.

Env var	Default	Meaning
`MUSIC_ROOT`	`$HOME/nextcloud/Dark Ambient 192kbps`	Library root
`PLAYLIST`	`$HOME/ezstream-1.0.1/playlist.m3u`	Output file
`MODE`	`all`	`all` or `one-per-album`
`MIN_TRACKS`	`50`	Abort threshold
`MIN_SIZE`	`64k`	Minimum file size
`EXCLUDE_RE`	*(empty)*	Extended regex of paths to omit

Exit codes are the interface to the caller: **0** = playlist rewritten (send SIGHUP), **2** = content unchanged (do nothing), **1** = error (previous playlist left intact).

Properties that matter for automation:

- **Atomic replacement.** Writes to `PLAYLIST.tmp.XXXXXX` in the same directory, then `mv`. ezstream never observes a partially written file.
- **Safety floor.** Fewer than `MIN_TRACKS` results aborts without touching the existing playlist. This protects against an unmounted or mid-sync Nextcloud wiping the library.
- **Absolute paths.** Independent of the process working directory.
- **Filtering.** Skips dotfiles, `*.part`, `*.tmp`, `*.crdownload`, files below `MIN_SIZE`, and any filename containing a newline (which M3U cannot express).
- **Change detection** compares SHA-256 of the non-comment lines. An earlier implementation used `diff -q` against process substitutions and produced `grep: write error: Broken pipe` on stderr, because `diff` closes the pipe at the first difference. Cosmetic only, but fixed.

5.2 `ezstream-watch.sh`

Long-running watcher.

```
inotifywait -m -r -e create -e close_write -e delete -e moved_to
            -e moved_from -e move_self -e delete_self
            --exclude '(^|/)\.|\.part$|\.tmp$|\.crdownload$|~$'
```

Events are **coalesced**: the first event starts a collection phase that ends after `QUIET_SECS` (default 120) of silence. A single Nextcloud sync run produces hundreds of events — observed: 363 events collapsed into one regeneration. Only then is `mkplaylist.sh` invoked, and SIGHUP is sent only on exit code 0.

Reload is performed via `RELOAD_CMD` (`systemctl --user reload ezstream`) if set, otherwise via the PID file, otherwise via `pgrep -u <user> -x ezstream`.

The exclusion of dotfiles is not cosmetic: `nextcloudcmd` keeps its sync database as `._sync_*.db` inside the synced directory, and every write to it would otherwise trigger a regeneration.

5.3 Unit — `~/.config/systemd/user/ezstream-playlist.service`

```
[Unit]
```

```
Description=Playlist watcher for ezstream
After=ezstream.service

[Service]
Type=simple
EnvironmentFile=/home/radio/.config/ezstream-playlist.env
ExecStart=/home/radio/bin/ezstream-watch.sh
Restart=always
RestartSec=10

[Install]
WantedBy=default.target
```

Ansible note. Do not inline these as `Environment=` lines. systemd splits `Environment=` on whitespace, so `Environment=MUSIC_ROOT=/home/radio/nextcloud/Dark Ambient 192kbps` silently truncates the value at `/home/radio/nextcloud/Dark`. Either quote the whole assignment (`Environment="MUSIC_ROOT=..."`) or use an `EnvironmentFile`, where each line is taken verbatim. The `EnvironmentFile` form is preferred and is what the role should template.

`/home/radio/.config/ezstream-playlist.env`:

```
MUSIC_ROOT=/home/radio/nextcloud/Dark Ambient 192kbps
PLAYLIST=/home/radio/ezstream-1.0.1/playlist.m3u
QUIET_SECS=120
RELOAD_CMD=systemctl --user reload ezstream
```

5.4 Alternative: playlist program

ezstream supports `<type>program</type>`, where a script is executed once per track and prints a single filename. This removes the need for SIGHUP entirely and avoids the reshuffle side effect, at the cost of implementing shuffle and repeat-avoidance in the script (`next-track.sh` does this with a 300-entry history file). Not currently in use; kept as a documented option.

---

6. Metadata bridge

6.1 Why it exists

Evidence chain, recorded so nobody repeats the investigation:

1. `<SONGTITLE></SONGTITLE>` empty in `stats?sid=1` while the stream ran fine.
2. ID3 tags verified present; ezstream's own log showed correctly assembled titles (`streaming: Troum - Wrota sfer - ...`).
3. A manual `admin.cgi?mode=updinfo` call set the title successfully → server and credentials fine.
4. `tcpdump` on the source and listener ports captured the manual call but **no** call from ezstream across multiple track changes → ezstream emits nothing over ICY.
5. `<protocol>HTTP</protocol>` (Icecast source) rejected by DNAS → not an option.

Note: the DNAS w3c log (`w3clog=`) records **listener sessions only**, not `admin.cgi` calls. It is not usable as evidence for this question — a false lead during the original investigation.

6.2 `ezstream-metadata.sh`

Follows `journalctl --user -u ezstream -f -n 0 -o cat`, matches lines containing `streaming:`, and for each new track:

1. Extracts the file path from the trailing `" (/...)"` group. Parsing splits on the **first** `" (/"` occurrence, because both titles and directory names contain parentheses (e.g. `Tjukurrpa (part one: harmonies)` inside `troum - tjukurrpa (part one harmonies)/`).
2. Reads `artist` and `title` via mutagen and builds `Artist - Title`. This is deliberate: ezstream's log line appends the album, which is not wanted in the player display. Falls back to the log string if mutagen is unavailable or the tags are empty.
3. Sends `GET /admin.cgi?sid=<sid>&mode=updinfo&pass=<pass>&song=<urlencoded>`. URL encoding is delegated to `curl --get --data-urlencode`.
4. Suppresses duplicates — journald delivers each ezstream line twice (stderr and syslog). On HTTP failure the title is *not* recorded as sent, so the duplicate line triggers a retry; this is why failures appear twice in the log.

