



FULL PROOF

Robinhood Chain failed transactions

The Full Proof for Proof Posit PP-006, version 2 - free, identical for every reader

DOCUMENT CONTROL NUMBER	JFD-20260928-FP-00002
CLASS OF RECORD	resolved
DATE	2026-09-27
COVERAGE	2026-04-30 → 2026-09-14T03:00:01Z
SOURCE FINGERPRINT (SHA-256)	6e2ef2d8b20d6cd2976a56b00e53d3428da57ab75a5683ad74f26a904a6db96e
VERIFICATION	https://gh.jetfyul.com/v/JFD-20260928-FP-00002

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-006 (FREE)

Robinhood Chain failed transactions · version 2

PART 1 — COVER

Field	Value
Full Proof	FP-006, version 2
Accomplices	Proof Posit PP-006, “Nearly One in Five User Transactions Fails on Robinhood Chain, and Still Pays: 1,551.08 ETH in Fees from 119,984,049 Failed Transactions”, sealed as JFD-20260926-PP-00002 (Glass Hull ledger kind-15 3171918; document SHA-256 4005f0a42b1de3374e4564784c688200fb1265475ea4e2220a40a68a89b9152d)
Version 1	Sealed as JFD-20260926-FP-00002 (kind-15 3173079; document SHA-256 855359021b1d998536306042fd7f8b88c5687539d572baf12161c89a7540073f). Its full text (SHA-256 52989de19107e56ce8721e20f10098ff48aba5f25ebb730982ef5cf870752f69) is delivered inside certificate JFD-20260926-CERT-00005 (kind-15 3173086; SHA-256 810553131fdc2ce1d002b9a0be06fc05d4f4b996d492176fe8dbb9a831518c8f). 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 f31bcd7662a6be6f2bd64c96db297e343c7c9ca0b0d29edcf0b7f27e181631a1 (kind-3 3147512); amendment A1 SHA-256 b6cf1c7f7006f2a80db22c1de87beaec1603644c8bbde7e9f59afc0d52740984 (kind-3 3158893)
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 52989de19107e56ce8721e20f10098ff48aba5f25ebb730982ef5cf870752f69).

It carries everything needed to reproduce, without us, the measurement how many USER transactions failed (receipt status 0) from genesis to the pin, what fees they paid to the wei, and that ArbOS system transactions (type 106) neither fail nor pay.

2.2 Summary

Quoted verbatim from the sealed Proof Posit PP-006, section I. Source SHA-256

17f940f50bfe3fd3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0, the source recorded by seal event kind-15 3171918 for JFD-20260926-PP-00002.

From its first block to 14 September 2026, Robinhood Chain processed **653,008,422 user transactions**. Of these, **119,984,049 failed, or 18.374%**. The chain also recorded 62,710,451 system transactions of its own; none failed, and none used gas.

A failed transaction still pays for the computation it used. Together, the failed transactions paid **1,551,080,321,319,255,460,000 wei, about 1,551.08 ETH**, in fees.

Two independent measurements agree on all three figures exactly: the user-transaction count, the failed count, and the fees to the wei.

2.3 Findings

Quoted verbatim from the sealed Proof Posit PP-006, section III (the tables, before III.i). Source SHA-256

17f940f50bfe3fd3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0, the source recorded by seal event kind-15 3171918 for JFD-20260926-PP-00002.

Measure	Value
User transactions, genesis to block 62,471,946	653,008,422
Failed user transactions	119,984,049 (18.374%)
System transactions (chain-internal)	62,710,451 (none failed; zero gas)
All transactions, for reference	715,718,873 (failed share 16.764%)
Fees paid by failed transactions	1,551,080,321,319,255,460,000 wei $\approx$ 1,551.08 ETH

By block range (failed count · fees in wei):

Block range	Failed	Fees (wei)
0 – 19,999,999	32,594,667	212,060,710,844,437,370,000
20,000,000 – 34,999,999	38,719,737	178,878,750,922,993,954,000
35,000,000 – 43,999,999	13,524,416	45,585,008,478,580,212,000
44,000,000 – 52,999,999	16,117,253	356,169,485,660,238,432,000
53,000,000 – 62,471,946	19,027,976	758,386,365,413,005,492,000

How the fees break down under the chain's revenue split. Every settlement to the Arbitrum Expansion Program that we measured split exactly 90% to the network and 10% to Arbitrum's ecosystem (Proof Posit PP-003). We state no allocation beyond what that sealed record shows.

2.4 Limits of observation

Quoted verbatim from the sealed Proof Posit PP-006, section IV. Source SHA-256

17f940f50bfe9d3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0, the source recorded by seal event kind-15 3171918 for JFD-20260926-PP-00002.

