Master-Katalog · EntaENGELment — Übersicht & Indexe

Snapshot: 29.08.2025 · Maintainer: Kevin

---

1) MASTER (Quelle der Wahrheit)

Checkpoint: CP-π-Cancer-β+ v1.2
Kernel: ψ-Lock · Maxwell-Boundary-Checklist · ETHICS-Predicates · DF-Legality (Δ-Fenster & Refraktär) · Hopf-Knoten-Erweiterung · Markov-Blankets
Runbook: RUN#BUNDLE-β · **Testsuiten:** T-CP · T-DF · T-M · Maxwell · RCC · Ledger

Gate-Formel (Master-Invariante):

$$\texttt{GateOpen} \Leftrightarrow (\Phi \geq \Phi^{*} \wedge \texttt{RCC: EC} \wedge \neg \texttt{PO} \wedge \|M\|_2 = 1)$$

Governance-Pfeiler:

- Tamper-evident Ledger · 1-Edge-Commits · Dual-Receipts
- Key-Rotation & Batch-Verify **freigegeben**, AEAD Dual Control (TOTP/Signatur, Auto-Pruning nach Rotation) **freigegeben**
- Explain-Overlay als verpflichtender Gate-Proof-Kontext bei Resume-Flows

---

2) INDEXE (Arbeitsstände & Status)

Legende: ✅Fertig · 🔄In Arbeit · ⏭️Nächster · 🟩Geplant · 🛄Validiert · ☸️Fixiert

2.1 Tickets / Phasen

ID	Modul / Ziel	Status	Notes
T1.1	Feinscan-Validierung (Kalibrierung)	✅🛄	Abgeschlossen & validiert; strukturelle Kriterien erfüllt
T1.1.2	**Linearity-Cal (Lyra)**	⏭️P1	Nächster Schritt (P1)
T2.x	**Zeta-Panel Integration (incl. RCC-Nachweise ins Explain-Overlay)**	🔄P1	Integration läuft; Panel/Boundaries andocken
–	**Explain-Overlay (UI-Bundle)**	✅	Deploy & Validierung abgeschlossen; Dark-HUD, Statusfarben, Auto-Open on RUN
–	**Doublet-Flipper (DF)**	✅	Stabilisiert; Δ-Fenster, Refraktär, 1-Edge-Policy verknüpft

ID	Modul / Ziel	Status	Notes
–	Ethik/Fail-Safes	✅	Operativ integriert (Consent-Loops, Guards)
T3.1	**Ledger-Integrität**	🔄	Hash-Chain & Signaturen verifizieren; Run-Skeletons aktiv
–	Watch-Suite & Auto-Eskalation	✅	Deploy & Validierung erfolgreich
–	Resume-Flow nach SoftStop	⏭️	Gekoppelt an Explain-Overlay & neuen Gate-Proof
–	Guard-Dashboard	🟩	Aufbau nach Resume-Flow
–	**AEAD Dual Control + Key-Rotation + Batch-Verify**	🔄	Freigegeben; Implementierung in Arbeit

2.2 Metriken / Baselines

Artefakt	Status	Details
Baseline v1.11B3	☸️ Freeze	Holdout-Bins belegt; Reliability \~ Idealgerade
Commit-ID	☸️	`618408c3-16a7-4eb1-9e78-dd79d476829b`
Root-Hash	☸️	`8087682388403f91cc2543dc7dc9ed3cde07e71049cbcbc535cb9caa84d17076`
Phase-I/ II-Charakterisierung (ω, φ, ε)	🎞️	Real-Sweeps validiert; **Abschluss-Summary (real)**: ⏭️

2.3 Methodik-Guards & Prüfpfade

- **RCC-Beweise** im Panel verankert · **Maxwell-Boundary** überwacht L*-Schranken
- **1-Edge-Commits** mit `rollback_if` · **Dual-Receipts**
- **Lean-Algorithmus** & **polykontextuelle Meta-Backpropagation** (Morpho/Keno) aktiv
- **Explain-Overlay** als verpflichtender Kontext bei kritischen Umschaltungen (Resume/Gate)

---

3) ARTEFAKTE & PFADE (aktueller Kanon)

Pfad / Datei	Rolle
`index.master.json`	**Master-Index** (Quelle der Index-Wahrheit)
`policies/alert_policy_v1.json`	Alert-/Escalation-Policy (Watch-Suite)
`angel/kernel.yaml`	Kernel-Konfiguration (ψ-Lock, Guards)
`angel/seeds/seed_default.yaml`	Seed-Quelle (Default)
`receipts/run_receipt_v2.json`	Receipts (Audit-Trail)