`/home/radio/.config/ezstream-metadata.env` (mode 0600):

```
DNAS_URL=http://s3.viastreaming.net:8835
DNAS_SID=1
DNAS_PASS=STREAM_PASSWORD
```

The port is part of `DNAS_URL`. Omitting it sends the request to port 80 and yields `HTTP 404` from the provider's web server — the exact failure seen after the production migration.

6.3 Unit — `~/.config/systemd/user/ezstream-metadata.service`

```
[Unit]
Description=Metadata bridge ezstream -> Shoutcast DNAS
After=ezstream.service
BindsTo=ezstream.service

[Service]
```

```
Type=simple
EnvironmentFile=/home/radio/.config/ezstream-metadata.env
ExecStart=/home/radio/bin/ezstream-metadata.sh
Restart=always
RestartSec=10

[Install]
WantedBy=default.target
```

`BindsTo=` stops the bridge when ezstream stops; without it the bridge would tail a dead unit.

6.4 Known weakness

The bridge depends on the exact wording of ezstream's log line. An ezstream upgrade could change it. Suggested monitoring:

```
journalctl --user -u ezstream-metadata --since -1h | grep -c 'gesetzt'
```

A count of zero over a period with track changes indicates a broken parser.

---

7. Nextcloud synchronisation

One-shot pull via `nextcloudcmd`, driven by a timer rather than a daemon.

Credentials go in `/home/radio/.netrc` (mode 0600), never on the command line where `ps` would expose them:

```
machine thomiel.spdns.de login <user> password <app-password>
```

Use a Nextcloud **app password**, which can be revoked individually and works with two-factor authentication enabled.

`~/.config/systemd/user/nextcloud-sync.service`:

```
[Unit]
Description=Nextcloud sync for the radio library

[Service]
Type=oneshot
TimeoutStartSec=2h
ExecStart=/usr/bin/flock -n /home/radio/.nextcloudcmd.lock \
    /usr/bin/nextcloudcmd -n --non-interactive --silent \
    /home/radio/nextcloud https://thomiel.spdns.de/nextcloud
```

`~/.config/systemd/user/nextcloud-sync.timer`:

```
[Unit]
Description=Run the Nextcloud sync periodically

[Timer]
OnBootSec=5min
OnUnitInactiveSec=30min
AccuracySec=1min

[Install]
WantedBy=timers.target
```

`OnUnitInactiveSec` measures from the **end** of the previous run, so a long sync cannot cause overlapping executions; `flock` is belt-and-braces.

First run must be performed interactively — a self-signed or otherwise untrusted certificate would make the non-interactive run abort.

---

8. Library normalisation

The stream is passthrough. Files whose bitrate, sample rate or channel count deviate from the announced 192 kbps / 44100 Hz / stereo risk the server dropping the source. Audit the library with mutagen:

```
from mutagen.mp3 import MP3
for line in open('/home/radio/ezstream-1.0.1/playlist.m3u'):
    p = line.strip()
    if not p or p.startswith('#'):
        continue
    i = MP3(p).info
    if round(i.bitrate/1000) != 192 or i.sample_rate != 44100 or i.channels != 2:
        print(round(i.bitrate/1000), i.sample_rate, i.channels, p)
```

The initial audit found 97 deviating files (mostly 48000 Hz). They were re-encoded with ffmpeg on the Windows workstation that holds the master copy of the Nextcloud folder (`C:\Users\mielk\Nextcloud2\Dark Ambient 192kbps`), using `recode-44100.ps1`:

- `-c:a libmp3lame -b:a 192k -ar 44100 -ac 2`
- `-map_metadata 0 -id3v2_version 3 -write_id3v1 1`, cover art copied with `-map 0:v? -c:v copy`, automatic retry with `-vn` if that fails
- originals moved to a backup tree before replacement

Two operational lessons, both relevant to any future automation:

- **Keep backups and logs outside the synced folder.** Writing the log inside the Nextcloud directory caused `Add-Content` to fail with a sharing violation because the desktop client held the file open, and a backup tree placed there would duplicate the entire corpus onto the server — and back into the playlist.
- Re-encoding is lossy-to-lossy. Prefer normalising from lossless masters where they exist.

---

9. Operations

9.1 Verifying a playlist reload

Cheapest check — the playlist's access time:

```
stat -c 'mtime: %y%natime: %x' /home/radio/ezstream-1.0.1/playlist.m3u
```

Under `relatime` (Debian default) the atime is updated on the first read after a write, so `atime > mtime` proves ezstream read the new file. Confirm the mount is not `noatime` with `findmnt -no OPTIONS -T <path>`.

Definitive check:

```
strace -f -e trace=openat -p "$(cat /home/radio/ezstream-1.0.1/ezstream.pid)" \
  | grep playlist.m3u
```

Note that `systemctl --user reload` reports success even if the process ignored the signal — it is not proof of anything.