1. **Why a transaction failed is not measured here.** Causes include slippage limits, expired quotes, competing transactions and contract rules. This finding counts outcomes, not causes.
2. **ETH is stated at 18 decimals, rounded for display.** The exact figure in wei governs. No dollar value is stated.
3. **Some users paid nothing themselves.** Robinhood's promotion covers network fees on eligible Robinhood Wallet swaps, and sponsors (paymasters) cover gas for some programmable accounts. The fees counted here were paid to the network regardless of who funded them.
4. **The window ends at the pinned block.** Later activity is not included.

2.5 Conclusion

Quoted verbatim from the sealed Proof Posit PP-006, section V. Source SHA-256

17f940f50bfe9d3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0, the source recorded by seal event kind-15 3171918 for JFD-20260926-PP-00002.

Nearly one user transaction in five on Robinhood Chain fails (18.374%). Failure is not free: failed transactions have paid about 1,551.08 ETH in fees over the chain's life to 14 September 2026, exactly counted and independently confirmed.

This finding measures outcomes. It draws no conclusion about any party's systems or intentions.

PART 3 — THE QUESTION AND THE FROZEN METHOD

3.1 The question

Quoted verbatim from the sealed Proof Posit PP-006, section II. Source SHA-256

17f940f50bfe9d3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0, the source recorded by seal event kind-15 3171918 for JFD-20260926-PP-00002.

What we measured. Over the chain's whole history to the pinned block:

- how many transactions failed
- what fees those failed transactions paid

Why it matters. Failed transactions are a cost borne by users and a source of fee revenue for the network. That cost appears in no published figure we could find.

Definitions. A **user transaction** is any transaction other than the chain's own system transactions (receipt type 106). A transaction has **failed** when its receipt records failure (status 0). The **fee** is the gas the transaction used multiplied by the effective price it paid per unit of gas, summed exactly in wei.

3.2 The frozen method, verbatim

Fingerprint (SHA-256)	f31bcd7662a6be6f2bd64c96db297e343c7c9ca0b0d29edcf0b7f27e181631a1
Frozen on the ledger	kind-3 3147512, created 2026-09-25 14:27:04.551949+00
Event hash	3838f10faee17d63d314c71708d9934d42c47830c8c138209d26b5d9917a0564

```
# FROZEN METHOD — PP-006 (failed transactions and the fees they paid)

**Status.** Frozen after exploratory measurement (exploratory runs 2026-09-24 07:20-07:29Z). The sealed figures are the RE-RUN of these statements under this frozen method.

**Population.** Every receipt in `glasshull.receipts`, genesis to the pin: block 62,471,946, hash 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (the store's E0 tip, 2026-09-14T03:00:01Z). USER transactions = every receipt with `type != 106`; `type = 106` is the ArbOS internal system transaction, reported separately and not counted as a user transaction.

**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.** Failed = receipt `status = 0`. Fee = `gas_used * effective_gas_price`, summed exactly in wei (effective_gas_price is stored big-endian and read with `reinterpretAsUInt128(reverse(...))`). Failure rate = user_failed / user_txs, stated as an exact fraction and rounded for display.

### Pass arm — failed count and fees in five block ranges covering the window
### P1 — blocks 0 .. 19,999,999

```sql
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number < 20000000
```

### P2 — blocks 20,000,000 .. 34,999,999

```sql
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number >= 20000000 AND block_number < 35000000
```

### P3 — blocks 35,000,000 .. 43,999,999

```sql
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number >= 35000000 AND block_number < 44000000
```

### P4 — blocks 44,000,000 .. 52,999,999

```sql
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number >= 44000000 AND block_number < 53000000
```
```

```

#### P5 — blocks 53,000,000 .. the pin

```sql
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei, max(block_number) AS last_blk FROM glasshull.receipts WHERE status = 0 AND block_number >= 53000000
```

#### P6 — user vs system split (the headline denominator)

```sql
SELECT countIf(type != 106) AS user_txs, countIf(type != 106 AND status = 0) AS user_failed, countIf(type = 106) AS system_txs, countIf(type = 106 AND status = 0) AS system_failed, sumIf(gas_used, type = 106) AS system_gas_used FROM glasshull.receipts
```

#### Refute arm

#### R1 — independent tally from the per-block table: all transactions minus successful ones

```sql
SELECT count() AS blocks, min(number) AS first_blk, max(number) AS last_blk, sum(txn) AS txn_all, sum(txn) - sum(txn_ok) AS txn_failed FROM glasshull.fees_by_block
```

#### R2 — fee computation checked against the chain's recorded per-block fees at one block

