



FULL PROOF

Robinhood Chain block production

The Full Proof for Proof Posit PP-005, version 2 - free, identical for every reader

DOCUMENT CONTROL NUMBER	JFD-20260928-FP-00001
CLASS OF RECORD	resolved
DATE	2026-09-27
COVERAGE	2026-09-04T00:00:00Z → 2026-09-04T23:59:59Z
SOURCE FINGERPRINT (SHA-256)	b8514b6422e2466e5ed61c81faa27ba8e037102b06af25345bf60334412eec3b
VERIFICATION	https://gh.jetfyul.com/v/JFD-20260928-FP-00001

K. L. Phillips
Chief Executive Officer, Jet Fyul Dynamics LLC

The pages that follow are the full text of this document's exact source, set as readable pages. The SHA-256 printed in every footer is the fingerprint of that source. The seal of this document — its ledger event, both hardware witnesses and its Bitcoin anchor — is recorded at the verification address above, not in these pages.

Jet Fyul Dynamics LLC · contact@jetfyul.com

THE FULL PROOF — PP-005 (FREE)

Robinhood Chain block production · version 2

PART 1 — COVER

Field	Value
Full Proof	FP-005, version 2
Accomplices	Proof Posit PP-005, “Robinhood Chain Produced Blocks Without Interruption on 4 September 2026”, sealed as JFD-20260926-PP-00001 (Glass Hull ledger kind-15 3171889; document SHA-256 530a8f9569113791289c92ff5a9380d4f0b60919d52cfa52da992651c8e57da9)
Version 1	Sealed as JFD-20260926-FP-00001 (kind-15 3173062; document SHA-256 a151ac8fe3d32927e4b265814091ba5b623258cb3afd30ecd722235a32033c78). Its full text (SHA-256 838c1e337546a636668b3be1a1ba172b1f9ab77cf01bcdcf0b8cab1b01276944) is delivered inside certificate JFD-20260926-CERT-00004 (kind-15 3173070; SHA-256 7c43d21919abcf4651aca0ffe54fd183a0b43970cd9304c7964d8c18b61c098b). This version sets out the same records in ten parts and adds no measurement.
Chain	Robinhood Chain, chain ID 4663
Pinned at	Block 62,471,946, hash 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (2026-09-14T03:00:01Z)
Frozen method	SHA-256 fea53b1c83bc9b7f4819eefa459d7178b1cbcd7b4cf7cadad0fb924086b91029 (kind-3 3147511); amendment A1 SHA-256 3b8e94babf28701db458622d614ccc399cc7dd2aad3d10d1c890952f0339738 (kind-3 3159143)
Price and delivery	Free. Public, no gate. Identical for every reader, so no per-copy fingerprint.

The ten parts. 1 Cover · 2 The finding · 3 The question and the frozen method · 4 Population and pin · 5 Complete identifiers · 6 The queries, verbatim · 7 The two measurements, side by side · 8 The refute arm · 9 The measured tables · 10 Re-run hashes and reproduction · Appendices R and S.

PART 2 — THE FINDING

2.1 What this Full Proof proves

From version 1, section 1 (source SHA-256 838c1e337546a636668b3be1a1ba172b1f9ab77cf01bcdcf0b8cab1b01276944).

It carries everything needed to reproduce, without us, the measurement that Robinhood Chain produced blocks continuously on 4 September 2026 (1,440 of 1,440 minutes; longest interval 2 s), with a refute arm showing the same measurement finds long intervals where they exist.

2.2 Summary

Quoted verbatim from the sealed Proof Posit PP-005, section I. Source SHA-256

3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff, the source recorded by seal event kind-15 3171889 for JFD-20260926-PP-00001.

On 4 September 2026, published reports described Robinhood Chain as having stopped for about 14 minutes beginning about 12:57 UTC (for example crypto.news, archived web.archive.org/web/20260924200651/...).

We measured every block the chain produced that day, from its own permanent record.

- **Block production never stopped.** Robinhood Chain produced 854,255 blocks carrying 13,978,496 transactions across all 1,440 minutes of the UTC day.
- **No minute fell short.** The lowest count in any single minute was 548 blocks.
- **The longest interval between consecutive blocks was 2 seconds.**
- **At the reported time, the chain ran normally.** In the minute beginning 12:57 UTC it produced 592 blocks, and the Ethereum reference the chain records advanced every minute of the day.

Since public mainnet opened on 1 July 2026, the longest interval between consecutive blocks is 16 seconds. It fell on the launch morning, 1 July at 05:35 UTC.

What paused was the chain's posting of data to Ethereum: 14 minutes in total, before 12:57 UTC. What we cannot see from the chain is whether any front end, public endpoint, or explorer was unavailable to users; Arbitrum reported a brief impact on infrastructure providers that rely on the chain's data stream. Section IV states that boundary.

2.3 Findings

Quoted verbatim from the sealed Proof Posit PP-005, section III (the tables, before III.i). Source SHA-256

3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff, the source recorded by seal event kind-15 3171889 for JFD-20260926-PP-00001.

Measure	4 September 2026 (UTC)
Minutes observed	1,440 of 1,440
Blocks produced	854,255
Transactions	13,978,496
Fewest blocks in any minute	548
Minutes with fewer than 500 blocks	0
Longest interval between consecutive blocks	2 seconds
Minutes in which the recorded Ethereum reference did not advance	0

The reported window, 12:30–13:30 UTC, minute by minute:

- Every minute produced between 563 and 657 blocks.
- The longest interval was 2 seconds.
- Transactions ran between 4,245 and 17,423 per minute.
- **The 12:57 UTC minute:** 592 blocks (54,282,091 to 54,282,682) and 5,868 transactions.

Context, public mainnet to the pinned block:

Measure	Value
Longest interval between consecutive blocks since 2026-07-01 00:00 UTC	16 seconds, block 678,037, 2026-07-01 05:35:35 UTC
The next four longest	13 s, 13 s, 12 s, 11 s, all within the launch morning of 1 July

2.4 Limits of observation

Quoted verbatim from the sealed Proof Posit PP-005, section IV. Source SHA-256

3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff, the source recorded by seal event kind-15 3171889 for JFD-20260926-PP-00001.

- 1. Service availability is not observable from the chain.** Our record shows blocks produced and transactions included. It does not show whether a wallet, front end, public endpoint, indexer or block explorer was reachable. If users could not submit or see transactions during the reported window, the cause lay outside block production, and the chain's own record cannot identify it.
- 2. Posting to Ethereum is a separate process.** Robinhood Chain periodically posts batches of its data to Ethereum. On 4 September 2026 that posting paused twice for long periods:
 - from 12:29:47 to 12:38:23 UTC (516 seconds)
 - from 12:42:47 to 12:48:11 UTC (324 seconds)

That is 840 seconds, 14 minutes in total, while the chain itself kept producing blocks (Glass Hull record 3137609). Two shorter gaps followed, at 13:34:47 UTC (144 seconds) and 13:45:59 UTC (132 seconds). All of these fall before or after the 12:57 UTC time given in the reports, not at it.

Arbitrum stated that day: "Robinhood Chain experienced no downtime. Earlier today, Robinhood Chain experienced batch posting delays due to L1 market blob behavior. Direct user transactions experienced no delays." (x.com/arbitrum/status/2095939844987891760, 2:19 PM ET, 4 September 2026; Glass Hull signed-in capture, sha256 213385cb..., OpenTimestamps-stamped.)

Ethereum's public record for those minutes shows the following (Glass Hull record 3137956):

- In both long pauses, the batch poster's last standing bid for blob space **exceeded** the blob base fee charged in every block of the pause: by about 4–13 times in the first, and about 4.5–8.8 times in the second.
- The first posting after each pause carried a higher bid.

The blob base fee alone therefore does not account for either pause. Whatever else delayed inclusion is not observable from included transactions, because replaced or dropped submissions leave no trace in chain data. We state no cause.

3. **Timestamps have one-second resolution**, so an interval under one second cannot be measured more finely than that.
4. **We did not contact Robinhood** and have no access to its internal systems or incident records.

2.5 Conclusion

Quoted verbatim from the sealed Proof Posit PP-005, section V. Source SHA-256 3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff, the source recorded by seal event kind-15 3171889 for JFD-20260926-PP-00001.

Robinhood Chain produced blocks without interruption throughout 4 September 2026, including at and around 12:57 UTC. Within the limits stated in Section IV, the chain's own record does not support a description of that day as a halt in block production. Since public mainnet opened, the chain has not gone more than 16 seconds without producing a block.

This finding measures block production. It draws no conclusion about any party's statements, systems or intentions.

PART 3 — THE QUESTION AND THE FROZEN METHOD

3.1 The question

Quoted verbatim from the sealed Proof Posit PP-005, section II. Source SHA-256 3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff, the source recorded by seal event kind-15 3171889 for JFD-20260926-PP-00001.

What we measured. Whether the chain produced blocks continuously on 4 September 2026 (UTC):

- blocks produced per minute
- the longest interval between consecutive blocks
- whether the chain's recorded Ethereum reference advanced each minute

We measured the same quantities from public mainnet (1 July 2026) to the pinned block, to place the day in context.

The reporting rule. A halt is any interval between consecutive blocks long enough to be seen as a stop. We report every interval exactly, and state the longest.