9.2 `systemctl --user` from another account

`systemctl --user` needs `XDG_RUNTIME_DIR` and a D-Bus address. An SSH login without `libpam-systemd`, or `su`/`sudo -u`, does not provide them and fails with `Failed to connect to bus: No medium found`. Either fix PAM (`apt install libpam-systemd`, `UsePAM yes` in `sshd_config`) or export:

```
export XDG_RUNTIME_DIR=/run/user/$(id -u radio)
export DBUS_SESSION_BUS_ADDRESS=unix:path=$XDG_RUNTIME_DIR/bus
```

For Ansible this means user-unit tasks must either run with a proper login or set these variables explicitly in the task environment.

9.3 inotify watch limits

Each directory consumes one watch.

```
find /home/radio/nextcloud -type d | wc -l
cat /proc/sys/fs/inotify/max_user_watches
```

If the count approaches the limit:

```
# /etc/sysctl.d/90-inotify.conf
fs.inotify.max_user_watches=524288
```

This is a system-level change and belongs in a separate role or a `sysctl` task, not in the user-scoped role.

9.4 Health checks

Check	Command	Expected
Stream up	`curl -s '<server>/stats?sid=1'`	`STREAMSTATUS` = 1
Title flowing	same, `SONGTITLE`	current track, changes over time
Single source	`pgrep -c -x ezstream`	1
Playlist fresh	`grep -cv '^#' playlist.m3u`	matches library size
Watcher alive	`systemctl --user is-active ezstream-playlist`	`active`

Two source clients on one mountpoint produce alternating `send: Socket error` / `reconnect: success` cycles with tracks lasting about a second. If that pattern appears, check `pgrep -c -x ezstream` first.

---

10. Notes for the Ansible implementation

10.1 Suggested role decomposition

Role	Scope
`common_radio_user`	user `radio`, lingering, `~/bin`, `~/.config`, sysctl for inotify
`nextcloud_sync`	package, `.netrc`, service + timer
`ezstream`	build/install, `dar.xml`, PID file location, unit
`radio_playlist`	`mkplaylist.sh`, `ezstream-watch.sh`, env file, unit
`radio_metadata`	`ezstream-metadata.sh`, mutagen, env file, unit

Separating playlist from metadata is worthwhile: the metadata bridge is the component most likely to be replaced (e.g. if the stream moves to Icecast, it disappears entirely).

10.2 Variables to expose

```
radio_user: radio
radio_music_root: "/home/{{ radio_user }}/nextcloud/Dark Ambient 192kbps"
radio_ezstream_dir: "/home/{{ radio_user }}/ezstream-1.0.1"
radio_playlist_path: "{{ radio_ezstream_dir }}/playlist.m3u"
radio_playlist_mode: all          # all | one-per-album
radio_min_tracks: 50
radio_quiet_secs: 120

radio_stream_host: s3.viastreaming.net
radio_stream_port: 8835
radio_stream_mount: /dar
radio_stream_bitrate: 192
radio_stream_samplerate: 44100
radio_stream_channels: 2
radio_stream_name: "Dark Ambient Radio"

radio_dnas_sid: 1
# vault
radio_source_password: !vault |
radio_dnas_password: !vault |
radio_nextcloud_app_password: !vault |
```

`radio_stream_*` values appear in three places — `dar.xml`, the library audit and the re-encode profile. Deriving all three from the same variables prevents the passthrough mismatch described in §8.

10.3 Idempotency and handlers

- Templating `dar.xml` → handler `restart ezstream` (not reload; ICY reconnect is required for server changes).
- Templating the env files → handler `restart` for the respective unit.
- `mkplaylist.sh` itself is idempotent and safe to run in check mode via `--check`-equivalent (`MODE=all` plus exit code 2 signals "no change"). Note that `MODE=one-per-album` is **not** idempotent — it picks randomly on every run.
- Enabling user units requires `systemd: scope: user` plus the environment from §9.2, or `become_user` with a proper login shell.

10.4 Secrets

Three secrets: the source password (in `dar.xml`), the metadata password (env file), and the Nextcloud app password (`.netrc`). All three should come from Ansible Vault. File modes: `dar.xml` 0640, env files 0600, `.netrc` 0600, all owned by `radio`.

10.5 Things deliberately not automated

- ffmpeg normalisation runs on a Windows workstation against the master copy of the library and is a manual, occasional task.
- Shoutcast DNAS itself is provider-hosted in production; only the test instance had a local `sc_serv.conf`. If a local server is ever reintroduced, note that `requirestreamconfigs=1` requires a `sid` parameter on every `admin.cgi` call.
- The SHOUTcast directory listing (`STREAMLISTEDERROR` observed on the test server) was never configured; it needs an authhash and is out of scope.

Appendices — script sources

All shell scripts are installed to `/home/radio/bin/` with mode `0755`, owner `radio:radio`. They take their configuration from environment variables with the defaults shown at the top of each file, so an Ansible role can template the `EnvironmentFile` and leave the scripts themselves byte-identical across hosts.