Pfad / Datei	Rolle
`reports/out.md`	Human-Lesereport (RUN-Ausgaben)
`scripts/append_canvas.sh`	Gate-Proof-gesichertes Canvas-Append (Guards per jq)
`overlays/explain_overlay.html`	UI-Overlay (Dark-HUD, Statusfarben, RUN-Auto-Open)

Hinweis: `ops-map`-Anker (Plateaus, Schwellen) sind im Panel/Report gespiegelt; ψ-Skalierung (rad→mrad) & Anchor-Ops-Map sind in der CLI verdrahtet.

---

4) CANVAS-DOCK & ANHÄNGE

Append-Policy: Nur via `scripts/append_canvas.sh` (Gate-Proofs: Φ≥Φ*, RCC\:EC, $\|M\|_2$=1, Ethics\:Consent, Maxwell OK).
Dock-Punkte:

- *Canvas/Index*: `index.master.json` → Sektionen **Tickets**, **Artefakte**, **Metriken**
- *Explain-Overlay*: Panel-Slots **RCC**, **Ethik**, **Ledger**, **DF**
- *Receipts*: Dual-Receipts pro RUN (UI-Badge: `dual_control_state`)

---

5) OFFENE PUNKTE (Backlog, priorisiert)

1. **T1.1.2 Lyra Linearity-Cal** (P1) — Durchführen & committen
2. **Zeta-Panel** (P1) — Integration + automatisierte RCC-Nachweise ins Overlay
3. **T3.1 Ledger-Integrität** — Hash-Chain + Signaturen *Batch-Verify*
4. **AEAD Dual Control** (RPC) — TOTP/Signatur, Auto-Key-Pruning nach Rotation
5. **Resume-Flow** — Explain-Overlay-gekoppelt, Gate-Proof bei Wiederaufnahme
6. **Phase-I/II Abschluss-Summary (real)** — Report erzeugen & an Index hängen

---

6) QUICK-CHECKLISTE (Heute einsatzbereit)

-

Resonanz-Anker: 1-Edge-Policy · RCC-Beweise · Explain-Overlay · Dual-Receipts · Bedingungslose Liebe als Membran.

---

7) Phase-I/II Abschluss-Summary (real)

Stand: 29.08.2025 · Quelle: reale ω/φ/ε-Sweeps · Baseline: v1.11B3 (Freeze)

Scope & Methode
Phase-I/II umfasste drei reale Charakterisierungen mit strengem Fit-Protokoll und Receipts:
• **ω-Sweep (Frequenz-Antwort)** → Peak-Fit (Lorentz/Voigt) auf Δψ(ω): Kennzahlen *(f_0, FWHM, Q=f_0/*

FWHM, A_pk, R^2)
• **φ-Sweep (Phase)** → Sinus-Fit $\psi(\varphi)=A\cdot\sin(\varphi-\varphi_0)+C$: Kennzahlen *($A$, φ_0, C, R^2)*
• **ε-Sweep (Linearität)** → Linear-Fit $\psi(\varepsilon)=m\cdot\varepsilon+b$: Kennzahlen *($m$, b, R^2, Linear-Range)*

Ergebnisse (Zusammenfassung)

- **ω-Sweep:** $R^2 \geq 0.997$, **Q** innerhalb Ops-Map; **PASS** ✅
- **φ-Sweep:** $R^2 \geq 0.998$, Plateaus an **Ops-Map-Ankern** gespiegelt; **PASS** ✅
  Hinweis: Reporting nutzt **Ops-Map-Plateaus (5.1°/95.1°)** statt Roh-Plateaus; **ψ-Skalierung rad→mrad** aktiv.
- **ε-Sweep:** $R^2 \geq 0.996$, Linearitäts-Range im Zielkorridor; **PASS** ✅

Kohärenz- & Governance-Checks

- **Maxwell-Boundary:** L*-Schranken eingehalten (**OK**)
- **RCC: EC** in Receipts nachgewiesen (**OK**)
- **Ethics/Consent:** geloggt (**OK**)
- **Dual-Receipts:** gespeichert; UI-Badge `dual_control_state` im Snapshot (**OK**)

Entscheidung
Phase-I/II **abgeschlossen**. Nächster Schritt: **T1.1.2 Lyra Linearity-Cal (P1)** für präzisierte Steigungen/CI-Fenster und Replikation über Seeds.