3.2 The frozen method, verbatim

Fingerprint (SHA-256)	fea53b1c83bc9b7f4819eefa459d7178b1cbcd7b4cf7cadad0fb924086b91029
Frozen on the ledger	kind-3 3147511, created 2026-09-25 14:27:03.560347+00
Event hash	855054189e60211fb06f59286c177d20c67273b0edf8ad20b73453d6748172de

```
# FROZEN METHOD — PP-005 (Robinhood Chain block production, 4 September 2026)

**Status.** Frozen after exploratory measurement (exploratory runs 2026-09-24 15:38-15:40Z, re-run hashes listed at the end). The sealed figures are the RE-RUN of these statements under this frozen method.

**Population.** Every block in `glasshull.blocks`, genesis to the pin: block 62,471,946, hash 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (the store's E0 tip, 2026-09-14T03:00:01Z). Coverage must read one continuous interval 0 .. 62,471,946.

**Surface.** Glass Hull evidence console, store surface (ClickHouse on A1, read-only user; 120 s / 10,000-row bounds); every statement is run at the pinned height and recorded as a run with a re-run hash.

**Definitions.** Interval between consecutive blocks = timestamp(n) - timestamp(n-1), ordered by block number, in whole seconds. A UTC minute = floor(timestamp / 60) · 60. The recorded Ethereum reference = `l1_block_number`; a minute in which max - min of it is 0 is a minute in which it did not advance. Each window statement reads 60 s before its window so the first minute's first interval is measured against the previous block.

### Pass arm
#### P1 — 2026-09-04 full UTC day, per-minute aggregate

```sql
SELECT count() AS minutes, min(l2_blocks) AS min_blocks_per_min, max(max_gap_s) AS max_gap_s_day, countIf(l2_blocks < 500) AS minutes_under_500_blocks, countIf(l1_advance = 0) AS minutes_l1_ref_flat, sum(l2_blocks) AS blocks_day, sum(txs) AS txs_day FROM (SELECT toStartOfMinute(toDateTime(timestamp, 'UTC')) AS m, count() AS l2_blocks, max(gap_s) AS max_gap_s, max(l1_block_number) - min(l1_block_number) AS l1_advance, sum(tx_count) AS txs FROM (SELECT number, timestamp, l1_block_number, tx_count, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks WHERE timestamp >= 1788479940 AND timestamp < 1788566400) WHERE timestamp >= 1788480000 GROUP BY m)
```

#### P2 — the reported window 12:30-13:30 UTC, per minute

```sql
SELECT toStartOfMinute(toDateTime(timestamp, 'UTC')) AS minute_utc, count() AS l2_blocks, min(number) AS first_block, max(number) AS last_block, max(gap_s) AS max_gap_s, min(l1_block_number) AS l1_min, max(l1_block_number) AS l1_max, sum(tx_count) AS txs FROM (SELECT number, timestamp, l1_block_number, tx_count, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks WHERE timestamp >= 1788524940 AND timestamp < 1788528600) WHERE timestamp >= 1788525000 GROUP BY minute_utc ORDER BY minute_utc
```

### Refute arm
The same interval measurement must return long intervals where they exist.

#### R1 — full history, the five longest intervals

```sql
SELECT number AS block_after_gap, toDateTime(timestamp, 'UTC') AS resumed_utc, gap_s FROM (SELECT number, timestamp, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks) WHERE number > 0 ORDER BY gap_s DESC, number ASC LIMIT 5
```

**Reporting rule for R1.** The row for block 1 is the interval from the genesis block, whose recorded timestamp is 0; it is reported and labelled as the genesis artifact, never presented as a stop.

```

```
### R2 — since public mainnet (2026-07-01 00:00Z), the five longest intervals

```sql
SELECT number AS block_after_gap, toDateTime(timestamp, 'UTC') AS resumed_utc, gap_s FROM (SELECT nu
number, timestamp, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.bl
ocks WHERE timestamp >= 1782863000) WHERE timestamp >= 1782864000 ORDER BY gap_s DESC, number ASC LI
MIT 5
```

### Reporting
Every figure published is read from the re-run's result bytes (row and column named); nothing is ret
yped. Exploratory re-run hashes (for the record, not for binding): PP005_day = 17fedf045ed4060954b8c
6a1d95ab5721c194a3a387b9d5cbfb04c93529a31f6, PP005_window = 87169581e67c858c95d342f9bff6fbd7b1b8f04d
7e1b7ef7cfe5db88323f1ea8, PP005_refute_full = 7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be
59143709b3, PP005_refute_mainnet = b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f.
```

3.3 Amendment A1, verbatim

| | |
|-----------------------|--|
| Fingerprint (SHA-256) | 3b8e94babf28701db458622d614ccc399cc7dd2aadcd3d10d1c890952f0339738 |
| Frozen on the ledger | kind-3 3159143, created 2026-09-26 03:11:53.270764+00 |
| Event hash | 600a62ea1ecbc88d915e5d5b5b8a84e77a005adccb2fd25eb3cdebefe867b3d22 |
| What it changes | no statement, population, definition or figure of the pass/refute arms; declares the object {question, method_version, pin, day, window_minutes[60], refute_full[5], refute_mainnet[5]}; defines checker route B |

```
# PP-005 METHOD — AMENDMENT A1: the canonical output object and the second measurement (A2 bronze)

**Amends.** The frozen method `PP-005-METHOD-frozen.md` (kind-3 3147511). It changes no statement, p
opulation, definition or figure of the pass or refute arms. It does two things: it declares the obje
ct each arm emits, so the arms compare by digest (OUTPUT-CONVENTIONS 1.1), and it defines the second
measurement approved by tpa-architect on 2026-09-25 (GH-SEAL-ALL-001 rulings, item 2). It is frozen
before the second measurement is run.

### A1.1 CANONICAL OUTPUT (both maker and checker emit EXACTLY this; the hash is over it)

...
{
  "question": "PP-005",
  "method_version": "v1+A1",
  "pin": { "block": <int>, "hash": "<str>" },
  "day": { "minutes": <int>, "min_blocks_per_min": <int>, "max_gap_s_day": <int>, "minutes_under_500_blocks":
<int>, "minutes_l1_ref_flat": <int>, "blocks_day": <int>, "txs_day": <int> },
  "window_minutes": [ { "minute_utc": "<str>", "l2_blocks": <int>, "first_block": <int>, "last_block": <int
>, "max_gap_s": <int>, "l1_min": <int>, "l1_max": <int>, "txs": <int> } ],
  "refute_full": [ { "block_after_gap": <int>, "resumed_utc": "<str>", "gap_s": <int> } ],
  "refute_mainnet": [ { "block_after_gap": <int>, "resumed_utc": "<str>", "gap_s": <int> } ]
}
...

- All keys are literal; no other key exists.
- `window_minutes` holds 60 rows in minute order (12:30 .. 13:29 UTC). `refute_full` and `refute_mai
nnet` hold 5 rows each, in the method's order.
- Times are UTC, formatted `YYYY-MM-DD HH:MM:SS`.
- Canonical JSON = `json.dumps(obj, sort_keys=True, separators=(",", ":"), ensure_ascii=False)` (UTF-
```

```
8); sha256 over those bytes.
```

```
### A1.2 Maker: route A, the frozen statements re-run under 3147511
```

- `day` is the single row of P1.
- `window\_minutes` are the rows of P2.
- `refute\_full` and `refute\_mainnet` are the rows of R1 and R2.
- `pin` is the height and block hash recorded on the re-run records (all must agree).
- Every value is read from the stored result bytes, after their sha256 is re-verified against the re-run record (3147552).

```
### A1.3 Checker: route B, the second measurement (A2 bronze, DuckDB)
```

```
**Engine and source.** DuckDB in gh-nitro over the bronze `blocks.parquet` partitions under `/mnt/a2/gh-bronze`: `snapshot-20260910T180834Z`, `snapshot-20260910T180834Z-gapfill` and `snapshot-20260914-0300-headext`, restricted to `number <= 62471946`.
```

```
**Integrity checks, before anything is computed.**
```

- Block numbers are de-duplicated. A block present in more than one partition must carry the same `timestamp`, `l1\_block\_number` and `tx\_count` in every copy. The count of such blocks, and the count whose copies disagree, is recorded. Any disagreement stops the checker.
- Coverage must read one continuous interval: `min = 0`, `max = 62471946`, `distinct = max + 1`.
- `pin` = block 62,471,946 and its hash as read from the bronze, and it must equal the maker's pin.

```
**The same four measurements, in DuckDB SQL.**
```

- The interval is `timestamp(n) - timestamp(n-1)` ordered by `number`. The first row of each read has no predecessor and is measured against 0, as the ClickHouse `lagInFrame` default does. This is the frozen definition, unchanged.
- P1 and P2 use the same 60-second lead-in and the same bounds as the frozen statements.
- R1 and R2 use the same bounds and the same order: `gap\_s DESC, number ASC`, 5 rows.

```
**Label.** Engine-independent, lineage-shared.
```

- The A1 store (route A) was loaded from this bronze. The comparison therefore shows that a different engine, over the raw extraction files rather than the loaded tables, reproduces every figure.
- It does not show that a second, independent extraction of the chain agrees.

```
### A1.4 Comparison
```

The comparison is the canonical digests of the two objects, recorded in full, with the differing keys listed. The verdict is `CLEARED` when the digests are equal and `DIVERGED` otherwise; a divergence is reported, never reconciled.

3.4 Order of events

| Event | Time (UTC) | Record |
|---------------------------------------|-------------------------------|------------------------------|
| Exploratory measurement | 2026-09-24 15:38–15:40Z | frozen method, Status line |
| Method frozen | 2026-09-25 14:27:03.560347+00 | ledger 3147511 |
| First re-run under the frozen method | 2026-09-25T14:27:56Z | reruns.json (ledger 3147552) |
| Last re-run | 2026-09-25T14:27:59Z | reruns.json (ledger 3147552) |
| Re-run record banked | 2026-09-25 14:29:59.168369+00 | ledger 3147552 |
| Amendment A1 frozen | 2026-09-26 03:11:53.270764+00 | ledger 3159143 |
| Second measurement (route B) finished | 2026-09-26T03:15:17Z | comparison file |
| Comparison banked | 2026-09-26 03:15:38.799703+00 | ledger 3159198 |

PART 4 — POPULATION AND PIN

4.1 Population

From the frozen method (Part 3.2).

Population. Every block in `glasshull.blocks`, genesis to the pin: block 62,471,946, hash `0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb` (the store's E0 tip, 2026-09-14T03:00:01Z). Coverage must read one continuous interval 0 .. 62,471,946.

4.2 Coverage at run time (route A, the store)

| Statement | Tables the statement reads | Coverage recorded on its run record (table min..max) | Coverage read at (UTC) |
|-----------|----------------------------|--|------------------------|
| P1 | blocks | blocks 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| P2 | blocks | blocks 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| R1 | blocks | blocks 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| R2 | blocks | blocks 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |

Every run record carries the same coverage fingerprint, `df27b42a736fc8de2d9fb6b69067303d162b72efe116c8bd95c1514d4fa7396d`. It is the SHA-256 of the canonical JSON `{"engine":"clickhouse","intervals":[[0,62471946]],"partitions":627}` (Appendix S §S.4), and it recomputes from that value.