Appendix A — `mkplaylist.sh`

Playlist generator (§5.1). Exit codes are the contract: 0 = rewritten, 2 = unchanged, 1 = error.

```
#!/usr/bin/env bash
#
# mkplaylist.sh -- erzeugt aus einer Verzeichnisstruktur eine M3U-Playlist fuer ezstream
#
# Exit-Codes:
```

```
#   0  Playlist wurde neu geschrieben  -> ezstream sollte SIGHUP bekommen
#   2  Inhalt unveraendert             -> nichts zu tun
#   1  Fehler / Sicherheitsabbruch     -> alte Playlist bleibt unangetastet
#
set -euo pipefail

# ------------------------------------------------------------------ Konfiguration
MUSIC_ROOT="${MUSIC_ROOT:-$HOME/nextcloud/Dark Ambient 192kbps}"
PLAYLIST="${PLAYLIST:-$HOME/ezstream-1.0.1/playlist.m3u}"

# all           = alle Titel aller Alben
# one-per-album = pro Albumverzeichnis genau ein zufaelliger Titel
MODE="${MODE:-all}"

# Sicherheitsnetz: weniger Titel als das -> Abbruch, alte Playlist bleibt stehen.
# (Schuetzt davor, dass eine halb synchronisierte oder nicht gemountete
#  Nextcloud die Playlist leer raeumt.)
MIN_TRACKS="${MIN_TRACKS:-50}"

# Minimale Dateigroesse; filtert abgebrochene Downloads heraus.
MIN_SIZE="${MIN_SIZE:-64k}"

# Optionaler erweiterter regulaerer Ausdruck fuer auszuschliessende Pfade,
# z. B. EXCLUDE_RE='/!Free/|/Hoerbuecher/'
EXCLUDE_RE="${EXCLUDE_RE:-}"
# --------------------------------------------------------------------------------

umask 022

log() { printf '%(%F %T)T mkplaylist: %s\n' -1 "$*" >&2; }

[[ -d "$MUSIC_ROOT" ]] || { log "Abbruch: Musikverzeichnis '$MUSIC_ROOT' existiert nicht"; exit 1; }
dest_dir=$(dirname -- "$PLAYLIST")
[[ -d "$dest_dir" ]]   || { log "Abbruch: Zielverzeichnis '$dest_dir' existiert nicht"; exit 1; }

# Temporaerdatei im Zielverzeichnis -> spaeteres mv ist ein atomarer rename().
# ezstream sieht die Playlist dadurch nie halb geschrieben.
tmp=$(mktemp -- "$PLAYLIST.tmp.XXXXXX")
trap 'rm -f -- "$tmp"' EXIT

# Alle Kandidaten NUL-getrennt und stabil sortiert einsammeln.
collect() {
    find "$MUSIC_ROOT" -type f -iname '*.mp3' \
        ! -name '.*' ! -iname '*.part' ! -iname '*.tmp' ! -iname '*.crdownload' \
        -size "+$MIN_SIZE" -print0 \
    | LC_ALL=C sort -z
}

# Ausschlussmuster anwenden und Dateinamen mit Zeilenumbruch aussortieren
# (die wuerden das M3U-Format zerstoeren).
filter() {
    local f
    while IFS= read -r -d '' f; do
        if [[ $f == *$'\n'* ]]; then
            log "uebersprungen (Zeilenumbruch im Dateinamen): ${f//$'\n'/ }"
            continue
        fi
        if [[ -n $EXCLUDE_RE ]] && [[ $f =~ $EXCLUDE_RE ]]; then
            continue
        fi
        printf '%s\0' "$f"
    done
}

# Pro Albumverzeichnis genau einen zufaelligen Titel ausgeben.
# Die Eingabe ist sortiert, Dateien eines Verzeichnisses liegen also beieinander.
pick_one_per_album() {
    local prev='' f d
    local -a group=()
    while IFS= read -r -d '' f; do
        d=${f%/*}
        if [[ $d != "$prev" ]]; then
            if (( ${#group[@]} > 0 )); then
                printf '%s\n' "${group[RANDOM % ${#group[@]}]}"
            fi
            group=()
            prev=$d
        fi
        group+=("$f")
```

```
    done
    if (( ${#group[@]} > 0 )); then
        printf '%s\n' "${group[RANDOM % ${#group[@]}]}"
    fi
}

case "$MODE" in
    all|one-per-album) ;;
    *) log "Abbruch: unbekannter MODE '$MODE' (erlaubt: all, one-per-album)"; exit 1 ;;
esac

{
    # Kommentarzeilen ignoriert ezstream.
    printf '# ezstream-Playlist, erzeugt am %(%F %T)T\n' -1
    printf '# Quelle: %s\n' "$MUSIC_ROOT"
    printf '# Modus:  %s\n' "$MODE"
    if [[ $MODE == all ]]; then
        collect | filter | tr '\0' '\n'
    else
        collect | filter | pick_one_per_album
    fi
} > "$tmp"

count=$(grep -cv '^#' -- "$tmp" || true)
count=${count:-0}

if (( count < MIN_TRACKS )); then
    log "Abbruch: nur $count Titel gefunden (MIN_TRACKS=$MIN_TRACKS) -- bestehende Playlist bleibt unveraendert"
    exit 1
fi

# Pruefsumme ueber den Inhalt ohne Kommentarzeilen. Bewusst NICHT mit
# "diff -q <(...) <(...)": diff schliesst die Pipes beim ersten Unterschied,
# das nachschiebende grep bekommt SIGPIPE und meldet "write error: Broken pipe".
filtered_sum() {
    { grep -v '^#' -- "$1" 2>/dev/null || true; } | sha256sum | cut -d' ' -f1
}

if [[ -f $PLAYLIST ]] && [[ $(filtered_sum "$PLAYLIST") == $(filtered_sum "$tmp") ]]; then
    log "unveraendert ($count Titel)"
    exit 2
fi

mv -f -- "$tmp" "$PLAYLIST"
trap - EXIT
chmod 0644 -- "$PLAYLIST"
log "geschrieben: $PLAYLIST ($count Titel)"
exit 0
```