```sql
SELECT (SELECT toString(sum(toUInt256(gas_used) * reinterpretAsUInt256(reverse(effective_gas_price)))) FROM glasshull.receipts WHERE block_number = 40000000) AS receipts_fee_wei, (SELECT toString(toUInt256(l2_fee_wei_dec) + toUInt256(l1_fee_wei_dec)) FROM glasshull.fees_by_block WHERE number = 40000000) AS fees_by_block_l2_plus_l1, (SELECT l2_fee_wei_dec FROM glasshull.fees_by_block WHERE number = 40000000) AS l2_only
```

#### Agreement rules
- sum of P1..P5 failed == R1 txn_failed == P6 user_failed + P6 system_failed.
- R2: receipts_fee_wei == fees_by_block_l2_plus_l1.
- System transactions: P6 system_failed and system_gas_used are reported; if system_gas_used = 0 the fee total is unaffected by the user/system split.
- Second route (independent, already banked): kind-3 3121915 (A8c), population type != 106, must equal the failed count and the failed-fee total to the wei.

#### Reporting
Every figure published is read from the re-run's result bytes; nothing is retyped. Exploratory re-run hashes (record only): PP006_r1 = 3a8a1838fdb4a0e5ffd0bb491b596a50234998944307004bbf4637edca122003, PP006_r2 = 9bb7065fc9a3151dd8d0baa4994a9b236858d451a1638aaf0ba1338bb5a2d1e6, PP006_r3 = c20040b9050994bc02dc6ada4a5ca6244d22f63694d3647824a8a7f7d4ffe705, PP006_r4 = afe1cfc145f1b79a7ebcfee8243de57bd3ed8a79ae320e09260cca150a66db98, PP006_r5 = 669cfdcf23a9c7418bec2fd84a70f28f0434e192fb171fc7c56d70373c589c74e, PP006_refute = 02dd95e4d55ef659ec1d49d5c06a788993ef3899a983d43d959b00278edf84c3, PP006_feesa_mple = 4c21c093f48db57e02203c9d781cafabdfc455d96bf14cd3f8cba188e5dbd8b5.

```

3.3 Amendment A1, verbatim

| | |
|------------------------------|--|
| Fingerprint (SHA-256) | b6cf1c7f7006f2a80db22c1de87beaec1603644c8bbde7e9f59afc0d52740984 |
| Frozen on the ledger | kind-3 3158893, created 2026-09-26 02:56:24.659289+00 |

| | |
|------------------------|--|
| Event hash | 67bdf17c9bf40c30bc052c881c07374baed7d6d606214d3ce4e81ade527c0f49 |
| What it changes | no statement, population, definition or figure; declares the object {question, method_version, pin, user_txs, user_failed, user_failed_fee_wei, system_txs, system_failed} and how each arm derives it |

PP-006 METHOD – AMENDMENT A1: the canonical output object (representation only)

****Amends.**** The frozen method `PP-006-METHOD-frozen.md` (kind-3 3147512, sha256 f31bcd76...). This amendment changes no statement, no population, no definition and no figure. It declares the object each arm emits, so that the two arms can be compared by digest (OUTPUT-CONVENTIONS 1.1). It is frozen before either arm object is built.

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

```
...
{
  "question": "PP-006",
  "method_version": "v1+A1",
  "pin": { "block": <int>, "hash": "<str>" },
  "user_txs": <int>,
  "user_failed": <int>,
  "user_failed_fee_wei": "<dec>",
  "system_txs": <int>,
  "system_failed": <int>
}
```

All keys are literal; no other key exists. Canonical JSON = `json.dumps(obj, sort\_keys=True, separators=(",", ":"), ensure\_ascii=False)` (UTF-8); sha256 over those bytes. Wei is a decimal string; every other number is a JSON integer.

A1.2 Maker: route A, the frozen statements re-run under 3147512

- `pin` = the height and block hash recorded on the re-run records (all must agree).
- `user\_txs`, `user\_failed`, `system\_txs`, `system\_failed` = the P6 re-run's columns of the same names.
- `user\_failed\_fee\_wei` = the sum of `fee\_failed\_wei` over P1..P5. P1..P5 count every failed receipt, of either type. The sum is therefore the USER failed fee only when P6 `system\_failed` = 0. The maker asserts that before it emits, and it emits nothing otherwise.
- The maker also asserts the frozen agreement rule: the sum of P1..P5 `failed` equals P6 `user\_failed` + P6 `system\_failed`.

A1.3 Checker: route B, independent of P1..P6

- `user\_failed`, `user\_failed\_fee\_wei` = the status-0 row of `failed\_fee\_whole\_window\_tip` in the A8c record (kind-3 3121915; file `laundry-a8c-failed-fee-whole-window.json`, sha256 79e5d397...).
 - That statement is a different statement: `type != 106`, GROUP BY status, UInt256 arithmetic. It was run on 2026-09-24, before this method was frozen.
- `user\_txs` = the sum of `txs` over both status rows of the same statement.
- `system\_txs` = R1 `txs\_all` - `user\_txs`.
- `system\_failed` = R1 `txs\_failed` - `user\_failed`.
 - R1 is read from the per-block table `glasshull.fees\_by\_block`, not from receipts.
- `pin` = the height and block hash on the R1 re-run record.
 - The A8c record's tip must be the same block and hash.