4.3 Coverage of the second measurement (route B, the bronze)

| Check | Value |
|------------------------------|--|
| Snapshots read | snapshot-20260910T180834Z, snapshot-20260910T180834Z-gapfill, snapshot-20260914-0300-headext |
| Partitions | 627 |
| Rows read | 62471947 |
| Coverage min / max | 0 / 62471946 |
| Distinct blocks | 62471947 |
| Blocks seen more than once | 0 |
| Blocks whose copies disagree | 0 |
| Engine | DuckDB 1.5.5 |

4.4 The pin

| | |
|------------------|---|
| Block | 62,471,946 |
| Hash | <code>0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb</code> |
| Block time (UTC) | 2026-09-14T03:00:01Z |

| | |
|---------------------------------|--|
| Height and head at every re-run | 62471946 / 62471946 (all 4 run records agree) |
| Block hash on every run record | 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (all 4 agree) |

Read at the pin, never at latest.

PART 5 — COMPLETE IDENTIFIERS

5.1 Chain, tables and blocks

| Identifier | Value |
|---|--|
| Chain ID | 4663 |
| Pinned block and hash | 62471946 ·
0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Store tables read | glasshull.blocks |
| Blocks in the reported window 12:30–13:29 UTC | 54266140 to 54301744 |
| Blocks after the five longest intervals, full history (R1) | 1, 36, 234, 34, 147 |
| Blocks after the five longest intervals since public mainnet (R2) | 678037, 677988, 678020, 678022, 678195 |
| Contracts and addresses | None; this finding concerns block production only. |

5.2 Documents

| Document | DCN | Ledger event | SHA-256 |
|---|-------------------------|---------------------------------|--|
| Proof Posit PP-005 | JFD-20260926-PP-00001 | kind-15 3171889 | 530a8f9569113791289c92ff5a9380d4f0b60919d52cfa52da992651c8e57da9 |
| Proof Posit PP-005, source | — | kind-15 3171889 (source_sha256) | 3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff |
| Full Proof FP-005, version 1 | JFD-20260926-FP-00001 | kind-15 3173062 | a151ac8fe3d32927e4b265814091ba5b623258cb3afd30ecd722235a32033c78 |
| Full Proof FP-005, version 1, full text | — | printed on the v1 cover | 838c1e337546a636668b3be1a1ba172b1f9ab77cf01bcdcf0b8cabcb1b01276944 |
| Certificate of version 1 | JFD-20260926-CERT-00004 | kind-15 3173070 | 7c43d21919abcf4651aca0ffe54fd183a0b43970cd9304c7964d8c18b61c098b |

5.3 Ledger events cited

| Event | Kin d | What it records | Created (UTC) | Event hash |
|----------|-------|---|-------------------------------|--|
| 3095 118 | 15 | corroboration (1–6 September intervals) | 2026-09-23 03:55:11.330129+00 | 33245b970ab3d58aa81185752480afb045b437bbe06b3cd0edf4deca5868414d |

| Event | Kind | What it records | Created (UTC) | Event hash |
|---------|------|---|-------------------------------|---|
| 3137609 | 3 | posting pauses | 2026-09-25 02:58:00.84014+00 | 20a58c5496f888cafebe2b1caf831c94292185c661fd229db8e0f4e8d4486e62 |
| 3137956 | 3 | blob market | 2026-09-25 03:23:07.126996+00 | df0770669f963afdc3c0eece6a0679883a729a11b3cae40e209f89feb590074c |
| 3147511 | 3 | method freeze | 2026-09-25 14:27:03.560347+00 | 855054189e60211fb06f59286c177d20c67273b0edf8ad20b73453d6748172de |
| 3147552 | 3 | re-run record | 2026-09-25 14:29:59.168369+00 | 69ebab908ab50df1d5ab84815664934dee0e4f6b61a3db56cf5978702f0bf79b |
| 3147767 | 3 | correction to 3147552 (what the re-run hash covers) | 2026-09-25 14:43:43.235442+00 | da8a59ecc0b18c55948b11e9849b33ed1eb96ec270de4dbf14ed408d5c7abb1c |
| 3159143 | 3 | amendment A1 freeze | 2026-09-26 03:11:53.270764+00 | 600a62ea1ecbc88d915e5d5b5b8a84e77a005adccb2fd25eb3cdebef867b3d22 |
| 3159198 | 3 | maker/checker comparison | 2026-09-26 03:15:38.799703+00 | def85cfe7cbb8a8986efa3d4f97df1d1579192598e4f9ae11d12518bd3ec537dc |
| 3171889 | 15 | Posit seal | 2026-09-26 17:07:35.741998+00 | 10999d8fec99806daba5ede7336d5935932d6526a98ca9f4f986a395e177cc0a |
| 3173062 | 15 | Full Proof v1 seal | 2026-09-26 18:20:41.736077+00 | 4e7dc5ea08083f9136b49f672eb9d232b8cfa03344a4480b344f760a1a2f120e |
| 3173070 | 15 | v1 certificate seal | 2026-09-26 18:21:01.315904+00 | 7da6b0d1d48a4970e8c1a11e4d5ec36b63c9afd2684da5d47664aabf05172e75 |

5.4 Record files

| Record | File | SHA-256 | Banked in |
|---|---|--|-----------|
| sealed Proof Posit source (seal event source_sha256) | PP-005-v2-ASSEMBLED.md | 3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff | — |
| sealed Full Proof v1 full text (the source fingerprint printed on the v1 cover; attachment of the v1 certificate) | FP-005-free-full-proof.md | 838c1e337546a636668b3be1a1ba172b1f9ab77cf01bcd0b8cabcb1b01276944 | — |
| frozen method | PP-005-METHOD-frozen.md | fea53b1c83bc9b7f4819eefa459d7178b1cbcd7b4cf7cadad0fb924086b91029 | 3147511 |
| method amendment A1 | PP-005-METHOD-amendment-A1-output-object.md | 3b8e94babf28701db458622d614cc399cc7dd2aad3d10d1c890952f0339738 | 3159143 |
| re-run record (statements, rows, result SHA-256) | reruns.json | baf200c4304443b0cd26244b97f8fb880d3f1bb4c90ebb433c56f05ed82847fe | 3147552 |
| maker arm object (route A) | PP005-maker-v1.json | 65fed4206040ccdbfd6f526748e1e708a473eef31d51551957d9f745bd213bc1 | 3159198 |

| Record | File | SHA-256 | Bank ed in |
|--|---------------------------|---|------------|
| checker arm object (route B) | PP005-checker-v1.json | 65fed4206040ccdbfd6f526748e1e708a473eef31d51551957d9f745bd213bc1 | 3159198 |
| maker/checker comparison | PP005-comparison-v1.json | d7a0e5777ee193c1549a7fa329b55d8e26ebd021d180f47abc7e389482549393 | 3159198 |
| Appendix R as attached to the v1 certificate | FP-005-appendix-R.md | 1a2a35e6f745f2d1b15f2c79ec79f69add81fcfb5a7e958b375578693074875c | — |
| Appendix S as attached to the v1 certificate | FP-005-appendix-S.md | 53f3b12900d819268c678206a6e6a244298086c43a3b3d0d2985d567ba dd18f6 | — |
| checker run record (route B) | PP005-checker-v1.run.json | 3ad9bf0904f66ad7d2d3daddef395013655173285572fab061e31a97f682d610 | 3159198 |

The console run record and result file of each statement are listed with it in Part 6.

Bronze snapshots read by route B: snapshot-20260910T180834Z, snapshot-20260910T180834Z-gapfill, snapshot-20260914-0300-headext.

PART 6 — THE QUERIES, VERBATIM

Every statement below was re-run on the store surface after the freeze, exactly as printed. The statement text is the normalised form of Appendix S §§.3, copied from the statement's run record. Reproducibility class for every statement: needs a full node or a complete extraction.

P1 — PP005_day (pass arm)

| | |
|-------------------|--|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:27:56Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 17fedf045ed4060954b8c6a1d95ab5721c194a3a387b9d5cbfb04c93529a31f6 |
| Columns (types) | minutes (UInt64), min_blocks_per_min (UInt64), max_gap_s_day (Int64), minutes_under_500_blocks (UInt64), minutes_l1_ref_flat (UInt64), blocks_day (UInt64), txs_day (UInt64) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 282b46ef92004ba52b931a510b46d990f04ca5f6999633b0418846010a6586e1 |

| | |
|-----------------|---|
| Run record file | runs\17fedf045ed4060954b8c6a1d95ab5721c194a3a387b9d5cbfb04c93529a31f6.json, SHA-256 d7c02a906dfdbbb13120f1f704d07760151ce917ca5cb75e46d17c4aee133caf |
| Result file | results\17fedf045ed4060954b8c6a1d95ab5721c194a3a387b9d5cbfb04c93529a31f6.json, SHA-256 282b46ef92004ba52b931a510b46d990f04ca5f6999633b0418846010a6586e1 (equal to the result SHA-256) |

```
SELECT count() AS minutes, min(l2_blocks) AS min_blocks_per_min, max(max_gap_s) AS max_gap_s_day, countIf(l2_blocks < 500) AS minutes_under_500_blocks, countIf(l1_advance = 0) AS minutes_l1_ref_flat, sum(l2_blocks) AS blocks_day, sum(tx_s) AS txs_day FROM (SELECT toStartOfMinute(toDateTime(timestamp, 'UTC')) AS m, count() AS l2_blocks, max(gap_s) AS max_gap_s, max(l1_block_number) - min(l1_block_number) AS l1_advance, sum(tx_count) AS txs FROM (SELECT number, timestamp, l1_block_number, tx_count, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks WHERE timestamp >= 1788479940 AND timestamp < 1788566400) WHERE timestamp >= 1788480000 GROUP BY m)
```