Appendix B — `ezstream-watch.sh`

Filesystem watcher with event coalescing (§5.2). Long-running; supervised by systemd.

```
#!/usr/bin/env bash
#
# ezstream-watch.sh -- ueberwacht das Musikverzeichnis, erzeugt die Playlist neu
#                      und schickt ezstream ein SIGHUP.
#
# Benoetigt: inotify-tools  (apt install inotify-tools)
#
set -uo pipefail

# ------------------------------------------------------------------ Konfiguration
MUSIC_ROOT="${MUSIC_ROOT:-$HOME/nextcloud/Dark Ambient 192kbps}"
PLAYLIST="${PLAYLIST:-$HOME/ezstream-1.0.1/playlist.m3u}"
MKPLAYLIST="${MKPLAYLIST:-$HOME/bin/mkplaylist.sh}"

# Ruhezeit: erst wenn so viele Sekunden lang kein Ereignis mehr kam, wird
# neu erzeugt. Verhindert hunderte Neuerzeugungen waehrend eines Nextcloud-Syncs.
QUIET_SECS="${QUIET_SECS:-120}"

# Wie ezstream zum Neueinlesen bewegt wird -- eines von beiden:
#   PIDFILE     : ezstream muss mit "-p <pidfile>" gestartet worden sein
#   RELOAD_CMD  : z. B. "systemctl --user reload ezstream" (hat Vorrang)
PIDFILE="${PIDFILE:-$HOME/ezstream-1.0.1/ezstream.pid}"
RELOAD_CMD="${RELOAD_CMD:-}"
# ------------------------------------------------------------------------------
```

```
export MUSIC_ROOT PLAYLIST

log() { printf '%(%F %T)T ezstream-watch: %s\n' -1 "$*" >&2; }

command -v inotifywait >/dev/null 2>&1 || {
    log "Abbruch: inotifywait fehlt -- 'sudo apt install inotify-tools'"; exit 1; }
[[ -x $MKPLAYLIST ]] || { log "Abbruch: '$MKPLAYLIST' fehlt oder ist nicht ausfuehrbar"; exit 1; }
[[ -d $MUSIC_ROOT ]] || { log "Abbruch: '$MUSIC_ROOT' existiert nicht"; exit 1; }

reload_ezstream() {
    if [[ -n $RELOAD_CMD ]]; then
        if $RELOAD_CMD; then log "Reload ausgefuehrt: $RELOAD_CMD"
        else log "Reload fehlgeschlagen: $RELOAD_CMD"; fi
        return
    fi

    local pid=''
    [[ -r $PIDFILE ]] && pid=$(head -n1 -- "$PIDFILE" | tr -dc '0-9')
    if [[ -z $pid ]]; then
        pid=$(pgrep -u "$(id -un)" -o -x ezstream 2>/dev/null || true)
    fi

    if [[ -n $pid ]] && kill -0 "$pid" 2>/dev/null; then
        kill -HUP "$pid" && log "SIGHUP an ezstream gesendet (PID $pid)"
    else
        log "kein laufender ezstream gefunden -- Playlist wird beim naechsten Start gelesen"
    fi
}

log "beobachte '$MUSIC_ROOT' (Ruhezeit ${QUIET_SECS}s)"

# Beim Start einmal aufraeumen, falls sich waehrend eines Ausfalls etwas geaendert hat.
"$MKPLAYLIST"; rc=$?
[[ $rc -eq 0 ]] && reload_ezstream

inotifywait -m -r -q \
    -e create -e close_write -e delete -e moved_to -e moved_from \
    -e move_self -e delete_self \
    --exclude '(^|/)\.|\.part$|\.tmp$|\.crdownload$|~$' \
    --format '%e %w%f' \
    -- "$MUSIC_ROOT" |
while IFS= read -r first_event; do
    # Ab hier Ereignisse einsammeln, bis QUIET_SECS lang Ruhe herrscht.
    n=1
    while IFS= read -r -t "$QUIET_SECS" _more; do
        n=$((n + 1))
    done

    log "$n Ereignis(se), Ruhe eingekehrt (zuletzt: $first_event)"

    "$MKPLAYLIST"; rc=$?
    case $rc in
        0) reload_ezstream ;;
        2) : ;;   # Inhalt unveraendert, kein SIGHUP noetig
        *) log "mkplaylist.sh meldete Fehler (Exit $rc) -- Playlist unveraendert" ;;
    esac
done

log "inotifywait beendet -- Skript endet"
exit 1
```