****Label.**** Statement-independent, lineage-shared: both routes read the same E0 extraction of the chain on the A1 store. What this comparison shows is that two different statements, over two different tables, reproduce the same five figures. It does not show that two different extractions of the chain

n agree.

A1.4 Comparison

The comparison is the canonical digests of the two objects, recorded in full. 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 07:20–07:29Z | frozen method, Status line |
| Checker's statement run (route B source, before the freeze) | 2026-09-24T08:03:30Z | A8c record, ledger 3121915 |
| Method frozen | 2026-09-25
14:27:04.551949+00 | ledger 3147512 |
| First re-run under the frozen method | 2026-09-25T14:28:03Z | reruns.json (ledger 3147552) |
| Last re-run | 2026-09-25T14:28:18Z | reruns.json (ledger 3147552) |
| Re-run record banked | 2026-09-25
14:29:59.168369+00 | ledger 3147552 |
| Amendment A1 frozen | 2026-09-26
02:56:24.659289+00 | ledger 3158893 |
| Comparison banked | 2026-09-26
02:57:07.412136+00 | ledger 3158904 |

PART 4 — POPULATION AND PIN

4.1 Population

From the frozen method (Part 3.2).

Population. Every receipt in glasshull.receipts, genesis to the pin: block 62,471,946, hash 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (the store's E0 tip, 2026-09-14T03:00:01Z). USER transactions = every receipt with type != 106; type = 106 is the ArbOS internal system transaction, reported separately and not counted as a user transaction.

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

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

| State ment | Tables the statement reads | Coverage recorded on its run record (table min..max) | Coverage read at (UTC) |
|------------|----------------------------|---|------------------------|
| P2 | receipts | receipts 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| P3 | receipts | receipts 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| P4 | receipts | receipts 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| P5 | receipts | receipts 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| P6 | receipts | receipts 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| R1 | fees_by_block | blocks 0..62471946, 627 partitions; transactions 0..62471946, 627 partitions; receipts 0..62471946, 627 partitions; logs 0..62471946, 627 partitions; transfers 0..62471946, 627 partitions | 2026-09-25T14:27:48Z |
| R2 | fees_by_block, receipts | receipts 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 The pin

| | |
|---------------------------------|--|
| Block | 62,471,946 |
| Hash | 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Block time (UTC) | 2026-09-14T03:00:01Z |
| Height and head at every re-run | 62471946 / 62471946 (all 8 run records agree) |
| Block hash on every run record | 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb (all 8 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.fees_by_block, glasshull.receipts |
| System transaction type (ArbOS) | 106 |
| Sample block for the fee check (R2) | 40000000 |
| Last block holding a failed receipt (P5 last_blk) | 62471935 |
| Block ranges of P1–P5 | P1 0 .. 19,999,999; P2 20,000,000 .. 34,999,999; P3 35,000,000 .. 43,999,999; P4 44,000,000 .. 52,999,999; P5 53,000,000 .. the pin |
| Contracts and addresses | None; the measurement counts receipts by status and type. |

5.2 Documents

| Document | DCN | Ledger event | SHA-256 |
|---|-------------------------|---------------------------------|---|
| Proof Posit PP-006 | JFD-20260926-PP-00002 | kind-15 3171918 | 4005f0a42b1de3374e4564784c688200fb1265475ea4e2220a40a68a89b9152d |
| Proof Posit PP-006, source | — | kind-15 3171918 (source_sha256) | 17f940f50bfe3fd3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0 |
| Full Proof FP-006, version 1 | JFD-20260926-FP-00002 | kind-15 3173079 | 855359021b1d998536306042fd7f8b88c5687539d572baf12161c89a7540073f |
| Full Proof FP-006, version 1, full text | — | printed on the v1 cover | 52989de19107e56ce8721e20f10098ff48aba5f25ebb730982ef5cf870752f69 |
| Certificate of version 1 | JFD-20260926-CERT-00005 | kind-15 3173086 | 810553131fdc2ce1d002b9a0be06fc05d4f4b996d492176fe8dbb9a831518c8f |

5.3 Ledger events cited

| Event | Kind | What it records | Created (UTC) | Event hash |
|---------|------|--------------------|-------------------------------|--|
| 3121915 | 3 | second route (A8c) | 2026-09-24 09:04:38.691151+00 | 2bed54f042d5cc91e200c9f70411898cd31a6de8b63ca696175040f4f6c11b16 |
| 3147512 | 3 | method freeze | 2026-09-25 14:27:04.551949+00 | 3838f10faee17d63d314c71708d9934d42c47830c8c138209d26b5d9917a0564 |
| 3147552 | 3 | re-run record | 2026-09-25 14:29:59.168369+00 | 69ebab908ab50df1d5ab84815664934dee0e4f6b61a3db56cf5978702f0bf79b |
| 3147553 | 3 | user/system split | 2026-09-25 14:30:00.022387+00 | 8471a2363c034da0446edf4317153dbddd1aa8d7dc21d7b72b90884c5e667d04 |