P2 — PP005_window (pass arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:27:56Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 87169581e67c858c95d342f9bff6fbd7b1b8f04d7e1b7ef7cfe5db88323f1ea8 |
| Columns (types) | minute_utc (DateTime('UTC')), l2_blocks (UInt64), first_block (UInt64), last_block (UInt64), max_gap_s (Int64), l1_min (UInt64), l1_max (UInt64), txs (UInt64) |
| Row count | 60 |
| Truncated | false |
| Result SHA-256 | 729320eb9e519c0be8612f85a05dd84b95e6d5c171b3242cc9634370f31e5a9f |
| Run record file | runs\87169581e67c858c95d342f9bff6fbd7b1b8f04d7e1b7ef7cfe5db88323f1ea8.json, SHA-256 50a394e678c53ec8dbe63ae771d47c9e79bcd2001fe1d027a7bfd0e7889f2b9b |
| Result file | results\87169581e67c858c95d342f9bff6fbd7b1b8f04d7e1b7ef7cfe5db88323f1ea8.json, SHA-256 729320eb9e519c0be8612f85a05dd84b95e6d5c171b3242cc9634370f31e5a9f (equal to the result SHA-256) |

```
SELECT toStartOfMinute(toDateTime(timestamp, 'UTC')) AS minute_utc, count() AS l2_blocks, min(number) AS first_block, max(number) AS last_block, max(gap_s) AS max_gap_s, min(l1_block_number) AS l1_min, max(l1_block_number) AS l1_max, sum(tx_count) AS txs FROM (SELECT number, timestamp, l1_block_number, tx_count, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks WHERE timestamp >= 1788524940 AND timestamp < 1788528600) WHERE timestamp >= 1788525000 GROUP BY minute_utc ORDER BY minute_utc
```

R1 — PP005_refute_full (refute arm)

| | |
|--------|----------------------------|
| Engine | clickhouse (store surface) |
|--------|----------------------------|

| | |
|-------------------|---|
| Run at (UTC) | 2026-09-25T14:27:56Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be59143709b3 |
| Columns (types) | block_after_gap (UInt64), resumed_utc (DateTime('UTC')), gap_s (Int64) |
| Row count | 5 |
| Truncated | false |
| Result SHA-256 | 69c781018e1a1c5fd3b8b5a17c117256e7436a8b4c233c52bab40fa1ab82c001 |
| Run record file | runs\7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be59143709b3.json, SHA-256 c9a4e3c762a2f8624d8e3fd39305346fd75f8f7f786f84b9211569b4ad57ad02 |
| Result file | results\7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be59143709b3.json, SHA-256 69c781018e1a1c5fd3b8b5a17c117256e7436a8b4c233c52bab40fa1ab82c001 (equal to the result SHA-256) |

```
SELECT number AS block_after_gap, toDateTime(timestamp, 'UTC') AS resumed_utc, gap_s FROM (SELECT number, timestamp, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks) WHERE number > 0 ORDER BY gap_s DESC, number ASC LIMIT 5
```

R2 — PP005_refute_mainnet (refute arm)

| | |
|-------------------|--|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:27:59Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f |
| Columns (types) | block_after_gap (UInt64), resumed_utc (DateTime('UTC')), gap_s (Int64) |
| Row count | 5 |
| Truncated | false |
| Result SHA-256 | 6acf6b3c9fe4274b18870a922b6087fde9a1624436e79b135b53f20c07281997 |
| Run record file | runs\b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f.json, SHA-256 93672165520f11e90e0350ea172df100c6d7022055164ca3ea5b86be122a6d4e |

| | |
|-------------|---|
| Result file | results\b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f.json, SHA-256 6acf6b3c9fe4274b18870a922b6087fde9a1624436e79b135b53f20c07281997 (equal to the result SHA-256) |
|-------------|---|

```
SELECT number AS block_after_gap, toDateTime(timestamp, 'UTC') AS resumed_utc, gap_s FROM (SELECT number, timestamp, timestamp - lagInFrame(timestamp) OVER (ORDER BY number) AS gap_s FROM glasshull.blocks WHERE timestamp >= 1782863000) WHERE timestamp >= 1782864000 ORDER BY gap_s DESC, number ASC LIMIT 5
```

PART 7 — THE TWO MEASUREMENTS, SIDE BY SIDE

| | Route A (maker) | Route B (checker) |
|-------------------|--|--|
| What it is | the frozen statements re-run on the store (ClickHouse) | DuckDB over the A2 bronze extraction |
| Object file | PP005-maker-v1.json | PP005-checker-v1.json |
| File SHA-256 | 65fed4206040ccdbfd6f526748e1e708a473eef31d51551957d9f745bd213bc1 | 65fed4206040ccdbfd6f526748e1e708a473eef31d51551957d9f745bd213bc1 |
| Canonical SHA-256 | 64fcaa140b467682c924a3298873dd448fa785928c66139ddf1c6f750fd0d021 | 64fcaa140b467682c924a3298873dd448fa785928c66139ddf1c6f750fd0d021 |

Verdict: CLEARED - both arms emit the identical canonical object. Differing leaves: none. Label: engine-independent, lineage-shared (the A1 store was loaded from this bronze). Banked as kind-3 3159198.

Both canonical digests recompute from the two object files under Appendix S §S.1.

7.1 The day, 4 September 2026 (P1)

| Field | Route A | Route B | Difference |
|--------------------------|----------|----------|------------|
| minutes | 1440 | 1440 | 0 |
| min_blocks_per_min | 548 | 548 | 0 |
| max_gap_s_day | 2 | 2 | 0 |
| minutes_under_500_blocks | 0 | 0 | 0 |
| minutes_l1_ref_flat | 0 | 0 | 0 |
| blocks_day | 854255 | 854255 | 0 |
| txs_day | 13978496 | 13978496 | 0 |

7.2 The five longest intervals, full history (R1)

| # | Route A: block · resumed (UTC) · gap s | Route B: block · resumed (UTC) · gap s | Difference |
|---|--|--|------------|
| 1 | 1 · 2026-04-30 16:52:11 · 1777567931 | 1 · 2026-04-30 16:52:11 · 1777567931 | 0 · 0 · 0 |
| 2 | 36 · 2026-05-04 18:08:47 · 252948 | 36 · 2026-05-04 18:08:47 · 252948 | 0 · 0 · 0 |
| 3 | 234 · 2026-05-11 01:20:39 · 186880 | 234 · 2026-05-11 01:20:39 · 186880 | 0 · 0 · 0 |
| 4 | 34 · 2026-05-01 19:37:47 · 81912 | 34 · 2026-05-01 19:37:47 · 81912 | 0 · 0 · 0 |
| 5 | 147 · 2026-05-08 05:51:08 · 64965 | 147 · 2026-05-08 05:51:08 · 64965 | 0 · 0 · 0 |

7.3 The five longest intervals, since public mainnet (R2)

| # | Route A: block · resumed (UTC) · gap s | Route B: block · resumed (UTC) · gap s | Difference |
|---|--|--|------------|
| 1 | 678037 · 2026-07-01 05:35:35 · 16 | 678037 · 2026-07-01 05:35:35 · 16 | 0 · 0 · 0 |
| 2 | 677988 · 2026-07-01 05:33:26 · 13 | 677988 · 2026-07-01 05:33:26 · 13 | 0 · 0 · 0 |
| 3 | 678020 · 2026-07-01 05:34:33 · 13 | 678020 · 2026-07-01 05:34:33 · 13 | 0 · 0 · 0 |
| 4 | 678022 · 2026-07-01 05:34:49 · 12 | 678022 · 2026-07-01 05:34:49 · 12 | 0 · 0 · 0 |
| 5 | 678195 · 2026-07-01 05:39:18 · 11 | 678195 · 2026-07-01 05:39:18 · 11 | 0 · 0 · 0 |

7.4 The reported window, 60 minutes (P2)

Route A's rows are printed in full in Part 9.2. For each minute, the seven numeric fields of route B are compared with route A's; the difference column is the sum of the absolute differences.

| Minute (UTC) | Blocks, A | Blocks, B | Transactions, A | Transactions, B | Fields compared | Difference |
|---------------------|-----------|-----------|-----------------|-----------------|-----------------|------------|
| 2026-09-04 12:30:00 | 563 | 563 | 17423 | 17423 | 7 | 0 |
| 2026-09-04 12:31:00 | 586 | 586 | 15996 | 15996 | 7 | 0 |
| 2026-09-04 12:32:00 | 620 | 620 | 13087 | 13087 | 7 | 0 |
| 2026-09-04 12:33:00 | 590 | 590 | 11606 | 11606 | 7 | 0 |
| 2026-09-04 12:34:00 | 584 | 584 | 13328 | 13328 | 7 | 0 |
| 2026-09-04 12:35:00 | 580 | 580 | 12327 | 12327 | 7 | 0 |
| 2026-09-04 12:36:00 | 581 | 581 | 11772 | 11772 | 7 | 0 |
| 2026-09-04 12:37:00 | 584 | 584 | 12780 | 12780 | 7 | 0 |
| 2026-09-04 12:38:00 | 605 | 605 | 11027 | 11027 | 7 | 0 |
| 2026-09-04 12:39:00 | 599 | 599 | 7197 | 7197 | 7 | 0 |
| 2026-09-04 12:40:00 | 592 | 592 | 8665 | 8665 | 7 | 0 |
| 2026-09-04 12:41:00 | 579 | 579 | 8343 | 8343 | 7 | 0 |
| 2026-09-04 12:42:00 | 588 | 588 | 9458 | 9458 | 7 | 0 |
| 2026-09-04 12:43:00 | 587 | 587 | 11107 | 11107 | 7 | 0 |
| 2026-09-04 12:44:00 | 590 | 590 | 7001 | 7001 | 7 | 0 |
| 2026-09-04 12:45:00 | 596 | 596 | 9943 | 9943 | 7 | 0 |
| 2026-09-04 12:46:00 | 590 | 590 | 10619 | 10619 | 7 | 0 |
| 2026-09-04 12:47:00 | 591 | 591 | 10066 | 10066 | 7 | 0 |
| 2026-09-04 12:48:00 | 589 | 589 | 11819 | 11819 | 7 | 0 |
| 2026-09-04 12:49:00 | 592 | 592 | 8452 | 8452 | 7 | 0 |
| 2026-09-04 12:50:00 | 593 | 593 | 8170 | 8170 | 7 | 0 |
| 2026-09-04 12:51:00 | 581 | 581 | 7322 | 7322 | 7 | 0 |
| 2026-09-04 12:52:00 | 622 | 622 | 6065 | 6065 | 7 | 0 |
| 2026-09-04 12:53:00 | 593 | 593 | 5121 | 5121 | 7 | 0 |