Appendix C — `ezstream-metadata.sh`

Metadata bridge (§6.2). Requires ezstream to run with `-vv`.

```
#!/usr/bin/env bash
#
# ezstream-metadata.sh -- schickt den laufenden Titel an Shoutcast DNAS.
#
# Hintergrund: ezstream sendet mit <protocol>ICY</protocol> keine Metadaten an
# DNAS 2.6, und das Icecast-Quellprotokoll (<protocol>HTTP</protocol>) kennt
# DNAS nicht. Dieses Skript verfolgt deshalb das Journal von ezstream (das mit
# "-vv" jeden Titelwechsel protokolliert) und setzt den Titel selbst per
# admin.cgi -- der Weg, der sich manuell als funktionierend erwiesen hat.
#
# Voraussetzung: ezstream laeuft mit "-vv" als systemd-User-Unit.
```

```
# Optional:      python3-mutagen (fuer sauberes "Artist - Title" aus den Tags)
#
set -uo pipefail

# ------------------------------------------------------------------ Konfiguration
DNAS_URL="${DNAS_URL:-http://127.0.0.1:8765}"
DNAS_SID="${DNAS_SID:-1}"
DNAS_PASS="${DNAS_PASS:-}"          # streampassword_1 aus der sc_serv.conf
EZ_UNIT="${EZ_UNIT:-ezstream}"      # Name der ezstream-User-Unit
# --------------------------------------------------------------------------------

log() { printf '%(%F %T)T ezstream-meta: %s\n' -1 "$*" >&2; }

[[ -n $DNAS_PASS ]] || { log "Abbruch: DNAS_PASS ist nicht gesetzt"; exit 1; }
command -v curl       >/dev/null || { log "Abbruch: curl fehlt"; exit 1; }
command -v journalctl >/dev/null || { log "Abbruch: journalctl fehlt"; exit 1; }

have_mutagen=0
if python3 -c 'import mutagen' 2>/dev/null; then
    have_mutagen=1
else
    log "Hinweis: python3-mutagen fehlt -- es wird der Titel aus dem Log benutzt"
    log "         (enthaelt zusaetzlich das Album). 'apt install python3-mutagen'"
fi

# "Artist - Title" aus den ID3-Tags lesen; leere Ausgabe bei Problemen.
tags_of() {
    python3 - "$1" 2>/dev/null <<'PY'
import sys
try:
    from mutagen import File
    f = File(sys.argv[1], easy=True) or {}
    a = (f.get('artist') or [''])[0].strip()
    t = (f.get('title')  or [''])[0].strip()
except Exception:
    a = t = ''
print(f"{a} - {t}" if a and t else (a or t))
PY
}

# Titel an DNAS uebergeben. curl uebernimmt das URL-Encoding.
send_title() {
    curl -s -o /dev/null -w '%{http_code}' --max-time 10 --get \
        --data-urlencode "song=$1" \
        -A "ezstream-metadata" \
        "$DNAS_URL/admin.cgi?sid=$DNAS_SID&mode=updinfo&pass=$DNAS_PASS"
}

log "verfolge Unit '$EZ_UNIT', Ziel $DNAS_URL (sid=$DNAS_SID)"

last=''
# -n 0 : nur neue Zeilen, -o cat : ohne Zeitstempel-Praefix
journalctl --user -u "$EZ_UNIT" -f -n 0 -o cat |
while IFS= read -r line; do
    [[ $line == *"streaming: "* ]] || continue

    rest=${line#*streaming: }
    path=''
    fallback=$rest
    if [[ $rest == *" (/"* ]]; then
        fallback=${rest%% (/*}      # alles vor der ersten " (/"-Gruppe
        path=/${rest#* (/}          # der Dateipfad
        path=${path%)}
    fi

    title=''
    if (( have_mutagen )) && [[ -n $path && -f $path ]]; then
        title=$(tags_of "$path")
    fi
    [[ -n $title ]] || title=$fallback

    # Journal liefert jede Zeile doppelt (stderr + syslog) -> entdoppeln
    if [[ -z $title || $title == "$last" ]]; then
        continue
    fi

    code=$(send_title "$title")
    if [[ $code == 200 ]]; then
        log "gesetzt: $title"
```

```
        last=$title
    else
        log "FEHLER (HTTP $code) bei: $title"
    fi
done

log "journalctl beendet -- Skript endet"
exit 1
```

Appendix D — `next-track.sh`

Optional playlist program (§5.4). Not currently deployed; documented as the SIGHUP-free alternative.

```
#!/usr/bin/env bash
#
# next-track.sh -- optionale Alternative: "Playlist Program" fuer ezstream.
#
# ezstream ruft dieses Skript vor jedem Titel genau einmal auf; es muss genau
# eine Zeile (einen Dateinamen) ausgeben und sich beenden. Dadurch wird die
# Playlist bei JEDEM Titelwechsel frisch gelesen -- ein SIGHUP ist nicht noetig.
#
# In dar.xml:
#   <type>program</type>
#   <filename>/home/radio/bin/next-track.sh</filename>
#   <shuffle>No</shuffle>          <!-- wirkt hier nicht, das Skript mischt -->
#   <stream_once>No</stream_once>
#
# Wichtig: keine Ausgaben auf stdout ausser dem Dateinamen.
#
set -uo pipefail

PLAYLIST="${PLAYLIST:-$HOME/ezstream-1.0.1/playlist.m3u}"
HISTORY="${HISTORY:-$HOME/ezstream-1.0.1/.played}"
HISTORY_SIZE="${HISTORY_SIZE:-300}"   # so viele Titel lang keine Wiederholung
FALLBACK="${FALLBACK:-$HOME/ezstream-1.0.1/dar-emergency-playlist}"

pick_from() {
    # Kommentare und Leerzeilen raus, existierende Dateien behalten wir bewusst
    # ungeprueft (spart I/O); ezstream ueberspringt fehlende Dateien.
    grep -v -e '^#' -e '^[[:space:]]*$' -- "$1" 2>/dev/null | shuf -n 1
}

src="$PLAYLIST"
[[ -s $src ]] || src="$FALLBACK"
[[ -s $src ]] || exit 1

track=''
for _ in $(seq 1 30); do
    candidate=$(pick_from "$src")
    [[ -n $candidate ]] || break
    if [[ -f $HISTORY ]] && grep -qxF -- "$candidate" "$HISTORY"; then
        continue          # zuletzt gespielt, neuen Versuch
    fi
    track=$candidate
    break
done
[[ -n $track ]] || track=$(pick_from "$src")
[[ -n $track ]] || exit 1

# Historie fortschreiben (nur die letzten HISTORY_SIZE Eintraege behalten)
if {
    printf '%s\n' "$track"
    if [[ -f $HISTORY ]]; then
        head -n $((HISTORY_SIZE - 1)) -- "$HISTORY"
    fi
} > "$HISTORY.new" 2>/dev/null; then
    mv -f -- "$HISTORY.new" "$HISTORY" 2>/dev/null
fi

printf '%s\n' "$track"
```

