C9.9 AMPLIFY LEDGER v.aug08.25

Runtime Audit Compliance + External Exposure Layer

Date: 08 August 2025

Inventor: Dr. Fernando Telles

Affiliations:
CDA AI (Cardiovascular Diagnostic Audit & AI Pty Ltd, ACN 638 019 431)
Telles Investments Pty Ltd (ACN 638 017 384)

Contact:
Dr.Telles@aihumansynergy.org
https://aihumansynergy.org

IP Rights:
US Provisional: #63/826,381
AU Provisional: #2025902482
AU Trade Mark: AI-Human Synergy™ #2535745 & AI-Human Synergy Sentinel Protocol™ #2549093
IP Priority Date: 17 June 2025 (Global Anchor)

Sentinel Protocol Ordinal Bitcoin Wallet:
https://ordinals.com/address/bc1pa3695d7x3cl3k4xut599s6e8yfjl5876uwpq82fqy4tsazxn77sss53mht

This document has been cryptographically hashed and immutably anchored under Sentinel Protocol v3.1 on **08 August 2025**. First public verification anchor: **Ordinal #5**, inscribed **30 July 2025**.
Current hash and payload available via `AMPLIFY_LEDGER_<meta_id>.json`.

C9.9*AMPLIFY*LEDGER_v3.1.md

Title: AMPLIFY*LEDGER*.json – Planning Directive\ **Protocol:** Sentinel Protocol v3.1\ **Version:** v3.1\ **Commander:** Dr. Fernando Telles\ **System:** AI-Human Synergy™\ **Classification:** Runtime Audit Compliance + External Exposure Layer\ **Status:** ✅ Canonical **Date Finalized:** 04 August 2025\ **Linked Modules:** [C5.2, C5.3, C5.7, C8.3, C9.5, C9.6, C9.7, C9.8]\ **Domain Route:** `auditlog.ai`, `aihumansynergy.ai`, `ai-humansyn.ai`, `humansyn.ai`

The AMPLIFY*LEDGER*.json file will serve as the canonical registry of validated sessions under each meta_id thread.

✅ Minimal viable schema:


```
{
    "meta_id": "SENTINFRA-SESS011",
    "audits": [
        {
            "meta_id": "SENTINFRA-SESS011",
            "anchored_file": {
                "session_log": "session_log_SENTINFRA-
SESS011_20250802T235546.017997Z.json",
                "sha256":
"6ffab34bf1eaf9ba9fc86bba6abf9b4d00c630965e680db19e8e3b3f93769614",
                "2ha": "267d75fce31ad442e6f858c8f10f15e511f8dd51",
                "ots": "session_log_SENTINFRA-
SESS011_20250802T235546.017997Z.hash.ots"
            },
            "confirmation_log": {
                "audit_log": "audit_log_SENTINFRA-
SESS011_20250803T000806.024503Z.json",
                "sha256":
"80fa9c165115ebeb0a4e1e018e277af003c3db469b1322763a40374000f4e249",
                "2ha": "a4d17d2446c5bc397a5b4c892ac40e27249b4a51",
                "ots": "audit_log_SENTINFRA-SESS011_20250803T000806.024503Z.hash.ots"
            },
            "op_return": {
                "txid":
"b600bfc29c42cbdef13aea5ac3fc42d5e59913b935e0da5de0f04fe299bb8bd1",
                "block_height": 908326,
                "payload": "SENTINEL|SENTINFRA-SESS011|
267d75fce31ad442e6f858c8f10f15e511f8dd51"
            },
            "validated_by": "VALIS_v2.4",
            "status": "anchored",
            "timestamp": "2025-08-03T22:50:13.158311Z"
        }
    ]
}
```

⍰ How to implement:

- Add AMPLIFY*LEDGER*.json writer block at the end of session_logger.py
- One ledger.json per meta*id, auto-updated per session*logger.finalize_session()
- Output folder: metadata/AMPLIFY*LEDGER/ or ledger*registry/

💡 2. Future Extensions – Breakdown + Recommendations

🔌 a) LightNode CLI (v4.1)

Purpose:

- Enable command-line interfacing with the audit stack for reproducibility + anchor verification.