| Minute (UTC) | Blocks, A | Blocks, B | Transactions, A | Transactions, B | Fields compared | Difference |
|---------------------|-----------|-----------|-----------------|-----------------|-----------------|------------|
| 2026-09-04 12:54:00 | 592 | 592 | 4818 | 4818 | 7 | 0 |
| 2026-09-04 12:55:00 | 592 | 592 | 5029 | 5029 | 7 | 0 |
| 2026-09-04 12:56:00 | 592 | 592 | 5702 | 5702 | 7 | 0 |
| 2026-09-04 12:57:00 | 592 | 592 | 5868 | 5868 | 7 | 0 |
| 2026-09-04 12:58:00 | 626 | 626 | 4575 | 4575 | 7 | 0 |
| 2026-09-04 12:59:00 | 593 | 593 | 5371 | 5371 | 7 | 0 |
| 2026-09-04 13:00:00 | 588 | 588 | 4795 | 4795 | 7 | 0 |
| 2026-09-04 13:01:00 | 583 | 583 | 4245 | 4245 | 7 | 0 |
| 2026-09-04 13:02:00 | 592 | 592 | 5225 | 5225 | 7 | 0 |
| 2026-09-04 13:03:00 | 591 | 591 | 4860 | 4860 | 7 | 0 |
| 2026-09-04 13:04:00 | 657 | 657 | 6358 | 6358 | 7 | 0 |
| 2026-09-04 13:05:00 | 591 | 591 | 5312 | 5312 | 7 | 0 |
| 2026-09-04 13:06:00 | 591 | 591 | 8107 | 8107 | 7 | 0 |
| 2026-09-04 13:07:00 | 591 | 591 | 11767 | 11767 | 7 | 0 |
| 2026-09-04 13:08:00 | 592 | 592 | 9677 | 9677 | 7 | 0 |
| 2026-09-04 13:09:00 | 590 | 590 | 9539 | 9539 | 7 | 0 |
| 2026-09-04 13:10:00 | 592 | 592 | 9225 | 9225 | 7 | 0 |
| 2026-09-04 13:11:00 | 601 | 601 | 8429 | 8429 | 7 | 0 |
| 2026-09-04 13:12:00 | 592 | 592 | 8497 | 8497 | 7 | 0 |
| 2026-09-04 13:13:00 | 592 | 592 | 8816 | 8816 | 7 | 0 |
| 2026-09-04 13:14:00 | 593 | 593 | 7923 | 7923 | 7 | 0 |
| 2026-09-04 13:15:00 | 592 | 592 | 8561 | 8561 | 7 | 0 |
| 2026-09-04 13:16:00 | 592 | 592 | 8152 | 8152 | 7 | 0 |
| 2026-09-04 13:17:00 | 620 | 620 | 7806 | 7806 | 7 | 0 |
| 2026-09-04 13:18:00 | 592 | 592 | 8732 | 8732 | 7 | 0 |
| 2026-09-04 13:19:00 | 591 | 591 | 8577 | 8577 | 7 | 0 |
| 2026-09-04 13:20:00 | 590 | 590 | 9150 | 9150 | 7 | 0 |
| 2026-09-04 13:21:00 | 588 | 588 | 10811 | 10811 | 7 | 0 |
| 2026-09-04 13:22:00 | 577 | 577 | 11067 | 11067 | 7 | 0 |
| 2026-09-04 13:23:00 | 624 | 624 | 10608 | 10608 | 7 | 0 |
| 2026-09-04 13:24:00 | 590 | 590 | 10203 | 10203 | 7 | 0 |
| 2026-09-04 13:25:00 | 591 | 591 | 10124 | 10124 | 7 | 0 |
| 2026-09-04 13:26:00 | 591 | 591 | 9749 | 9749 | 7 | 0 |
| 2026-09-04 13:27:00 | 588 | 588 | 13519 | 13519 | 7 | 0 |
| 2026-09-04 13:28:00 | 591 | 591 | 11414 | 11414 | 7 | 0 |

| Minute (UTC) | Blocks, A | Blocks, B | Transactions, A | Transactions, B | Fields compared | Difference |
|---------------------|-----------|-----------|-----------------|-----------------|-----------------|------------|
| 2026-09-04 13:29:00 | 590 | 590 | 11398 | 11398 | 7 | 0 |

Fields compared in the window: 420; total difference: 0.

PART 8 — THE REFUTE ARM

Quoted verbatim from the sealed Proof Posit PP-005, section III.i, Refute arm. Source SHA-256

3ae1d260dc0a2186ae6dbe0613c68b7e4c4e52e15fb6ffc8daabafd8fd2cb8ff, the source recorded by seal event kind-15 3171889 for JFD-20260926-PP-00001.

Refute arm. The same interval measurement returns long silences where they exist. Over the chain's full history it finds intervals of 252,948 s, 186,880 s, 81,912 s and 64,965 s, all during the pre-launch period of April–May 2026, before public mainnet. The instrument therefore does detect a stop when one is present.

- Full history: re-run 7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be59143709b3
- Since public mainnet: re-run b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f

R1 — PP005_refute_full

The method's reporting rule (Part 3.2): **Reporting rule for R1.** The row for block 1 is the interval from the genesis block, whose recorded timestamp is 0; it is reported and labelled as the genesis artifact, never presented as a stop.

| block_after_gap | resumed_utc | gap_s | reading |
|-----------------|---------------------|------------|---|
| 1 | 2026-04-30 16:52:11 | 1777567931 | genesis artifact (see the reporting rule); not a stop |
| 36 | 2026-05-04 18:08:47 | 252948 | pre-launch interval |
| 234 | 2026-05-11 01:20:39 | 186880 | pre-launch interval |
| 34 | 2026-05-01 19:37:47 | 81912 | pre-launch interval |
| 147 | 2026-05-08 05:51:08 | 64965 | pre-launch interval |

Re-run hash 7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be59143709b3 · result SHA-256

69c781018e1a1c5fd3b8b5a17c117256e7436a8b4c233c52bab40fa1ab82c001.

R2 — PP005_refute_mainnet

| block_after_gap | resumed_utc | gap_s |
|-----------------|---------------------|-------|
| 678037 | 2026-07-01 05:35:35 | 16 |
| 677988 | 2026-07-01 05:33:26 | 13 |
| 678020 | 2026-07-01 05:34:33 | 13 |
| 678022 | 2026-07-01 05:34:49 | 12 |
| 678195 | 2026-07-01 05:39:18 | 11 |

Re-run hash b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f · result SHA-256 6acf6b3c9fe4274b18870a922b6087fde9a1624436e79b135b53f20c07281997.

PART 9 — THE MEASURED TABLES

9.1 Block production, 4 September 2026, the whole UTC day (P1)

| minute
s | min_blocks_per_min | max_gap_s_day | minutes_under_500_blocks | minutes_l1_ref_flat | blocks_day | txs_day |
|-------------|--------------------|---------------|--------------------------|---------------------|------------|----------|
| 1440 | 548 | 2 | 0 | 0 | 854255 | 13978496 |

9.2 Block production, minute by minute, 12:30–13:29 UTC (P2)

| minute_utc | l2_blocks | first_block | last_block | max_gap_s | l1_min | l1_max | txs |
|---------------------|-----------|-------------|------------|-----------|----------|----------|-------|
| 2026-09-04 12:30:00 | 563 | 54266140 | 54266702 | 2 | 25903963 | 25903968 | 17423 |
| 2026-09-04 12:31:00 | 586 | 54266703 | 54267288 | 1 | 25903968 | 25903973 | 15996 |
| 2026-09-04 12:32:00 | 620 | 54267289 | 54267908 | 1 | 25903973 | 25903978 | 13087 |
| 2026-09-04 12:33:00 | 590 | 54267909 | 54268498 | 1 | 25903978 | 25903983 | 11606 |
| 2026-09-04 12:34:00 | 584 | 54268499 | 54269082 | 1 | 25903983 | 25903988 | 13328 |
| 2026-09-04 12:35:00 | 580 | 54269083 | 54269662 | 1 | 25903988 | 25903993 | 12327 |
| 2026-09-04 12:36:00 | 581 | 54269663 | 54270243 | 1 | 25903993 | 25903998 | 11772 |
| 2026-09-04 12:37:00 | 584 | 54270244 | 54270827 | 1 | 25903998 | 25904003 | 12780 |
| 2026-09-04 12:38:00 | 605 | 54270828 | 54271432 | 1 | 25904003 | 25904008 | 11027 |
| 2026-09-04 12:39:00 | 599 | 54271433 | 54272031 | 1 | 25904008 | 25904013 | 7197 |
| 2026-09-04 12:40:00 | 592 | 54272032 | 54272623 | 1 | 25904013 | 25904018 | 8665 |
| 2026-09-04 12:41:00 | 579 | 54272624 | 54273202 | 2 | 25904018 | 25904023 | 8343 |
| 2026-09-04 12:42:00 | 588 | 54273203 | 54273790 | 1 | 25904023 | 25904028 | 9458 |
| 2026-09-04 12:43:00 | 587 | 54273791 | 54274377 | 1 | 25904028 | 25904033 | 11107 |
| 2026-09-04 12:44:00 | 590 | 54274378 | 54274967 | 1 | 25904033 | 25904037 | 7001 |
| 2026-09-04 12:45:00 | 596 | 54274968 | 54275563 | 1 | 25904038 | 25904042 | 9943 |
| 2026-09-04 12:46:00 | 590 | 54275564 | 54276153 | 1 | 25904043 | 25904047 | 10619 |
| 2026-09-04 12:47:00 | 591 | 54276154 | 54276744 | 1 | 25904048 | 25904052 | 10066 |
| 2026-09-04 12:48:00 | 589 | 54276745 | 54277333 | 1 | 25904052 | 25904057 | 11819 |
| 2026-09-04 12:49:00 | 592 | 54277334 | 54277925 | 1 | 25904057 | 25904062 | 8452 |
| 2026-09-04 12:50:00 | 593 | 54277926 | 54278518 | 1 | 25904062 | 25904067 | 8170 |
| 2026-09-04 12:51:00 | 581 | 54278519 | 54279099 | 1 | 25904067 | 25904072 | 7322 |
| 2026-09-04 12:52:00 | 622 | 54279100 | 54279721 | 1 | 25904072 | 25904077 | 6065 |

