CHAINNODE · WELCOME PACKAGE

Independent Verification Guide

How to verify a ChainNode compliance report using only a web browser
and the public Polygon blockchain — no ChainNode account required.

Version	Audience	Platform	chainnode.pro
1.0	Auditors & Compliance Officers	ChainNode	Welcome Package

How to Independently Verify a ChainNode Report

A guide for auditors, compliance officers, insurers, and regulators

This document walks a third party — someone with no ChainNode account and no access to the customer's systems — through verifying that a ChainNode compliance report has not been altered since it was generated. You need nothing but a web browser.

---

1. Why you can trust this process

ChainNode writes a cryptographic fingerprint of every asset configuration to the **Polygon public blockchain**. Polygon is an independent, public ledger — ChainNode cannot edit, delete, or backdate entries on it. If the customer ever altered their database after a report was generated, the re-computed fingerprint would no longer match the one on-chain, and the tamper would be immediately visible to you.

This guide shows you how to perform that match in under two minutes per record.

---

2. What you'll need

- The ChainNode compliance report (PDF or CSV) you are auditing.
- A web browser.
- Access to **https://polygonscan.com** — the public blockchain explorer. No login required.

You do **not** need:

- A ChainNode account
- Any credentials from the customer
- Any special software, node, or wallet
- Any cooperation from ChainNode the company

---

3. What's on every report row

Every row in a ChainNode compliance report — asset, change event, or alert — includes these fields relevant to verification:

FIELD	WHAT IT IS
Asset ID	ChainNode's stable identifier for the device
Hostname / IP	The device's network identity at the time of the event
Asset Type	Classification (server, PLC, sensor, etc.)
Config Hash	A 64-character SHA-256 fingerprint of the full configuration
TX Hash	The Polygon transaction that recorded the fingerprint
Block Number	The Polygon block containing the transaction
Logged At	The timestamp of the on-chain write (ISO-8601 UTC)

The **TX Hash** is the key. It is your independent pointer into the public ledger.

---

4. Verification walk-through

Step 1 — Pick a row to verify

Open the report and choose any row. You should repeat this for a representative sample — for a typical audit, verifying 5 to 10 randomly selected rows is sufficient to establish confidence in the full report.

Copy the **TX Hash** from that row. It looks like:

```
0x7f3a8e2d1c4b5a6e9f0d1c2b3a4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e
```

Step 2 — Open the transaction on Polygonscan

Paste the TX hash into the search bar at **https://polygonscan.com** and press Enter. You will see a public page like this:

```
Transaction Details
──────────────────────────────────────────
Transaction Hash:    0x7f3a8e2d...
Status:              Success
Block:               54,123,456
Timestamp:           5 mins ago (2026-04-08 14:32:11 UTC)
From:                0x9a... (ChainNode contract caller)
To:                  0xB3... (ChainNodeProofOfExistence)
Method:              logData
```

If the page loads successfully, this transaction is **real, public, and immutable**. No one — not ChainNode, not the customer, not even Polygon itself — can alter or delete it.

Step 3 — Confirm the block number and timestamp

On the Polygonscan page, check:

- **Block number** matches the one in the report row.
- **Timestamp** matches (or is within a few seconds of) the "Logged At" value in the report.

If either disagrees significantly, flag the row.

Step 4 — Decode the transaction data

Scroll to the **Input Data** section and click **Decode Input Data**. You will see the function call:

```
Function: logData(
    string sensorId,
    uint256 value,
    string unit,
    string metadata
)

sensorId:  "chainnode_asset"
value:     0
unit:      "sha256_hash:a3f7e1b2c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7
metadata:  {
    "product": "chainnode",
    "organization_id": "5c3f...",
    "chainnode_asset_id": "cn_line1-plc-01",
    "hostname": "line1-plc-01",
    "ip_address": "10.1.4.22",
    "asset_type": "iot_controller",
    "config_hash": "a3f7e1b2c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9
    "change_type": "discovered",
    "timestamp": "2026-04-08T14:32:09Z"
}
```

Step 5 — Cross-check the four critical fields

Confirm every one of the following matches between the report row and the on-chain metadata:

CHECK	REPORT SAYS…	ON-CHAIN SAYS…	MATCH?
Config hash	`a3f7e1b2…`	`a3f7e1b2…` (in both `unit` and `metadata.config_hash`)	✓ / ✗
Asset ID	`cn_line1-plc-01`	`metadata.chainnode_asset_id`	✓ / ✗
Hostname	`line1-plc-01`	`metadata.hostname`	✓ / ✗
Organization ID	(matches customer's tenant UUID)	`metadata.organization_id`	✓ / ✗

If **all four fields match**, the row is verified. The customer cannot have altered that asset's recorded configuration — the evidence is frozen on a public ledger and the hash in the report is demonstrably the same hash that was written on-chain.