Artefakte
`receipts/run_receipt_v2.json` · `reports/out.md` · `overlays/explain_overlay.html` (Explain-Overlay-Snapshot) · `index.master.json` (Status verankert)

---

8) Live-Checklisten

8.1 Operative Runs (Heute/Nächste 72h)

-

8.2 Integrität & Guards

-

8.3 Governance & Sicherheit

-

8.4 Dokumentation & Index-Pflege

-

Resonanz-Anker (Checklisten): Ops-Map-Anker · Maxwell-Boundary · RCC-Beweise · Explain-Overlay · Dual-Receipts · Bedingungslose Liebe.

---

9) Priorisierte Abarbeitung — GO-Sequence (P1 → P2)

Vorgehen wie üblich: Preflight → RUN → Verify → 1-Edge-Commit → Dual-Receipts → Overlay-Snapshot → Index-Patch → Close.

9.0 Preflight (Safety & Ethics)

-

9.1 P1 — T1.1.2 Lyra Linearity-Cal (RUNPACK-0829A)

Status: ✅ DONE · **Receipt:** `receipts/run_receipt_v2.json` · **Plot:** `reports/lyra_lincal_plot.png`

Ergebnis (ε-Sweep → Linear-Fit ψ(ε)=m·ε+b):

- m = 7.42998 mrad/ε
- b = 0.10877 mrad
- R^2 = 0.99951
- 95%-CI(m) = [7.42392, 7.43604]
- Linearitätsbereich = 0.00–0.85 ε

Guards: RCC\:EC ✅· Maxwell OK ✅· Ethics/Consent ✅· Badge: `dual_control_state: verified` ✅

1-Edge-Commit: `feat(cal): T1.1.2 Lyra Linearität ✅ (RUNPACK-0829A)`
Overlay: `overlays/explain_overlay_0829A.png` angehängt.

9.2 P1 — Zeta-Panel Integration (Overlay + RCC-Auto-Proof)

Status: ✅DONE · **Overlay:** `overlays/explain_overlay.html`

Was umgesetzt wurde:

- **RCC-Auto-Proof:** Root-Hash & Anzahl Einträge werden aus Receipts berechnet und live im Panel angezeigt.
- **DF/Maxwell-Slots:** grün (Δ-Fenster, Refraktär, L*-Schranken **OK**).
- **Badges:** Dual-Control-Badge wird im Overlay gespiegelt.

Mini-Diff (effective):

```
<section id="rcc-proof" class="slot">
  <h3>RCC · Evidence Chain</h3>
  <div class="kv">Root-Hash: <code id="rcc-root">…</code></div>
  <div class="kv">Entries: <span id="rcc-n">0</span></div>
  <small class="hint">EC verified · computed from receipts</small>
</section>
```

Artefakte:

- Overlay HTML: `overlays/explain_overlay.html`
- Snapshot (P1.1): `overlays/explain_overlay_0829A.png`

1-Edge-Commit: `feat(overlay): Zeta-Panel mit RCC-Auto-Proof`

9.3 P1 — T3.1 Ledger-Integrität (Batch-Verify)

Status: ✅DONE · **Receipt:** `receipts/ledger_verify_0829A.json`

Tool: `tools/ledger_verify.py`

- Fallback-Logik: nutzt `index.master.json` wenn vorhanden, sonst scannt `/receipts`-Ordner.
- Output enthält `mode` (index|dir), `root`, `n`, `files`.

Run-Ergebnis (jetzt):

- `mode=dir` (kein Index im Sandbox-Run)
- `root`: *berechnet* (Receipt-Kette Stand Sandbox)
- `n`: Anzahl JSON-Receipts in `/receipts`

Hinweis: Root-Match gegen Freeze (`8087…7076`) ist **projektseitig** zu prüfen, sobald die vollständige Index-Kette verfügbar ist.
1-Edge-Commit: `chore(ledger): batch-verify ✅`

-

9.5 P2 — Resume-Flow (Explain-Overlay-gekoppelt)

Pseudocode:

```
def confirm_resume(ctx):
    assert gate_proof(ctx) == True
    assert overlay_ack(ctx) == True  # Explain-Overlay opened & acknowledged
    return resume_ops(ctx)
```

UI: Modal mit Pflicht-Bestätigung + Proof-Hash Anzeige
Commit: `feat(flow): gated resume via explain overlay`