| minute_utc | l2_blocks | first_block | last_block | max_gap_s | l1_min | l1_max | txs |
|---------------------|-----------|-------------|------------|-----------|----------|----------|-------|
| 2026-09-04 12:53:00 | 593 | 54279722 | 54280314 | 1 | 25904077 | 25904082 | 5121 |
| 2026-09-04 12:54:00 | 592 | 54280315 | 54280906 | 1 | 25904082 | 25904087 | 4818 |
| 2026-09-04 12:55:00 | 592 | 54280907 | 54281498 | 1 | 25904087 | 25904092 | 5029 |
| 2026-09-04 12:56:00 | 592 | 54281499 | 54282090 | 1 | 25904092 | 25904097 | 5702 |
| 2026-09-04 12:57:00 | 592 | 54282091 | 54282682 | 1 | 25904097 | 25904102 | 5868 |
| 2026-09-04 12:58:00 | 626 | 54282683 | 54283308 | 1 | 25904102 | 25904107 | 4575 |
| 2026-09-04 12:59:00 | 593 | 54283309 | 54283901 | 1 | 25904107 | 25904112 | 5371 |
| 2026-09-04 13:00:00 | 588 | 54283902 | 54284489 | 1 | 25904112 | 25904117 | 4795 |
| 2026-09-04 13:01:00 | 583 | 54284490 | 54285072 | 1 | 25904117 | 25904122 | 4245 |
| 2026-09-04 13:02:00 | 592 | 54285073 | 54285664 | 1 | 25904122 | 25904127 | 5225 |
| 2026-09-04 13:03:00 | 591 | 54285665 | 54286255 | 1 | 25904127 | 25904132 | 4860 |
| 2026-09-04 13:04:00 | 657 | 54286256 | 54286912 | 1 | 25904132 | 25904137 | 6358 |
| 2026-09-04 13:05:00 | 591 | 54286913 | 54287503 | 1 | 25904137 | 25904142 | 5312 |
| 2026-09-04 13:06:00 | 591 | 54287504 | 54288094 | 1 | 25904142 | 25904147 | 8107 |
| 2026-09-04 13:07:00 | 591 | 54288095 | 54288685 | 1 | 25904147 | 25904152 | 11767 |
| 2026-09-04 13:08:00 | 592 | 54288686 | 54289277 | 1 | 25904152 | 25904157 | 9677 |
| 2026-09-04 13:09:00 | 590 | 54289278 | 54289867 | 1 | 25904157 | 25904162 | 9539 |
| 2026-09-04 13:10:00 | 592 | 54289868 | 54290459 | 1 | 25904162 | 25904167 | 9225 |
| 2026-09-04 13:11:00 | 601 | 54290460 | 54291060 | 1 | 25904167 | 25904172 | 8429 |
| 2026-09-04 13:12:00 | 592 | 54291061 | 54291652 | 1 | 25904172 | 25904177 | 8497 |
| 2026-09-04 13:13:00 | 592 | 54291653 | 54292244 | 1 | 25904177 | 25904182 | 8816 |
| 2026-09-04 13:14:00 | 593 | 54292245 | 54292837 | 1 | 25904182 | 25904187 | 7923 |
| 2026-09-04 13:15:00 | 592 | 54292838 | 54293429 | 1 | 25904187 | 25904192 | 8561 |
| 2026-09-04 13:16:00 | 592 | 54293430 | 54294021 | 1 | 25904192 | 25904197 | 8152 |
| 2026-09-04 13:17:00 | 620 | 54294022 | 54294641 | 1 | 25904197 | 25904201 | 7806 |
| 2026-09-04 13:18:00 | 592 | 54294642 | 54295233 | 1 | 25904201 | 25904206 | 8732 |
| 2026-09-04 13:19:00 | 591 | 54295234 | 54295824 | 1 | 25904206 | 25904211 | 8577 |
| 2026-09-04 13:20:00 | 590 | 54295825 | 54296414 | 1 | 25904211 | 25904216 | 9150 |
| 2026-09-04 13:21:00 | 588 | 54296415 | 54297002 | 1 | 25904216 | 25904221 | 10811 |
| 2026-09-04 13:22:00 | 577 | 54297003 | 54297579 | 2 | 25904221 | 25904226 | 11067 |
| 2026-09-04 13:23:00 | 624 | 54297580 | 54298203 | 1 | 25904226 | 25904231 | 10608 |
| 2026-09-04 13:24:00 | 590 | 54298204 | 54298793 | 1 | 25904231 | 25904236 | 10203 |
| 2026-09-04 13:25:00 | 591 | 54298794 | 54299384 | 1 | 25904236 | 25904241 | 10124 |
| 2026-09-04 13:26:00 | 591 | 54299385 | 54299975 | 1 | 25904241 | 25904246 | 9749 |
| 2026-09-04 13:27:00 | 588 | 54299976 | 54300563 | 1 | 25904246 | 25904251 | 13519 |

| minute_utc | l2_blocks | first_block | last_block | max_gap_s | l1_min | l1_max | txs |
|---------------------|-----------|-------------|------------|-----------|----------|----------|-------|
| 2026-09-04 13:28:00 | 591 | 54300564 | 54301154 | 1 | 25904251 | 25904256 | 11414 |
| 2026-09-04 13:29:00 | 590 | 54301155 | 54301744 | 1 | 25904256 | 25904261 | 11398 |

PART 10 — RE-RUN HASHES AND REPRODUCTION

10.1 Re-run hashes

| State ment | Name | Ar m | Run at (UTC) | Re-run hash (the question) | Result SHA-256 (the answer) |
|------------|-----------------------|---------|----------------------|--|--|
| P1 | PP005_day | pas s | 2026-09-25T14:27:56Z | 17fedf045ed4060954b8c6a1d95ab5721c194a3a387b9d5cbfb04c93529a31f6 | 282b46ef92004ba52b931a510b46d990f04ca5f6999633b0418846010a6586e1 |
| P2 | PP005_win dow | pas s | 2026-09-25T14:27:56Z | 87169581e67c858c95d342f9bfff6fb d7b1b8f04d7e1b7ef7cfe5db88323f1ea8 | 729320eb9e519c0be8612f85a05dd84b95e6d5c171b3242cc9634370f31e5a9f |
| R1 | PP005_refu te_full | ref ute | 2026-09-25T14:27:56Z | 7bdc155659455ce77dd0e716402113311022baf3aff6d04a8e33be59143709b3 | 69c781018e1a1c5fd3b8b5a17c117256e7436a8b4c233c52bab40fa1ab82c001 |
| R2 | PP005_refu te_mainnet | ref ute | 2026-09-25T14:27:59Z | b86da9693c639a1fdd62c94c7ff40040cb184913447e45fc16372bdac25d0e8f | 6acf6b3c9fe4274b18870a922b6087fde9a1624436e79b135b53f20c07281997 |

Each re-run hash recomputes from its run record under Appendix S §S.5, and each result SHA-256 equals the SHA-256 of the stored result file, which is the canonical JSON of the result object (§S.2).

10.2 How to reproduce this without us

From version 1, section 10, with its cross-references changed to this layout.

1. Obtain chain data for Robinhood Chain (4663) covering blocks 0 .. 62,471,946 (Appendix R §R.1).
2. Confirm block 62,471,946 has hash
0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (§R.2). If not, stop.
3. Prove your coverage is continuous (§R.6).
4. Run each statement in Part 6 verbatim against your copy.
5. Serialize each result as Appendix S specifies and compare its SHA-256 with the result SHA-256 in Part 6. Matching both the re-run hash and the result SHA-256 is the strongest confirmation.
6. If a result differs, follow §R.11.

10.3 Recomputing our hashes

The objects below are hashed as canonical JSON (Appendix S §S.1). Every value is copied from the run records.

Coverage fingerprint (§S.4):

```
{"engine":"clickhouse","intervals":[[0,62471946]],"partitions":627}
```

SHA-256: df27b42a736fc8de2d9fb6b69067303d162b72efe116c8bd95c1514d4fa7396d.

Re-run hash (§§.5), for any statement in Part 6:

```
{"block_hash":"0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb","chain_id":4663,"coverage_fingerprint":"df27b42a736fc8de2d9fb6b69067303d162b72efe116c8bd95c1514d4fa7396d","engine":"clickhouse","pin_version":2,"statement":"<the statement exactly as printed in Part 6>"}
```

READING APPENDICES R AND S WITH THIS LAYOUT

Appendices R and S are the standard appendices issued with every Glass Hull Full Proof. They refer to the section numbers of the earlier layout. In this Full Proof those references resolve as follows.

| Appendix reference | In this Full Proof |
|--|--|
| §2 (method; decoding and population rules) | Part 3.2 and 3.3 (and Part 4.1) |
| §3 (queries, expected outputs, result SHA-256) | Part 6, with the full result rows in Parts 8 and 9 |
| §8 (definitions) | the Definitions paragraph of the frozen method, Part 3.2 |
| §10 (how to reproduce this without us) | Part 10.2 |

APPENDIX R — REPRODUCTION DISCIPLINE

Standard appendix to every Glass Hull Full Proof · seat draft 2026-09-24 · pairs with Full Proof §10, "How to reproduce this without us"

This appendix sets out the discipline under which your run should return our bytes, not merely our numbers. Every rule here comes from an error we made or caught in our own work. Follow them in order.

R.1 WHAT YOU NEED

- 1. Data access.** You need one of the following for Robinhood Chain (chain ID 4663):
 - (a) A full or archive node you operate. Required for any finding that reads contract state at a past block.
 - (b) A complete extraction of blocks, transactions, receipts and logs over the stated window.
 - (c) A paid RPC plan. This suffices only for findings marked *RPC-reproducible* in §R.9.
- 2. The method section of this Full Proof (§2) and its queries (§3), used verbatim.** Paraphrasing a query is how results drift.
- 3. Integer arithmetic of at least 128 bits, 256 preferred,** in your query engine or language.