| Event | Kind | What it records | Created (UTC) | Event hash |
|---------|------|---|-------------------------------|--|
| 3147767 | 3 | correction to 3147552 (what the re-run hash covers) | 2026-09-25 14:43:43.235442+00 | da8a59ecc0b18c55948b11e9849b33ed1eb96ec270de4dbf14ed408d5c7abb1c |
| 3158893 | 3 | amendment A1 freeze | 2026-09-26 02:56:24.659289+00 | 67bdf17c9bf40c30bc052c881c07374baed7d6d606214d3ce4e81ade527c0f49 |
| 3158904 | 3 | maker/checker comparison | 2026-09-26 02:57:07.412136+00 | 5a407ce17abfa19df724b31c295815fff8f63117a67e26f14042dd6e26e8db94 |
| 3171918 | 15 | Posit seal | 2026-09-26 17:09:16.286282+00 | 56a3050369ccfd9ea479c3d3cd23541e3a82549256d52db684dda51b1fa9a7b2 |
| 3173079 | 15 | Full Proof v1 seal | 2026-09-26 18:21:39.860285+00 | ca8854b33ec0c6b8e9542660b057bdef7498ce97e5f4525a658e34864f3aa3d6 |
| 3173086 | 15 | v1 certificate seal | 2026-09-26 18:22:00.229345+00 | 08ae68c5e2bff83afa901af54545453d3f8c0c6aee0b014fe5664ca98ff1349d |

5.4 Record files

| Record | File | SHA-256 | Banked in |
|---|---|--|-----------|
| sealed Proof Posit source (seal event source_sha256) | PP-006-v2-ASSEMBLED.md | 17f940f50bfefd3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0 | — |
| sealed Full Proof v1 full text (the source fingerprint printed on the v1 cover; attachment of the v1 certificate) | FP-006-free-full-proof.md | 52989de19107e56ce8721e20f10098ff48aba5f25ebb730982ef5cf870752f69 | — |
| frozen method | PP-006-METHOD-frozen.md | f31bcd7662a6be6f2bd64c96db297e343c7c9ca0b0d29edcf0b7f27e181631a1 | 3147512 |
| method amendment A1 | PP-006-METHOD-amendment-A1-output-object.md | b6cf1c7f7006f2a80db22c1de87beaec1603644c8bbde7e9f59afc0d52740984 | 3158893 |
| re-run record (statements, rows, result SHA-256) | reruns.json | baf200c4304443b0cd26244b97f8fb880d3f1bb4c90ebb433c56f05ed82847fe | 3147552 |
| maker arm object (route A) | PP006-maker-v1.json | f46df0b5da75a70709745ba818d7f396913bc5b23dda80a8785dd63dc5d254d0 | 3158904 |
| checker arm object (route B) | PP006-checker-v1.json | f46df0b5da75a70709745ba818d7f396913bc5b23dda80a8785dd63dc5d254d0 | 3158904 |
| maker/checker comparison | PP006-comparison-v1.json | f923a61e44cafc8cbc45a8f8369ee6576088dc5814bbaef54725d0746b76292c | 3158904 |
| Appendix R as attached to the v1 certificate | FP-006-appendix-R.md | 1a2a35e6f745f2d1b15f2c79ec79f69add81fcfb5a7e958b375578693074875c | — |

| Record | File | SHA-256 | Bank ed in |
|--|--|--|------------|
| Appendix S as attached to the v1 certificate | FP-006-appendix-S.md | 53f3b12900d819268c678206a6e6a244298086c43a3b3d0d2985d567ba
dd18f6 | — |
| checker source statement (route B) | laundry-a8c-failed-fee-whole-window.json | 79e5d3974ec80f9d23fd16005918c6b4eb192f1fa0e8afa701e68aa0964f8bc7 | 3121915 |

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

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 §S.3, copied from the statement's run record. Reproducibility class for every statement: needs a full node or a complete extraction.

P1 — PP006_r1 (pass arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:03Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 3a8a1838fdb4a0e5ffd0bb491b596a50234998944307004bbf4637edca122003 |
| Columns (types) | failed (UInt64), fee_failed_wei (String) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 8f3556308c71318fc34e7cddc8a8ccb951292ceeb702631365dab8a7aaf122d1 |
| Run record file | runs\3a8a1838fdb4a0e5ffd0bb491b596a50234998944307004bbf4637edca122003.json, SHA-256 df8490eedfa84f77983609d51accbe18645119d3f4a8b69dd67c5a6d1d97be3f |
| Result file | results\3a8a1838fdb4a0e5ffd0bb491b596a50234998944307004bbf4637edca122003.json, SHA-256 8f3556308c71318fc34e7cddc8a8ccb951292ceeb702631365dab8a7aaf122d1 (equal to the result SHA-256) |

```
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number < 20000000
```