Example:

```
lightnode verify --file audit_log.json
lightnode anchor-status --meta_id SENTINFRA-SESS011
```

Implementation:

- Wrap `extract_hashes.py`, `hashvalidatorblock.py`, and `opreturnanchor.py` into a CLI package (lightnode-cli)
- Package via setuptools, publish on PyPI
- Add `.json` response options for API integrations

🌐 b) Cross-node meta_id Relay Network

Purpose:

- Distribute meta_id entries across validator mirrors for redundancy + proof-of-presence.

Execution Strategy:

- Each validator (incl. CDA AI) hosts a mirror endpoint /amplify/
- Push model: session*logger.py sends AMPLIFY*LEDGER_.json via secure API
- Pull model: auditlog.ai or mirror checks hash and signature from other nodes

Note: Requires: - Signature enforcement (ECDSA or Bitcoin private key) - Optional: Anchor `.json` of the ledger itself

---

🟢 c) GitHub Sync + Badge

Purpose:

- Make GitHub commits visibly verifiable (like a reproducibility badge)

How:

- Add hash*audit*report.csv + .2ha to /hashes/
- Include a GitHub Action:

```
jobs:
  validate-hash:
    steps:
      - uses: actions/checkout@v3
      - run: python verify_hash.py --input hashes/report.csv
```

Badge Example:

🧭 d) Ledger Explorer at auditlog.ai/verify

Purpose:

Front-end interface to verify .2ha, .ots, and OP_RETURN.

Recommendation:

Domain	Role	Why This Matters in C9.11
aihumansynergy.org	Canonical Batch Distribution	Primary “download portal” for the blinded 100-file challenge ZIP. This is the only location validators trust to retrieve the official batch. Keeps distribution under one controlled endpoint so hashing & anchoring can be pre-verified by the Commander before release.
humansyn.ai	Live Validation Ledger & Result Disclosure	Displays the pre-anchored `.2ha` of the correct file for the current challenge, the reward pool, validator OP_RETURN submissions, and countdown to closure. Acts as the scoreboard + history archive.
auditlog.ai *(optional in C9.11, but powerful)*	Public Verifier & Proof-of-Work Explorer	Lets anyone independently verify that a submitted `.2ha` or TXID is valid for a given challenge — linking back to Bitcoin proof and to the validator’s track record.
ai-humansyn.ai	Routing / Meta Layer	Redirects to the correct `meta_id` challenge page or validator profile, useful for marketing campaigns and QR codes embedded in validator tools.

Domain	Role	Why This Matters in C9.11
(optional) humansyn.org	Neutral Public-Facing Summary	High-level explainer about the blinded proof program for non-validators (journalists, partners), without operational detail.

Final Structure: - `auditlog.ai` → long-term verification & proof explorer (historical, audit-grade) - `humansyn.ai` → real-time challenge validation + result disclosure

