Echo House • Trust Launch Playbook v1.0

Make the claims verifiable, the artifacts public, and the tests repeatable. These steps upgrade Genesis anchoring, the reference verifier, the visual explainer, and the adversarial program—mapped to SPEC v2.1 (BLAKE3 payload, JCS prev■link, 1s/2s segments, 2×2 batching).

1) Publicly Anchor the Genesis Record & Primitives

Publish exact inputs to trust: the Charter + Technical Annex, cryptographic choices, and sample bundles. Anyone can recompute and match.

1. Hash both Charter and Annex with BLAKE3 and SHA■256; publish the digests alongside the files (Git commit, website, and optional notarization).
2. Anchor the BLAKE3 digest to a timestamped place you don't control (e.g., a signed GitHub release or an RFC3161 timestamp).
3. Include a short 'How to verify' section with one■line commands and expected digests.

Commands (copy/paste):

```
# macOS/Linux
b3sum Quadnition_Charter_v1.pdf Annex_Technical_v1.pdf
shasum -a 256 Quadnition_Charter_v1.pdf Annex_Technical_v1.pdf

# Expected (publish these)
# b3 Charter: BLAKE3=
# sha Charter: SHA256=
# b3 Annex: BLAKE3=
# sha Annex: SHA256=
```

2) Publish a Minimal, Auditable Verifier

Keep the SDK closed; open a tiny verifier that replays the exact verification steps. Ship valid and tampered examples.

- Inputs: session_bundle.zip = { media/*, manifest.json, seal_chain.json }.
- Outputs: PASS/FAIL with reasons (e.g., 'JCS prev■link mismatch at segment 7').
- Exact steps: re■hash media (BLAKE3), verify HMAC header, verify JCS prev■link, check segment cadence (1s or 2s immutable per session).
- CLI UX: quad-verify bundle.zip → 'PASS' or 'FAIL: reason'.

```
repo/
README.md
cli.js # Node CLI
lib/hash.js # BLAKE3/SHA-256
lib/jcs.js # RFC 8785 wrapper
lib/verify.js # chain + media checks
examples/valid/*.zip
```

```
examples/tampered/*.zip
```

3) Visualize the Architecture & Tamper Detection

Executives decide with their eyes. Provide two simple diagrams and an animated GIF.

- Proof Map: four concurrent proofs (Thought • Sound • Image • Machine) funnel into one Manifest Hash; Micro■Seal v2 appends link■chained receipts.
- Tamper Path: show how a splice replaces Segment N; the prev■link hash from N■1 no longer matches → chain breaks visibly.
- Figma export kit: dark/light variants, 16:9 slide and square social sizes.

4) Structured Adversarial Program (Bug Bounty)

Invite serious eyes with scoped, testable challenges tied to your success gates.

- Zero■PII Challenge: reward if any PII can be extracted from logs or payloads where you claim none exists.
- Decoy Latency Challenge: reward for any reproducible p95 > 250 ms under the published test harness.
- Watermark Robustness Challenge: reward for a common■attack case where EngraVeil recovery < 95% at conf ≥ 0.8.
- Chain Bypass Challenge: reward if a forged session passes verifier without a live, timed human challenge.

5) Strengthen Human■in■the■Loop (Zero■PII)

Add richer but anonymous signals. Hash locally; never upload raw signals.

- Interaction Rhythm: hash unique■to■session keypress/gesture cadence (no raw biometrics).
- Device Context: hash short windows of accelerometer/gyroscope energy to prove a 'living' hand.
- Focus Proof: hash the schedule of challenge/response times (Δt series); bind to session ID.

6) Transparency Log & Inclusion Proofs

Emit a Merkle root per session day; publish roots with inclusion proofs for each receipt so partners can audit without asking you.

7) 14■Day Pilot: What 'Good' Looks Like

Your buyer should see this in a shared dashboard. If you miss, the pilot is free.

1. Detection: ≥ 95% @ 0.8 on the robustness grid (crop/keystone/scale/print■scan/audio).
2. Latency: camera verify < 1 s perceived; Micro■Seal verdict p95 ≤ 4 s (2×2 batching).

3. Integrity: >99.5% chain continuity; zero silent splice passes.
4. Privacy: 0 PII events; logs redact by design.
5. Reliability: crash■free > 99.8%; batch drop rate < 0.1%.

8) Buyer■Safe Copy (Drop■in)

Starlit GEARS is a single SDK for on■device verification (GlowCheck), invisible watermarking (EngraVeil™), and signed chain■of■capture receipts (Vault + Micro■Seal). It’s private by default—no PII leaves the device. Targets: ≥95% detection at 0.8, <1s verify, and p95 ≤4s for sealed verdicts. Start with a 14■day pilot; if we miss the gates, you pay $0.

Appendix A — Acceptance Harness (Shell)

```
# 1) Launch verifier (local)
API_BASE=http://localhost:8080 node dist/server_v2.js

# 2) Run robustness lab (images/audio)
python tools/watermark_seal_lab.py --in ./samples --out ./lab_out --wm-url
http://127.0.0.1:8080/wm --seal-url http://127.0.0.1:8080/seal --limit 100

# 3) Verify session bundles
quad-verify examples/valid/session_001.zip
quad-verify examples/tampered/session_splice.zip

# 4) Report checklist
# - ROC/PR at 0.8
# - p50/p95 latency camera + micro■seal
# - battery drain %/hr
# - zero PII attestation
```

Appendix B — Diagram Export Spec (for Design)

- Canvas: 1920×1080 and 1080×1080; export SVG + PNG; dark and light.
- Use solid lines for claimed ornament; broken lines for environment (design filings).
- Animation: 6–10 frames GIF for tamper break (splice) and GlowCheck pulse.