P2 — PP006_r2 (pass arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:06Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 9bb7065fc9a3151dd8d0baa4994a9b236858d451a1638aaf0ba1338bb5a2d1e6 |
| Columns (types) | failed (UInt64), fee_failed_wei (String) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 06d2f396fe4a6c057b55ebf59a0b8d896f49407ccf2f1f2b7901438941387c01 |
| Run record file | runs\9bb7065fc9a3151dd8d0baa4994a9b236858d451a1638aaf0ba1338bb5a2d1e6.json, SHA-256 e7fe7612e6962fc407efa4d29534c825184f0eb881c421689e188c4bf0b478eb |
| Result file | results\9bb7065fc9a3151dd8d0baa4994a9b236858d451a1638aaf0ba1338bb5a2d1e6.json, SHA-256 06d2f396fe4a6c057b55ebf59a0b8d896f49407ccf2f1f2b7901438941387c01 (equal to the result SHA-256) |

```
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number >= 20000000 AND block_number < 35000000
```

P3 — PP006_r3 (pass arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:11Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | c20040b9050994bc02dc6ada4a5ca6244d22f63694d3647824a8a7f7d4ffe705 |
| Columns (types) | failed (UInt64), fee_failed_wei (String) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 85837330e3332835766ad030644dda6a0d88690d1dae0d49c0cd8b2f254a001 |

| | |
|-----------------|--|
| Run record file | runs\c20040b9050994bc02dc6ada4a5ca6244d22f63694d3647824a8a7f7d4ffe705.json, SHA-256 b38546bfeedc24456ec106f379c50f5358d7599914797cb92b8713ea62bb6bdb |
| Result file | results\c20040b9050994bc02dc6ada4a5ca6244d22f63694d3647824a8a7f7d4ffe705.json, SHA-256 85837330e3332835766ad030644dda6a0d88690d1dae0d49c0cd8b2f254a001 (equal to the result SHA-256) |

```
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number >= 3500000
0 AND block_number < 44000000
```

P4 — PP006_r4 (pass arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:12Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | afe1cfc145f1b79a7ebcfee8243de57bd3ed8a79ae320e09260cca150a66db98 |
| Columns (types) | failed (UInt64), fee_failed_wei (String) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 0f5e069714de71138eb8c6e82c91ec0ea027244d56ee89c88f0ed107bf32e31d |
| Run record file | runs\afe1cfc145f1b79a7ebcfee8243de57bd3ed8a79ae320e09260cca150a66db98.json, SHA-256 426ed9d5a32594bfd9b9bdecc0e51aee1638049711319bcdd1d8c86ea361cda5 |
| Result file | results\afe1cfc145f1b79a7ebcfee8243de57bd3ed8a79ae320e09260cca150a66db98.json, SHA-256 0f5e069714de71138eb8c6e82c91ec0ea027244d56ee89c88f0ed107bf32e31d (equal to the result SHA-256) |

```
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei FROM glasshull.receipts WHERE status = 0 AND block_number >= 4400000
0 AND block_number < 53000000
```

P5 — PP006_r5 (pass arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:14Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 669cfdcf23a9c7418bec2fd84a70f28f0434e192fb171fc7c56d70373c589c74e |

| | |
|-----------------|--|
| Columns (types) | failed (UInt64), fee_failed_wei (String), last_blk (UInt64) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 6d0b2d508e1c1a4f70df701bba2d2d81438a4d63bb20a2a1557175548e3f8cdc |
| Run record file | runs\669cfd23a9c7418bec2fd84a70f28f0434e192fb171fc7c56d70373c589c74e.json, SHA-256 0c876b04d7da3c27ed08c965551cc905958a7104fefb360484b8eaf2af60683c |
| Result file | results\669cfd23a9c7418bec2fd84a70f28f0434e192fb171fc7c56d70373c589c74e.json, SHA-256 6d0b2d508e1c1a4f70df701bba2d2d81438a4d63bb20a2a1557175548e3f8cdc (equal to the result SHA-256) |

```
SELECT count() AS failed, toString(sum(toUInt128(gas_used) * reinterpretAsUInt128(reverse(effective_gas_price)))) AS fee_failed_wei, max(block_number) AS last_blk FROM glasshull.receipts WHERE status = 0 AND block_number >= 53000000
```