R.2 PIN BEFORE YOU RUN

1. Every result is stated at a **pinned block: number and hash**. Read at that block, never at "latest". A block number alone does not identify which chain state produced a figure.
2. Where you read contract state, request it at the pinned block, by hash where your endpoint supports it (EIP-1898).
3. Record the pinned hash your endpoint returns for the pinned number. **If it differs from ours, stop. You are not reading the same chain state**, and no later comparison is meaningful.

R.3 EXACT INTEGERS ONLY

1. **No floating-point arithmetic, anywhere a value is summed, multiplied or compared.** Amounts stay in wei, or in the token's base units, until final display. A 64-bit float holds about 15–16 significant digits; wei amounts routinely exceed that, and the loss is silent.
2. Convert to ETH, USDG or any display unit **once, at the end**, and state the conversion.
3. Every token's decimals are stated in this Full Proof (for example, USDG = 6). Do not assume 18.

R.4 DECODE AS WE DECODE

1. **Addresses:** compare as lowercase hexadecimal without checksum casing.
2. **Byte-encoded values:** where a field is stored as raw bytes, decode it as the Full Proof states. For example, a gas price stored as big-endian bytes is reversed to little-endian before integer interpretation. Every such rule for this finding is listed in §2.
3. **Event data:** decode topics and data exactly as §2 specifies; indexed and non-indexed fields are not interchangeable.
4. **Transaction status:** 1 is success, 0 is failure. System transactions are included or excluded exactly as §2 states.

R.5 TIME

1. **All times are UTC.** Block timestamps have one-second resolution. Some tools default to local time; set UTC explicitly.
2. Every "day", "week" or "window" is defined in §2 by **block range**, with its UTC boundaries. Use the block range.

R.6 PROVE YOUR POPULATION IS CLOSED BEFORE ANY COUNT MEANS ANYTHING

1. **Coverage.** Your data must contain every block in the stated range, with no gaps: the number of distinct blocks equals $(last - first + 1)$. Prove this before running any count.

2. **Duplicates.** Remove duplicates explicitly and record how many you removed. A storage key or primary key does not guarantee uniqueness.
3. **Provider limits.** Many RPC providers cap log queries by block span or result count and **truncate silently**. Page by block range and prove each page returned complete.
4. **Tables that end early.** If you join to a helper table (block times, prices, labels), confirm it covers your whole range. A helper that ends before your data silently drops every later row.

R.7 SAME DEFINITIONS, STATED BEFORE THE RESULT

1. Use the definitions published in §8 of this Full Proof, which were fixed before computation. Where a public claim uses a different definition, we state both, and the result under each.
 - *Example:* whether a "burn" counts transfers to the zero address only, or also to a dead address, changes the verdict on at least one public claim.
2. Do not substitute a definition you prefer and then compare numbers. That is a different measurement.

R.8 ORDERING AND BYTES

1. **Every multi-row result is totally ordered**, as §3 specifies. Without a total order, correct rows arrive in varying order and the result hash changes.
2. **To compare hashes, serialize exactly as our canonical serialization specifies** (Appendix S): column order, types, encoding, separators and number formatting. Matching our re-run hash, not just our figures, is the strongest confirmation you can give.
3. If your figures match and your hash does not, check serialization before concluding the result differs.

R.9 BOTH ARMS, IN ORDER

1. **Run the pass arm first**, then the refute arm. The refute arm shows the same method returns the opposite result when that result is true. An empty or null result is evidence only when its refute arm passes.
2. Expected outputs for both arms, with their re-run hashes, are in §3.
3. **Reproducibility class of this finding:** needs a full node or complete extraction (every query in §3 scans the full window).

R.10 KNOWN PITFALLS

1. Floating-point casts on value columns (§R.3).
2. Local-time defaults (§R.5).
3. Helper tables that end before your data (§R.6.4).
4. Unbounded queries over a partial local copy, which return a correct answer for the wrong population.

5. Query timeouts that return an empty body rather than an error. Treat a timeout as no result, never as zero.
6. Reading at "latest" instead of the pinned block (§R.2).
7. Assuming a finding measured on an archive node can be reproduced through a metered RPC endpoint (§R.9.3).

R.11 IF YOUR RESULT DIFFERS

1. Re-check §R.2 (the pin) and §R.6 (coverage) first. They account for most differences.
 2. Then §R.3 (floats) and §R.8 (ordering and serialization).
 3. **If the difference stands, tell us before you publish it:** contact@jetfyul.com, citing this Full Proof's DCN, your pinned hash, your coverage proof and your result. A difference we cannot explain is a finding, and it will be treated as one: investigated, and corrected publicly if we are wrong. Under our standing commitment, a corrected Full Proof is issued free to every licensee.
-

Appendix S (canonical serialization) is issued with the first Full Proof released under this appendix and applies to all that follow.

APPENDIX S — CANONICAL SERIALIZATION

Standard appendix to every Glass Hull Full Proof · CC draft 2026-09-25 · pairs with Appendix R §R.8 ("Ordering and bytes")

Appendix R tells you to match our **bytes**, not only our numbers. This appendix defines those bytes: how a result is written down before it is hashed, and how each hash in §3 of this Full Proof is formed. Every rule below was checked against run records of this Full Proof. Recomputing the coverage fingerprint, the re-run hash and the result hash from the records under these rules reproduced all three, for every query checked (S.7).

S.1 CANONICAL JSON

Every object we hash is first written as canonical JSON:

1. **Object keys are sorted** by Unicode code point, at every level.
2. **No insignificant whitespace.** The separator between items is `,`, and between key and value it is `:`. No spaces, no line breaks, no trailing newline.
3. **Encoding is UTF-8.** Non-ASCII characters are written as themselves, not as `\u` escapes. Inside strings, `"` and `\` are backslash-escaped. `\b`, `\f`, `\n`, `\r` and `\t` use those short forms, and any other control character is written as `\u00XX` in lowercase hex.

4. **Integers** are written in plain decimal, with no sign for non-negative values, no leading zeros and no exponent.
5. **No floating-point numbers appear in any hashed object.** Every amount that can exceed 2^{53} is carried as a decimal **string** (`toString(...)` in the queries of §3). A query that casts an evidentiary value to floating point is flagged at run time, and our seal path refuses it.
6. `true`, `false` **and** `null` are written in lowercase.

Reference implementation (Python): `json.dumps(obj, sort_keys=True, separators=(",", ":"), ensure_ascii=False).encode("utf-8")`.

S.2 THE RESULT OBJECT

A query result is hashed as exactly this object, and nothing else:

```
{
  "columns": [<column name>, ...],
  "row_count": <N>,
  "rows": [[<value>, ...], ...],
  "truncated": false,
  "types": [<type name>, ...]
}
```

(The keys are shown here in canonical order.)

- **columns:** the column names, in the order the query's `SELECT` lists them.
- **types:** the store's type name for each column, in the same order (for example `UInt64`, `String`, `Int64`).
- **rows:** one array per row, with values in column order. **Rows are in the order the query's `ORDER BY` fixes.** Every multi-row query in §3 has a total order (Appendix R §R.8.1).
- **row_count:** the number of rows.
- **truncated:** `false` for every result we publish. A truncated result is never evidence.

How values are written:

| store type | written as | example |
|---|--|--------------------------|
| <code>UInt8 ... UInt64</code> , <code>Int8 ... Int64</code> | JSON integer | 854255 |
| sums and wei amounts (<code>toString(...)</code> in the query) | JSON string of decimal digits | "1551080321319255460000" |
| <code>DateTime</code> (UTC) | JSON string YYYY-MM-DD
hh:mm:ss | "2026-09-04 12:57:00" |
| hashes and addresses (<code>lower(hex(...))</code> in the query) | JSON string of lowercase hex, no <code>0x</code> | "6be6970f..." |
| <code>String</code> | JSON string | |

result_sha256 = SHA-256 over the canonical JSON (S.1) of the result object. It is written as 64 lowercase hex characters. It is the hash that proves you obtained **the same result**.

S.3 THE STATEMENT

Before hashing, a query statement is normalised as follows:

1. SQL comments are removed.
2. Surrounding whitespace is trimmed.
3. **One** trailing ; is removed, together with any whitespace before it.

Nothing else changes: internal whitespace, case and line breaks are kept as written. The statement text in §3 is the normalised form. Use it byte for byte.

S.4 THE PIN AND THE COVERAGE FINGERPRINT

- **Pin:** the block number and its hash, written as 0x followed by 64 lowercase hex characters (for this Full Proof, block 62,471,946 = 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb).
- **The coverage fingerprint** identifies the population the query read. It is SHA-256 over the canonical JSON of:

```
{ "engine": "clickhouse", "intervals": [[first_block, last_block], ...], "partitions": P }
```

- **intervals:** the continuous block ranges held, sorted and merged. Two ranges are merged when one ends exactly one block before the next begins.
- **partitions:** the number of loaded partitions.
- For this Full Proof the value is { "engine": "clickhouse", "intervals": [[0, 62471946]], "partitions": 627 }.
- Your own coverage proof (Appendix R §R.6) should produce **the same interval list**. The partition count is a property of our storage layout; use ours when recomputing our hashes.

S.5 THE RE-RUN HASH

re-run hash = SHA-256 over the canonical JSON of:

```
{ "block_hash": "<pin hash, S.4>", "chain_id": 4663, "coverage_fingerprint": "<S.4>", "engine": "clickhouse", "pin_version": 2, "statement": "<S.3>" }
```

The re-run hash identifies the question, not the answer. It commits to the statement, the chain, the population and the pin. The same query at the same pin always has the same re-run hash. **The answer is committed separately by result_sha256 (S.2).** A reproduction matches us when **both** hashes match:

- the re-run hash shows you asked the identical question of the identical population;
- result_sha256 shows you got the identical answer.

S.6 DOCUMENTS

Every sealed document (Proof Posit, Full Proof, pre-registration, notice) is hashed over its **stored bytes**:

1. UTF-8, with no byte-order mark.
2. Line endings are LF only.

3. The file ends with **exactly one** trailing newline.

Document SHA-256 = SHA-256 over those bytes, verbatim. No seal block is embedded in the hashed bytes; the seal record is published on the verify page. Any change, including to whitespace, produces a different document.