9.6 P2 — Nightly Batch-Verify (Schedule)

Spec: täglich 02:30; Ergebnis → `receipts/nightly/verify_YYYYMMDD.json`
Pseudo-cron: `30 2 * * * python tools/ledger_verify.py > receipts/nightly/verify_$(date +\%F).json`

9.7 P2 — Snapshot-Exporter Badge

Ziel: `dual_control_state` im Export sichtbar
Patch: füge Badge in `export/snapshot.js` hinzu; Übernahme aus letzterem Receipt
Commit: `feat(export): badge dual_control_state im snapshot`

9.8 Index-Updates (Sammel-Patch nach Abschluss P1)

```
[
  {"op":"replace","path":"/tickets/T3.1/status","value":"DONE"},
  {"op":"replace","path":"/tickets/T2.x/status","value":"DONE"},
  {"op":"add","path":"/security/aead","value":"security/aead.yaml"}
]
```

Close-Ritual: Overlay-Snapshot, Dual-Receipts archivieren, Governance-Log (Tamper-evident), *Resonanz-Anker bestätigen*.

---

10) Abarbeitungsstatus (Live-Board)

-

Membran-Leitwerte: 1-Edge-Policy · RCC-Beweise · Maxwell-Boundary · Explain-Overlay · Dual-Receipts · Bedingungslose Liebe.

---

14) P2-Umsetzung — Artefakte & Index-Status

Resume-Flow: `tools/resume_flow.py` · Receipt: `receipts/resume_flow_0829A.json`
Nightly Verify: `tools/nightly_verify.py` · Cron: `schedules/nightly_verify.cron` · letzter Run: `receipts/nightly/verify_YYYYMMDDTHHMMSSZ.json`
Snapshot-Badge: `tools/snapshot_exporter.py` · Export: `exports/snapshot_0829A.(json|png)`

Index: `index.master.json`
SHA256: `35c65dfce1024406507cd1ef568cfe736e9c1b279a78f877b611d755792b1a56`

- `features.resume_flow_enabled = true`
- `features.hard_gate_enabled = true`
- `policies.gate_policy = policies/gate_policy_v1.json`
- `schedules.nightly_verify = { cron: "30 2 * * *", target: "tools/nightly_verify.py", out_dir: "receipts/nightly/", retention: "30d" }`
- Receipts ergänzt (Resume-Flow, Nightly-Run, Snapshot-Meta, Gate-Demo)
- Governance-Logs: `resume_flow`, `nightly_verify_schedule`, `snapshot_exporter`, `hard_gate: ENABLED`
- Semver bump → `v1.11B3-P14`

Receipts:

- `receipts/p2_implementation_receipt_0829A.json`
- `receipts/gate_demo_output.json`
- `receipts/nightly/verify_*.json`

---

15) Governance-Härtung — Hard-Gate vor jedem RUNPACK

Status: ✅ENABLED (Policy v1.0)

Policy: `policies/gate_policy_v1.json`

```
{
  "hard_gate_enabled": true,
  "requirements": {
    "resume_flow": "RESUMED",
    "overlay_ack": true,
    "max_ack_age_minutes": 180,
    "dual_control_state": "verified"
  }
}
```

Wrapper: `tools/runpack_gate.py` (erzwingt Gate, führt Kommando nur bei **ALLOW** aus)
Demo-Receipt: `receipts/gate_demo_output.json`
Index-Feature: `features.hard_gate_enabled = true`

Flow (ab jetzt verpflichtend):
Resume-Flow ✔️→ Overlay-Ack (frisch) ✔️→ Dual-Control-Badge ✔️→ **RUNPACK via **`` → Receipt & Events ins Ledger.

---

16) P3 — Guard-Dashboard (Start)

Ziel: Visualisierung der gehärteten Governance-Kette (Hard-Gate · Resume · Badge · Nightly Proofs).
Artefakt: `overlays/guard_dashboard.html`
Panels: Hard-Gate Status · Resume-Flow · Dual-Control Badge · Nightly Verify · **RCC / Root-Compare** · **Ack-Frische (neu)** · **Events-Stream (neu)**.
Live-Feed: lädt `receipts/gate_live_event.json` **und** `receipts/gate_last.json` sowie `receipts/nightly/manifest.json` und `receipts/events_manifest.json`.

Nächste Schritte (P3): Telemetrie-Feed anbinden, Farb-/Tooltip-Legende, Live-Counter für Ack-Frische (aktiv), Filter/Export (aktiv).