P6 – PP006_user (pass arm)

| | |
|-------------------|--|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:18Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 91c1eaf41047134087c1ef1d434e926197df8c5e27c2d1e54f3e1561333fdb3 |
| Columns (types) | user_txs (UInt64), user_failed (UInt64), system_txs (UInt64), system_failed (UInt64), system_gas_used (UInt64) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 8d0259ad70b0ca1bf09c5b129e69a3cfb2518a4450dc618bf87de6efd5a31126 |
| Run record file | runs\91c1eaf41047134087c1ef1d434e926197df8c5e27c2d1e54f3e1561333fdb3.json, SHA-256 89e4c0456a66265630069a2dc863536a9f9a16ad3fc9414d550068921f7d7f0e |
| Result file | results\91c1eaf41047134087c1ef1d434e926197df8c5e27c2d1e54f3e1561333fdb3.json, SHA-256 8d0259ad70b0ca1bf09c5b129e69a3cfb2518a4450dc618bf87de6efd5a31126 (equal to the result SHA-256) |

```
SELECT countIf(type != 106) AS user_txs, countIf(type != 106 AND status = 0) AS user_failed, countIf(type = 106) AS system_txs, countIf(type = 106 AND status = 0) AS system_failed, sumIf(gas_used, type = 106) AS system_gas_used FROM glasshull.receipts
```

R1 — PP006_refute (refute arm)

| | |
|-------------------|---|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:16Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 02dd95e4d55ef659ec1d49d5c06a788993ef3899a983d43d959b00278edf84c3 |
| Columns (types) | blocks (UInt64), first_blk (UInt64), last_blk (UInt64), txs_all (UInt64), txs_failed (Int64) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | 6b2e2eaec50f8649facf07b3d00e62a167efb52404b9072795de2b58d6891e34 |
| Run record file | runs\02dd95e4d55ef659ec1d49d5c06a788993ef3899a983d43d959b00278edf84c3.json, SHA-256 06a48dd9828982f5fa82c15877b01b07c6b4ef713931fe73fcaa6675cd830b85 |
| Result file | results\02dd95e4d55ef659ec1d49d5c06a788993ef3899a983d43d959b00278edf84c3.json, SHA-256 6b2e2eaec50f8649facf07b3d00e62a167efb52404b9072795de2b58d6891e34 (equal to the result SHA-256) |

```
SELECT count() AS blocks, min(number) AS first_blk, max(number) AS last_blk, sum(txs) AS txs_all, sum(txs) - sum(txs_ok) AS txs_failed FROM glasshull.fees_by_block
```

R2 — PP006_feesample (refute arm)

| | |
|-------------------|--|
| Engine | clickhouse (store surface) |
| Run at (UTC) | 2026-09-25T14:28:18Z |
| Height · pin hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Re-run hash | 4c21c093f48db57e02203c9d781cafabdfc455d96bf14cd3f8cba188e5dbd8b5 |
| Columns (types) | receipts_fee_wei (Nullable(String)), fees_by_block_12_plus_11 (Nullable(String)), 12_only (Nullable(String)) |
| Row count | 1 |
| Truncated | false |
| Result SHA-256 | ecce8cd4e0de481e92c908bd75f65130099034fd98e84011ca0f15239dc784dc |
| Run record file | runs\4c21c093f48db57e02203c9d781cafabdfc455d96bf14cd3f8cba188e5dbd8b5.json, SHA-256 0c75c6816912439a591efe9629b635d8562e8bfdd1753380bc257dc9f74c38c9 |

| | |
|-------------|---|
| Result file | results\4c21c093f48db57e02203c9d781cafabdfc455d96bf14cd3f8cba188e5dbd8b5.json, SHA-256 ecce8cd4e0de481e92c908bd75f65130099034fd98e84011ca0f15239dc784dc (equal to the result SHA-256) |
|-------------|---|

```
SELECT (SELECT toString(sum(toUInt256(gas_used) * reinterpretAsUInt256(reverse(effective_gas_price)))) FROM glasshull.receipts WHERE block_number = 40000000) AS receipts_fee_wei, (SELECT toString(toUInt256(l2_fee_wei_dec) + toUInt256(l1_fee_wei_dec)) FROM glasshull.fees_by_block WHERE number = 40000000) AS fees_by_block_l2_plus_l1, (SELECT l2_fee_wei_dec FROM glasshull.fees_by_block WHERE number = 40000000) AS l2_only
```

Route B source — the checker's statement (A8c, failed_fee_whole_window_tip)

A different statement over the same receipts table, run before the method was frozen and banked as kind-3 3121915. Amendment A1.3 (Part 3.3) states how the checker derives its figures from it and from R1.

| | |
|------------------------------|---|
| Engine | A1 console ClickHouse (glasshull, full window 0..62471946) |
| Run at (UTC) | 2026-09-24T08:03:30Z |
| Tip block · hash | 62471946 · 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb |
| Row count | 2 |
| Result SHA-256 (as recorded) | 1fda1ebcad916aadb30b4c7edfebee47ecdb4ef5a12b20c1ae50e3d7182c0214 |

```
SELECT status, count() AS txs, toString(sum(toUInt256(gas_used) * reinterpretAsUInt256(reverse(effective_gas_price)))) AS fee_wei FROM glasshull.receipts WHERE type != 106 GROUP BY status ORDER BY status
```

| status | txs | fee_wei |
|--------|-----------|-------------------------|
| 0 | 119984049 | 1551080321319255460000 |
| 1 | 533024373 | 18251209359430093266000 |