If **any field does not match**, the row has either been edited after the fact or the report is referencing the wrong transaction. Flag it and request explanation.

---

5. What a successful verification proves

When every sampled row passes the check in §4, you have cryptographically established that:

1. **The data existed** at the time stated. Polygon timestamps are consensus-verified by a global network of independent validators; they cannot be backdated.
2. **The data has not been altered** since. If the customer edited any configuration field after the write, the re-computed SHA-256 would differ from the on-chain hash.
3. **The data belongs to the customer's tenant.** The `organization_id` in the on-chain metadata ties the record to a specific tenant UUID — it cannot be "borrowed" from another customer's records.
4. **ChainNode itself cannot rewrite history.** The transactions are written from a public wallet to a public contract on a public chain. Neither ChainNode nor the customer controls Polygon.

This is stronger than a notary stamp, stronger than a timestamped PDF, and stronger than a vendor's internal audit log, all of which rely on trusting the entity that produced them.

---

6. What this verification does *not* prove

To be precise about what a ChainNode report is and isn't:

- It **does not** prove the customer's physical devices exist. It proves that *at the time of the scan*, the agent reported a device with the fingerprinted configuration.
- It **does not** prove the agent was running on the correct network. That is established by the customer's deployment records.
- It **does not** prove a human has reviewed the data. Reviews and sign-offs are a process layer on top of the ChainNode evidence.
- It **does not** prove anything about time periods *between* writes. If scans run hourly, the blockchain tells you about hourly snapshots, not about what happened in the 59 minutes between them.

A ChainNode report establishes a **tamper-evident, time-anchored record of what the customer's network looked like to the agent at the moment each scan ran**. That is what the blockchain layer gives you, and it is enough for every common compliance regime.

---

7. Understanding what's on-chain vs. off-chain

ChainNode uses a **hash-anchor** architecture. The blockchain stores a cryptographic fingerprint plus a small metadata envelope; the rich, human-readable data stays in the customer's ChainNode database, linked to the on-chain proof by the transaction hash.

On-chain (per transaction):

- `product` identifier (`"chainnode"`)
- `organization_id` (tenant UUID)
- `chainnode_asset_id`
- `hostname`
- `ip_address`
- `asset_type`
- `config_hash` (SHA-256 fingerprint of the full configuration)
- `change_type` (discovered / modified / removed / reinstated)
- `timestamp` (ISO-8601)

Plus the Polygon-supplied block number and transaction hash.

Off-chain in the ChainNode database (joined via `blockchain_tx_hash`):

- Full identity: hostname, all IPs, MAC, OS name and version, make, model, firmware version
- Governance fields: department, assigned owner, location, risk level, status
- Lifecycle: first seen, last seen, full change timeline
- AI classification, confidence, risk notes
- The complete raw discovery payload captured by the agent

This split is intentional. Writing every field to the blockchain would be expensive and unnecessary; the hash alone is enough to detect any tampering with the off-chain copy. You get the same audit property at a fraction of the cost and complexity.

---

8. Sampling recommendations

For a typical compliance review, the following sampling plan is defensible:

REPORT SIZE	ROWS TO VERIFY
≤ 50 rows	All rows
51 – 500 rows	10% random sample, minimum 10 rows
501 – 5,000 rows	5% random sample, minimum 50 rows
> 5,000 rows	1% random sample, minimum 100 rows, plus all rows marked as anomalies

Always include:

- Every row flagged as an **anomaly** in the report
- Every row representing a **Critical** severity alert
- A random sample across the remaining rows

Any single failed verification should trigger a full review of that asset's change history.

---

9. Reporting a discrepancy

If any row fails the verification described in §4:

1. **Do not disclose** the finding to the customer until you have confirmed it on a second independent machine (browser cache or network interception is an unlikely but possible explanation for a single mismatch).
2. Capture a screenshot of both the report row and the Polygonscan transaction detail.
3. Request from the customer: the full change history for the affected asset, and the agent logs for the time window in question.
4. Escalate per your organization's audit process.

A cryptographic mismatch is not an accounting disagreement — it is a binary result.

---

10. Summary

You need **only a web browser**, **only the ChainNode report**, and **only Polygonscan** to establish that a ChainNode compliance report is exactly what the customer claims it is. The trust model does not depend on ChainNode the company, on the customer's goodwill, or on any private infrastructure. Every claim in the report is anchored to a public ledger that the customer does not control.

That is the guarantee ChainNode was built to provide. This document is how you collect it.

ChainNode is a trademark of its respective owner. Polygon is a trademark of Polygon Labs. This guide corresponds to ChainNode platform version 1.0 and applies to all reports generated on or after the platform's first production deployment.