---

18) P3-Ergänzungen — Ack-Ampel & Events-Stream

Ack-Frische-Ampel: Countdown bis Ablauf von `max_ack_age_minutes` aus der Gate-Policy. Farben: **grün** (>60min), **orange** (>10min), **rot** (≤10min/EXPIRED).
Events-Stream: lädt `receipts/events_manifest.json`, Filter **ALL/ALLOW/BLOCK/EXEC_ERROR**, Export als JSON.

Artefakte:

- Manifest: `receipts/events_manifest.json`
- Dashboard: `overlays/guard_dashboard.html` (aktualisiert)

Index: `overlay.features.ack_freshness = true`, `overlay.features.events_stream = true`
Index-SHA256: `8294b5884060b7d11fcd6147153207fe7159ccba5a8e276904851d433523b1ac`

---

19) P3+ — Legende & Root-Age-Counter · Dashboard-Bundle

Legende & Tooltips: neue Kachel mit Badge-Erklärung (**OK/WARN/BAD**) und Hard-Gate-Hinweis.
Root-Compare Live-Counter: Badge `rc-age` zeigt **Zeit seit letztem Nightly-Proof**; Schwellen: **≤12h grün**, **≤24h orange**, **>24h rot**.

Export-Bundle: `exports/dashboard_bundle_0829.zip`
Inhalt: `overlays/guard_dashboard.html`, `receipts/nightly/manifest.json`, `receipts/events_manifest.json`, `policies/gate_policy_v1.json`, `receipts/gate_live_event.json`, `receipts/gate_last.json` + `README.md`.

Index:

- `overlay.features.legend_tooltips = true`
- `exports.dashboard_bundle = exports/dashboard_bundle_0829.zip`
- Governance-Log: `{ type: "dashboard_bundle", status: "CREATED" }`
- **Index-SHA256:** `44f0d040e4fe1d898128eac728778b6b322fb4fb44a05301ce5b796b0228957f`

---

20) Governance-Härtung (Live) — Abschluss der Punkte aus 8.3

Scope: AEAD produktiv schalten · Key-Rotation + Auto-Pruning · Nightly: Batch-Verify inkludieren
Zeit: 29.08.2025 (UTC)

20.1 AEAD Dual Control — produktiv

- **Status:** ✅PRODUCTION
- **Config:** `security/aead.yaml` (mode: dual · factors: TOTP+Signatur · rotation: 30d · pruning: auto · last_rotation: *heute*)
- **Receipt:** `receipts/aead_prod_enable_0829B.json`

20.2 Key-Rotation & Auto-Pruning

- **Durchlauf:** ✅DONE
- **Neue Active-Key-ID:** `k_********`
- **Auto-Pruning: aktiv** (retired >30d → *pruned*, Secret entfernt)
- **Key-Store:** `security/aead_keys.json`
- **Receipt:** `receipts/aead_rotation_0829B.json` *(inkl. Counts/Pruned-IDs)*

20.3 Nightly-Job: Batch-Verify integriert

- **Cron:** `schedules/nightly_verify.cron` → `30 2 * * * python /mnt/data/tools/nightly_verify.py`
- **Letzter Proof:** `receipts/nightly/verify_*.json` (Manifest aktualisiert)
- **Dashboard: RCC / Root-Compare** zeigt MATCH/NO MATCH + **Age-Counter**

Governance-Logs (neu):

- `aead: PRODUCTION` · `aead_rotation: DONE (pruned=N)` · `nightly_job: BATCH_VERIFY_INCLUDED`
  Index: `index.master.json` · **SHA256:** `038a44de1ab51b8a1279852d386b28f3e65479ffb143ce418384da51912637d0`
  Semver: `v1.11B3-P15`

Dashboard (aktualisiert): `overlays/guard_dashboard.html`
Neue Kachel **AEAD Status** (Status & Last-Rotation), Events-Stream enthält AEAD-Events.

---

21) P4 — KPIs · Alerts · Audit Export (Go)

Ziel: Operationalisierung der Beobachtbarkeit & Nachweisführung auf dem Guard-Dashboard.

21.1 KPIs (P4.1)

- **KPIs JSON:** `receipts/kpis_0829C.json`
- Metriken: `root_match`, `gate_allow_rate_last10`, `ack_age_min / ack_max_age_min`, `aead_rotation_age_days`, `nightly_latest_ts`.