Appendix E — `recode-44100.ps1` (abridged)

Library normalisation (§8). Runs on the Windows workstation holding the master copy. The embedded file list of 97 paths is truncated below for readability; the attached file contains it in full. Save as **UTF-8 with BOM** — Windows PowerShell 5.1 reads BOM-less files as ANSI and would fail to locate paths containing `Dhyāna`, `Séance` or `Höhle`.

```
<#
.SYNOPSIS
    Kodiert eine feste Liste von MP3s auf 44100 Hz / 192 kbit/s / Stereo um.

.DESCRIPTION
    Diese Dateien weichen vom Streamformat des Dark Ambient Radio ab und koennen
    dazu fuehren, dass Shoutcast DNAS die Quelle abwirft. Das Skript kodiert sie
    mit ffmpeg neu, legt die Originale vorher in einen Backup-Ordner und ersetzt
    sie erst, wenn die neue Datei fehlerfrei erzeugt wurde.

    Metadaten (ID3) und ein eingebettetes Cover werden uebernommen.

.PARAMETER Root
    Wurzelverzeichnis der Sammlung.

.PARAMETER FFmpeg
    Pfad zu ffmpeg.exe. Standard: "ffmpeg" aus dem PATH.

.PARAMETER BackupRoot
    Ablage der Originale. Standard: <Root>\_backup_vor_recode

.PARAMETER DryRun
    Zeigt nur an, was passieren wuerde.

.PARAMETER SkipOk
    Ueberspringt Dateien, die bereits 44100 Hz / Stereo haben.

.EXAMPLE
    .\recode-44100.ps1 -DryRun
    .\recode-44100.ps1
#>

[CmdletBinding()]
param(
    [string] $Root       = 'C:\Users\mielk\Nextcloud2\Dark Ambient 192kbps',
    [string] $FFmpeg     = 'ffmpeg',
    [string] $BackupRoot = '',
    [switch] $DryRun,
    [switch] $SkipOk
)

$ErrorActionPreference = 'Stop'
[Console]::OutputEncoding = [System.Text.Encoding]::UTF8

# Default ausserhalb der Nextcloud, sonst synchronisiert sich die halbe Sammlung doppelt.
if (-not $BackupRoot) { $BackupRoot = Join-Path $env:USERPROFILE 'recode-backup' }

# ---------------------------------------------------------------------- Zielformat
$TargetRate     = 44100
$TargetChannels = 2
$TargetBitrate  = '192k'

# ---------------------------------------------------------------------- Dateiliste
# Pfade relativ zu $Root
$Files = @(
  '!Free\Rauhnacht - Nachtgeister II.mp3',
  '!Free\The Synthetic Awakening - Drone September 2018 (Extended Time Alone Mix).mp3',
  '!Free\V.A. [2011] Heathen Harvest Samhain Sampler (NUR 5 TRACKS)\04.Halgrath-DeepUnderwaterDarkestTale.mp3',
  # ... 91 further entries omitted here; see the file attachment for the full list ...
  'troum & all sides - shutun\all sides - where is my gun _.mp3'
)

# ---------------------------------------------------------------------- Vorpruefung
function Resolve-Tool([string] $Name, [string] $Given) {
    if ($Given -and (Test-Path -LiteralPath $Given)) { return (Resolve-Path -LiteralPath $Given).Path }
    $cmd = Get-Command $Given -ErrorAction SilentlyContinue
    if ($cmd) { return $cmd.Source }
    $cmd = Get-Command $Name -ErrorAction SilentlyContinue
    if ($cmd) { return $cmd.Source }
    return $null
}

$ffmpegExe = Resolve-Tool 'ffmpeg' $FFmpeg
if (-not $ffmpegExe) {
    Write-Host "ffmpeg nicht gefunden. Entweder in den PATH aufnehmen oder mit -FFmpeg angeben." -ForegroundColor Red
    exit 1
}
# ffprobe liegt normalerweise neben ffmpeg
```

```
$ffprobeExe = Join-Path (Split-Path $ffmpegExe -Parent) 'ffprobe.exe'
if (-not (Test-Path -LiteralPath $ffprobeExe)) { $ffprobeExe = Resolve-Tool 'ffprobe' 'ffprobe' }

if (-not (Test-Path -LiteralPath $Root)) {
    Write-Host "Wurzelverzeichnis nicht gefunden: $Root" -ForegroundColor Red
    exit 1
}

Write-Host "ffmpeg : $ffmpegExe"
Write-Host "ffprobe: $(if ($ffprobeExe) { $ffprobeExe } else { '(nicht gefunden, Pruefung entfaellt)' })"
Write-Host "Wurzel : $Root"
Write-Host "Backup : $BackupRoot"
Write-Host "Ziel   : $TargetRate Hz / $TargetBitrate / $TargetChannels Kanaele"
Write-Host "Dateien: $($Files.Count)"
if ($DryRun) { Write-Host "MODUS  : Trockenlauf, es wird nichts veraendert" -ForegroundColor Yellow }
Write-Host ('-' * 70)

# ---------------------------------------------------------------------- Helfer
function Get-AudioInfo([string] $Path) {
    if (-not $ffprobeExe) { return $null }
    $out = & $ffprobeExe -v error -select_streams a:0 `
        -show_entries stream=sample_rate,channels,bit_rate `
        -of default=noprint_wrappers=1:nokey=0 -- $Path 2>$null
    if ($LASTEXITCODE -ne 0 -or -not $out) { return $null }
    $info = @{}