S.7 WORKED CHECK

For PP-006's query P1, the record gives:

| value | source |
|--|--|
| re-run hash
3a8a1838fdb4a0e5ffd0bb491b596a50234998944307004bbf4637edca122003 | recomputed from S.3–S.5 |
| result [[32594667, "212060710844437370000"]] with columns ["failed", "fee_failed_wei"] | the query's answer |
| result_sha256 | recomputed from S.2; equals the stored result file's own SHA-256 |

The same check passes for PP-005's P1 (17fedf04...) and PP-006's P6 (91c1eaf4...). These were computed on 2026-09-25 from the run records, not from memory.

S.8 IF YOUR HASH DIFFERS AND YOUR FIGURES MATCH

Check the following, in order:

1. Key order and whitespace (S.1).
2. Integers written as strings, or strings as integers (S.2 table).
3. Row order (S.2; Appendix R §R.8.1).
4. A trailing semicolon or comment left in the statement (S.3).
5. The pin hash's case or the 0x prefix (S.4).

A difference that survives these checks is a finding (Appendix R §R.11).

NOTE ON THIS DOCUMENT

The pages above reproduce the sealed document. The sealed record is a file.

What the Notary authorised, what both hardware witnesses signed and what the ledger event records is the file JFD-20260928-FP-00001 . pdf - 461556 bytes, SHA-256 1c9145483b73df4c8f4ff7f56aa9fbdf76af028b3a21becaaf9f22b6744e68ed.

The sealed document JFD-20260928-FP-00001 . pdf (461556 bytes, SHA-256 1c9145483b73df4c8f4ff7f56aa9fbdf76af028b3a21becaaf9f22b6744e68ed) is embedded in this PDF as a file attachment.

Its exact source, whose fingerprint the document prints, FP-005-v2-source .md (62455 bytes, SHA-256 b8514b6422e2466e5ed61c81faa27ba8e037102b06af25345bf60334412eec3b) is embedded in this PDF as a file attachment.

Appendix R (reproduction discipline), carried verbatim in the exact source, appendix-R-v2-source .md (6456 bytes, SHA-256 7c195b4e34ced101b4bd1e099d6b1f03d86fad500fc5d201ea311255a452dc6b) is embedded in this PDF as a file attachment.

Appendix S (canonical serialization), carried verbatim in the exact source, appendix-S-v2-source .md (6785 bytes, SHA-256 dfd81872f053020404305257bcf9e4473db155cceabc0bb350734ff67d234f65) is embedded in this PDF as a file attachment.

The anchor proof as stamped at sealing JFD-20260928-FP-00001 . pdf .ots (716 bytes, SHA-256 71f6d96a68d49212454fc99b6c3c9389ea9f0c69c0484b9efc696c9ba5d71e6c) is embedded in this PDF as a file attachment.

The seal record from which the certificate is built seal-record-JFD-20260928-FP-00001 . json (7250 bytes, SHA-256 51cd0eb0b627888cdf94e92bad472901c6b4ad3461169ba32169d0cbbc978017) is embedded in this PDF as a file attachment.

The complete anchor proof, carrying the Bitcoin attestation, JFD-20260928-FP-00001 . pdf .anchored .ots (2580 bytes, SHA-256 eaa15712c948252894843d5990305b3563e1915a9b2df8d907f95a415c1e131e) is embedded in this PDF as a file attachment.

The anchor confirmation from which the certificate's Bitcoin block is taken anchor-confirmation-JFD-20260928-FP-00001 . json (2371 bytes, SHA-256 f2746526895537aa375f58a32b8c790be20519b5444b0d506c77b04a825dfa89) is embedded in this PDF as a file attachment.

Open the attachments pane of your PDF reader to extract them, and hash each file as stored.

What follows, in order

The seal page, carrying the seal record's values: fingerprint, ledger event, both witnesses and the anchor. After it, the certificate, built from the same seal record.

K. L. Phillips

K. L. Phillips, a man

Principal, Jet Fyul Dynamics LLC · The author stands behind every word.



BLOCK 2 — ISSUER

ISSUER Jet Fyul Dynamics LLC · Wyoming
RAIL Matchstick Tribunal
CUSTODY both witness keys held by the issuer, in silicon, on separate hosts; never exported

BLOCK 4 — WHO SIGNED IT

Two independent hardware witnesses. Conforming tests of a defined procedure — not a primary and a backup.

| | WITNESS 1 | WITNESS 2 |
|------------------------------|-------------------------------------|-------------------------------------|
| chip identity | a9-atecc608b-slot0 | a10-atecc608b-slot0 |
| serial | 0123ADA0711D8597EE | 0123DC797ACC1195EE |
| curve | ECDSA-P256 (secp256r1) | ECDSA-P256 (secp256r1) |
| public key | d62d03f8dc3813fb5b74f2b64339d185... | 853e69fc700990303abfe61358bfa456... |
| signature over artifact hash | → ledger event 3205872 | → ledger event 3205872 |
| signed (UTC) | → event 3205872 | → event 3205872 |
| federation peer | a10-atecc608b-slot0 | a9-atecc608b-slot0 |
| host · custody | A9 · Jet Fyul Dynamics LLC | A10 · Jet Fyul Dynamics LLC |
| certifies in accordance with | Fed. R. Evid. 902(13), 902(14) | Fed. R. Evid. 902(13), 902(14) |
| verified offline | → event 3205872 | → event 3205872 |

Each signature verifies on its own, against its own public key, without the other and without anything of ours.
One signature is not a seal: if either witness is unreachable, no seal is produced. Both chips are the same make and model, provisioned and owned by the issuer — independent in custody and location, not in supply chain or ownership.
Full public keys, signatures, and the Rule 902(11) certification of a qualified person: ledger event → event 3205872 · <https://gh.jetfyul.com/v/JFD-20260928-FP-00001>

Blocks 1-3 are inside these bytes; Block 4 live cells, Block 5 (Ed25519 a1-ledger) and Block 6 (Bitcoin anchor) live on the ledger and at <https://gh.jetfyul.com/v/JFD-20260928-FP-00001>

CERTIFICATE OF AUTHENTICITY

Robinhood Chain block production - JFD-20260928-FP-00001

The record

Document control number: JFD-20260928-FP-00001

Fingerprint (SHA-256 of the sealed file, raw bytes as stored): 1c9145483b73df4c8f4ff7f56aa9fbdf76af028b3a21becaaf9f22b6744e68ed

Sealed file: JFD-20260928-FP-00001.pdf, 461556 bytes

Source fingerprint printed inside the document: b8514b6422e2466e5ed61c81faa27ba8e037102b06af25345bf60334412eec3b

Ledger event (kind-15): 3205872, event hash bc9e1e56ec40750b0a5fdb1fb657ca02ec3896fc6ea1d054f6a8b80379a060e1

Notary authorisation (kind-3): 3205870

Seal request signed by both witnesses: SHA-256 2c2a55f822519a8ab0e47f1ecf65a8d199b6507b7184f30c5c7dfffb54be917e4

Anchor: OpenTimestamps, **ATTESTED in Bitcoin block 968951** - block hash 0000000000000000000011421e6860144f03bf6dcef6486d1b2a49bff076b3407, block time 2026-09-28T05:44:56Z.

Calendars that attested: <https://finney.calendar.eternitywall.com> (block 968952); <https://alice.btc.calendar.opentimestamps.org> (block 968951). The earliest attested block is the anchor.

Anchor check: the committed value equals block 968951's merkle root b59498e53bacdcd a186b07c496ec83b48e5d6ea44884d52320f127ae3c4c1ebd, and mempool.space and blockstream.info agree on that merkle root and on the block hash (checked 2026-09-28T06:34:26Z).

Proofs: the proof as stamped at sealing, JFD-20260928-FP-00001.pdf.ots and the complete proof JFD-20260928-FP-00001.pdf.anchored.ots, which carries the Bitcoin attestation itself.

Verification address: <https://gh.jetfyul.com/v/JFD-20260928-FP-00001>

The two hardware witnesses

Witness 1 - host A9, a9-atecc608b-slot0, serial 0123ADA0711D8597EE, curve ECDSA-P256 (secp256r1), signed 2026-09-28T05:43:27.434987+00:00

Public key: d62d03f8dc3813fb5b74f2b64339d185742aabcae1bdcfd2f7a7ad59f1dd3c5d50e57effa4a3c888f58c3eca9d274bd7f0b8d263fa38bbf0bc6fbf f6035c2411

Signature: 06d1bfa9b5fef1345f137628c7d99042a601c166f74c3d1f0c603e1e8b40a0d9f238a403f8f9f014c7a31d332a1ee40e521b3ab1d2848593b70b73b e17dd22c4

Witness 2 - host A10, a10-atecc608b-slot0, serial 0123DC797ACC1195EE, curve ECDSA-P256 (secp256r1), signed 2026-09-28T05:43:29.718128+00:00

Public key: 853e69fc700990303abfe61358bfa456160eaae1529dbc74805d1639cfc88e78497bbe060e7ae5d54466b2ce0a2753e14204a21eed4a132a61ace3294f9beba1

Signature: 3c1198d4d24ca2143d229346012dae08dbf5a4586d12cf80dc6a3a5e58c66245146262729dbf4c161f20df56fa2c0b8e03806bdeb69b815d4ea9bf5dc7d530cd

How to verify this without the Issuer, in six steps

(1) Extract JFD-20260928-FP-00001.pdf from this PDF's attachments and hash it as stored, with no normalisation. Its SHA-256 must equal the fingerprint above.

(2) Read ledger event 3205872 by its id. Its payload must carry that fingerprint as doc_sha256 and JFD-20260928-FP-00001 as dcn.

(3) Verify witness 1's signature as ECDSA over P-256, with the public key above, over the seal request bytes whose SHA-256 is 2c2a55f822519a8ab0e47f1ecf65a8d199b6507b7184f30c5c7dfffb54be917e4.

(4) Verify witness 2's signature the same way. Each signature must verify under its own key and fail under the other's.

(5) Read the OpenTimestamps proof against the same fingerprint. It must carry a Bitcoin block-header attestation at block 968951; ask any block explorer for that block and compare its merkle root with the value the proof commits to.

(6) Open <https://gh.jetfyul.com/v/JFD-20260928-FP-00001> and confirm it shows the same fingerprint, ledger event, witnesses and anchor.

This certificate goes to what the record is and that it has not been altered. It is not a sworn declaration and does not assert the truth of anything the document says.