---

Sentinel Protocol v3.1 Public Commit Script (Tested and Validated) `amplify_ledger_writer_v2.3.py`

```
 import os
import json
import hashlib
import requests
from datetime import datetime
from pathlib import Path

# === INTRO ===
print("\U0001f512 AMPLIFY_LEDGER Writer v2.3 – Sentinel Protocol v3.1")

# === PROMPTS ===
audit_log_path = input("📄 Enter full path to the audit log `.json` file: ").strip()
session_log_path = input("📄 Enter full path to the corresponding session log `.json`
file: ").strip()
txid = input("🔗 Enter Bitcoin TXID: ").strip()

# === HASH + 2ha + OTS FILENAME: SESSION LOG ===
with open(session_log_path, "rb") as f:
    session_bytes = f.read()
sha256_session = hashlib.sha256(session_bytes).hexdigest()
session_2ha_path = session_log_path.replace(".json", ".2ha")
ots_session_path = session_log_path.replace(".json", ".hash.ots")
if not os.path.exists(session_2ha_path):
    raise FileNotFoundError(f"❌ .2ha not found: {session_2ha_path}")
if not os.path.exists(ots_session_path):
    raise FileNotFoundError(f"❌ .ots file not found: {ots_session_path}")
with open(session_2ha_path, "r") as f:
    ripemd160_session = f.read().strip()

# === HASH + 2ha + OTS FILENAME: AUDIT LOG ===
with open(audit_log_path, "rb") as f:
    audit_bytes = f.read()
sha256_audit = hashlib.sha256(audit_bytes).hexdigest()
audit_2ha_path = audit_log_path.replace(".json", ".2ha")
ots_audit_path = audit_log_path.replace(".json", ".hash.ots")
if not os.path.exists(audit_2ha_path):
    raise FileNotFoundError(f"❌ .2ha not found: {audit_2ha_path}")
if not os.path.exists(ots_audit_path):
    raise FileNotFoundError(f"❌ .ots file not found: {ots_audit_path}")
with open(audit_2ha_path, "r") as f:
    ripemd160_audit = f.read().strip()

# === BLOCK HEIGHT ===
def fetch_block_height(txid):
    try:
        url = f"https://mempool.space/api/tx/{txid}"
        response = requests.get(url)
        response.raise_for_status()
        return response.json().get("status", {}).get("block_height")
    except Exception as e:
        print(f"[!] Could not fetch block height: {e}")
        return None

block_height = fetch_block_height(txid)

# === META_ID ===
meta_id = Path(session_log_path).stem.replace("session_log_", "").split("_")[0]
```

```
timestamp = datetime.utcnow().isoformat() + "Z"

# === LEDGER OBJECT ===
entry = {
    "meta_id": meta_id,
    "anchored_file": {
        "session_log": os.path.basename(session_log_path),
        "sha256": sha256_session,
        "2ha": ripemd160_session,
        "ots": os.path.basename(ots_session_path)
    },
    "confirmation_log": {
        "audit_log": os.path.basename(audit_log_path),
        "sha256": sha256_audit,
        "2ha": ripemd160_audit,
        "ots": os.path.basename(ots_audit_path)
    },
    "op_return": {
        "txid": txid,
        "block_height": block_height,
        "payload": f"SENTINEL|{meta_id}|{ripemd160_session}"
    },
    "validated_by": "VALIS_v2.4",
    "status": "anchored",
    "timestamp": timestamp
}

# === WRITE LEDGER ===
LEDGER_ROOT = Path("ledger_registry")
LEDGER_ROOT.mkdir(exist_ok=True)
ledger_path = LEDGER_ROOT / f"AMPLIFY_LEDGER_{meta_id}.json"

if ledger_path.exists():
    with open(ledger_path, "r") as f:
        existing = json.load(f)
    existing["audits"].append(entry)
else:
    existing = {
        "meta_id": meta_id,
        "audits": [entry]
    }

with open(ledger_path, "w") as f:
    json.dump(existing, f, indent=4)

print(f"[✅] AMPLIFY LEDGER updated: {ledger_path}")
```

🔐 `amplify_ledger_writer.py` Default Classification:

- `amplify_ledger_writer.py` is classified as **Public Validator Tool** — Disclosable as Open Source

🔷 Rationale:

- It does not access user content, only logs + hashes.
- It supports zero-custody ledger registration under C9.5.
- Intended for:
  - ✅ CDA AI internal ledger updates
  - ✅ Independent validator mirrors
  - ✅ Public verification explorers (auditlog.ai/verify)
  - Does not include private key signing, Ordinal inscription logic, or ledger-wide GitHub webhook logic — those are deferred to v4.0 tools (LightNode stack).

✅ Status: Canonical Zero-Trust Runtime Protocol Live

> "No trust. No storage. Just proof."\ — Sentinel Protocol v3.1\ — AI-Human Synergy™