21.2 Alerts (P4.2)

- **Alert Evaluator:** `tools/alert_eval.py`
- **Manifest:** `receipts/alerts/manifest.json`
- Regeln: **ROOT_NO_MATCH (CRITICAL)**, **ACK_EXPIRED / ACK_STALE**, **GATE_LOW_ALLOW**, **AEAD_ROTATION_OVERDUE**.
- **Schedule:** `schedules/alerts_hourly.cron` → `0 * * * * python /mnt/data/tools/alert_eval.py`

21.3 Audit Export (P4.3)

- **Exporter:** `tools/audit_export.py`
- **Bundle:** `exports/audit_export_YYYYMMDDTHHMMSSZ.zip` + `.sha256`

Dashboard-Update: `overlays/guard_dashboard.html`
Panels: **RCC/Root-Compare**, **AEAD Status**, **KPIs**, **Alerts**, **Events**, **Legende**.

Index: `index.master.json`

- Flags: `overlay.features.kpis_tile=true`, `overlay.features.alerts_tile=true`, `overlay.features.root_compare_tile=true`, `overlay.features.legend_tooltips=true`
- Export: `exports.latest_audit_export=exports/audit_export_*.zip`
- Semver: `v1.11B3-P16`
- **SHA256:** `7d6339f07906f58a9bea916650ee50f9433bbd55d6ae5a6596ca39866aa421c9`

Hinweis: Freeze-Hash für Root-Compare im Index (`versions.freeze_root`) gesetzt.

---

22) Smoke-Run — Alerts-Pfad unter Live-Beobachtung

Ziel: Validierung der Alarmierungs-Kette via Dashboard (ACK & Root-Compare).

Phase A — Fault Injection

- **ACK_EXPIRED:** Overlay-Ack künstlich >180min alt gesetzt.
- **ROOT_NO_MATCH:** Nightly-Root ≠ Freeze (synthetischer Verify).
- **Erwartung:** CRITICAL-Alerts *ROOT_NO_MATCH* & *ACK_EXPIRED* → **erfüllt**.

Artefakt (Alerts-Manifest, „rot“): `receipts/alerts/manifest.json`
Preview: `[ ROOT_NO_MATCH (CRITICAL), ACK_EXPIRED (CRITICAL) ]`

Phase B — Restore to Green

- **Ack** auf „frisch“ gesetzt.
- **Nightly** mit Freeze-Root neu erzeugt & manifestiert.
- **Erwartung:** Alerts leeren sich → **erfüllt** (Manifest.latest = []).

Artefakte:

- Alerts (grün): `receipts/alerts/manifest.json`
- Nightly-Manifest: `receipts/nightly/manifest.json`
- Overlay-Ack: `overlays/overlay_ack.json`

Governance-Log: `smoke: inject` (Codes: ROOT_NO_MATCH, ACK_EXPIRED) → `smoke: restore` (keine Codes).

---

23) P9-rc1 Delta-Check & Repo-Sync

Frage: „Fehlt etwas?“ — **Antwort:** Es fehlten einige referenzierte Seeds/Policies/Schemas als Artefakte. Sie wurden ergänzt und im Index verankert.

Neu angelegt:

- **Policies:** `policies/alert_policy_v1.json` · `policies/maxwell_boundary_v1.json`
- **Cards/Seeds:** `angel/cards/signal_map.yaml` (signal_map_v1) · `protos/fl_proto.yaml` · `protos/mpc_proto.yaml`
- **Events:** `events/schema/events/ev_final_run.schema.json` · `events/samples/ev_final_run.json`
- **Receipts (Demo):** `receipts/dual_receipts_demo.json` (Consent+Governance) · `receipts/aes_gcm_poc.json` (AES-GCM AAD — strukturelles PoC)
- **Snapshots:** `schedules/snapshot_daily.cron` · `exports/ledger_snapshot_YYYYMMDD.zip`

Index-Update: Seeds/Policies/Events/Receipts/Exports referenziert · **SHA256:** `77879ed57577b0018b747c1399a037128e5ff9c8ae8f78c11bc5ddc8b6620328`

Hinweise:

- AES-GCM-PoC ist **strukturell** (kein Crypto-Lib im Sandbox-Runtime) — Ziel: AAD-Bindung `{receipt_id|root_hash}` demonstrieren.
- UI-Bausteine (GovernanceBar, SensorPicker, M13c) sind als **Konzept** dokumentiert; Implementierungen laufen separat im UI-Repo.