    foreach ($line in $out) {
        if ($line -match '^(\w+)=(.*)$') { $info[$Matches[1]] = $Matches[2] }
    }
    return $info
}

function Invoke-FFmpeg([string] $In, [string] $Out, [switch] $NoCover) {
    $args = @('-hide_banner', '-loglevel', 'error', '-y', '-i', $In)
    if ($NoCover) {
        $args += @('-vn')
    } else {
        $args += @('-map', '0:a:0', '-map', '0:v?', '-c:v', 'copy')
    }
    $args += @(
        '-c:a', 'libmp3lame',
        '-b:a', $TargetBitrate,
        '-ar',  "$TargetRate",
        '-ac',  "$TargetChannels",
        '-map_metadata', '0',
        '-id3v2_version', '3',
        '-write_id3v1', '1',
        $Out
    )
    & $ffmpegExe @args 2>&1 | ForEach-Object { $_ }
    return $LASTEXITCODE
}

# ---------------------------------------------------------------------- Hauptschleife
# Protokoll NICHT in den Nextcloud-Ordner legen: der Desktop-Client haelt die
# Datei offen und Add-Content scheitert mit einer Freigabeverletzung.
$logPath  = Join-Path $env:TEMP ('recode-44100_{0}.log' -f (Get-Date -Format 'yyyyMMdd-HHmmss'))
$done     = 0
$skipped  = 0
$failed   = @()
$i        = 0

foreach ($rel in $Files) {
    $i++
    $src = Join-Path $Root $rel
    $tag = "[{0,3}/{1}]" -f $i, $Files.Count

    if (-not (Test-Path -LiteralPath $src)) {
        Write-Host "$tag FEHLT   $rel" -ForegroundColor Red
        $failed += "$rel  (Datei nicht gefunden)"
        continue
    }

    $info = Get-AudioInfo $src
    $desc = if ($info) {
        "{0} Hz, {1}ch, {2} kbit/s" -f $info['sample_rate'], $info['channels'],
            $(if ($info['bit_rate']) { [math]::Round([int]$info['bit_rate'] / 1000) } else { '?' })
    } else { 'unbekannt' }

    if ($SkipOk -and $info -and
```

```
        [int]$info['sample_rate'] -eq $TargetRate -and
        [int]$info['channels']    -eq $TargetChannels) {
        Write-Host "$tag OK      $rel  ($desc)" -ForegroundColor DarkGray
        $skipped++
        continue
    }

    Write-Host "$tag $rel" -ForegroundColor Cyan
    Write-Host "         vorher: $desc"

    if ($DryRun) { continue }

    $tmp = "$src.tmp_recode.mp3"
    if (Test-Path -LiteralPath $tmp) { Remove-Item -LiteralPath $tmp -Force }

    $rc = Invoke-FFmpeg -In $src -Out $tmp
    if ($rc -ne 0 -or -not (Test-Path -LiteralPath $tmp)) {
        # Zweiter Versuch ohne eingebettetes Cover
        if (Test-Path -LiteralPath $tmp) { Remove-Item -LiteralPath $tmp -Force }
        Write-Host "         Cover-Uebernahme fehlgeschlagen, zweiter Versuch ohne Cover" -ForegroundColor Yellow
        $rc = Invoke-FFmpeg -In $src -Out $tmp -NoCover
    }

    if ($rc -ne 0 -or -not (Test-Path -LiteralPath $tmp) -or (Get-Item -LiteralPath $tmp).Length -lt 10000) {
        Write-Host "         FEHLER (ffmpeg Exit $rc) -- Original bleibt unveraendert" -ForegroundColor Red
        if (Test-Path -LiteralPath $tmp) { Remove-Item -LiteralPath $tmp -Force }
        $failed += "$rel  (ffmpeg Exit $rc)"
        continue
    }

    # Original sichern (Verzeichnisstruktur bleibt erhalten)
    $bak    = Join-Path $BackupRoot $rel
    $bakDir = Split-Path $bak -Parent
    if (-not (Test-Path -LiteralPath $bakDir)) { New-Item -ItemType Directory -Path $bakDir -Force | Out-Null }
    Move-Item -LiteralPath $src -Destination $bak -Force

    Move-Item -LiteralPath $tmp -Destination $src -Force

    $after = Get-AudioInfo $src
    $adesc = if ($after) {
        "{0} Hz, {1}ch, {2} kbit/s" -f $after['sample_rate'], $after['channels'],
            $(if ($after['bit_rate']) { [math]::Round([int]$after['bit_rate'] / 1000) } else { '?' })
    } else { 'unbekannt' }
    Write-Host "         nachher: $adesc" -ForegroundColor Green

    try {
        Add-Content -LiteralPath $logPath -Value "$rel | vorher: $desc | nachher: $adesc" -Encoding UTF8 -ErrorAction Stop
    } catch {
        Write-Host "         (Protokollzeile konnte nicht geschrieben werden)" -ForegroundColor DarkYellow
    }
    $done++
}

Write-Host ('-' * 70)
Write-Host "Umkodiert: $done   Uebersprungen: $skipped   Fehler: $($failed.Count)"
if ($failed.Count) {
    Write-Host "Fehlerhafte Dateien:" -ForegroundColor Red
    $failed | ForEach-Object { Write-Host "  $_" -ForegroundColor Red }
}
if (-not $DryRun -and $done) {
    Write-Host "Originale liegen in: $BackupRoot"
    Write-Host "Protokoll: $logPath"
}
```