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) | the A8c statement plus the per-block tally R1 |
| Object file | PP006-maker-v1.json | PP006-checker-v1.json |
| File SHA-256 | f46df0b5da75a70709745ba818d7f396913bc5b23dda80a8785dd63dc5d254d0 | f46df0b5da75a70709745ba818d7f396913bc5b23dda80a8785dd63dc5d254d0 |
| Canonical SHA-256 | f51620cc77266453787dead9bca9f390d9d843afc2891848ff6c4e6682e121ea | f51620cc77266453787dead9bca9f390d9d843afc2891848ff6c4e6682e121ea |

Verdict: CLEARED - both arms emit the identical canonical object. Differing keys: none. Label: statement-independent, lineage-shared (both routes read the same E0 extraction on the A1 store). Banked as kind-3 3158904.

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

7.1 The five figures

| Field | Route A | Route B | Difference |
|---------------------|--|--|------------|
| pin.block | 62471946 | 62471946 | 0 |
| pin.hash | 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb | 0x6be6970fbf6ebda579ae87bad4df04a3b7dd14486a40508a747d079fbef462eb | 0 |
| user_txs | 653008422 | 653008422 | 0 |
| user_failed | 119984049 | 119984049 | 0 |
| user_failed_fee_wei | 1551080321319255460000 | 1551080321319255460000 | 0 |
| system_txs | 62710451 | 62710451 | 0 |
| system_failed | 0 | 0 | 0 |

7.2 How each route reaches its figures

| Field | Route A (maker): from | Route B (checker): from |
|---------------------|--|---|
| user_txs | P6 user_txs = 653008422 | A8c status 0 + status 1 txs = 119984049 + 533024373 = 653008422 |
| user_failed | P6 user_failed = 119984049 | A8c status 0 txs = 119984049 |
| user_failed_fee_wei | sum of P1–P5 fee_failed_wei = 212060710844437370000 + 178878750922993954000 + 45585008478580212000 + 356169485660238432000 + 758386365413005492000 = 1551080321319255460000 (P6 system_failed = 0) | A8c status 0 fee_wei = 1551080321319255460000 |
| system_txs | P6 system_txs = 62710451 | R1 txs_all – user_txs = 715718873 – 653008422 = 62710451 |
| system_failed | P6 system_failed = 0 | R1 txs_failed – user_failed = 119984049 – 119984049 = 0 |

PART 8 — THE REFUTE ARM

Quoted verbatim from the sealed Proof Posit PP-006, section III.i, Refute arm. Source SHA-256

17f940f50bfe9d3fc756fc4ef6de19aaf27ad49638928b0690fe6a32813adcd0, the source recorded by seal event kind-15 3171918 for JFD-20260926-PP-00002.

Refute arm. An independent tally of every transaction in every block, minus the successful ones, gives **119,984,049** failed: identical to the pass arm. The fee computation was also checked at a sample block against the chain's recorded per-block fees, and matched exactly.

R1 — PP006_refute

| blocks | first_blk | last_blk | txs_all | txs_failed |
|----------|-----------|----------|-----------|------------|
| 62471947 | 0 | 62471946 | 715718873 | 119984049 |

Re-run hash 02dd95e4d55ef659ec1d49d5c06a788993ef3899a983d43d959b00278edf84c3 · result SHA-256 6b2e2eaec50f8649facf07b3d00e62a167efb52404b9072795de2b58d6891e34.

R2 — PP006_feesample

| receipts_fee_wei | fees_by_block_l2_plus_l1 | l2_only |
|------------------|--------------------------|----------------|
| 18142822886000 | 18142822886000 | 18142822886000 |

Re-run hash 4c21c093f48db57e02203c9d781cafabdfc455d96bf14cd3f8cba188e5dbd8b5 · result SHA-256 ecce8cd4e0de481e92c908bd75f65130099034fd98e84011ca0f15239dc784dc.

The agreement rules, and how they came out

The rules, from the frozen method (Part 3.2):

- sum of P1..P5 failed == R1 txs_failed == P6 user_failed + P6 system_failed.
- R2: receipts_fee_wei == fees_by_block_l2_plus_l1.
- System transactions: P6 system_failed and system_gas_used are reported; if system_gas_used = 0 the fee total is unaffected by the user/system split.
- Second route (independent, already banked): kind-3 3121915 (A8c), population type != 106, must equal the failed count and the failed-fee total to the wei.

Their results, from the re-run record (reruns.json, pp006_agreement, banked in 3147552):

| Check | Value |
|----------------------------|------------------------|
| sum_P1_P5_failed | 119984049 |
| sum_P1_P5_failed_fee_wei | 1551080321319255460000 |
| R1_txs_failed | 119984049 |
| P6_user_failed | 119984049 |
| P6_system_failed | 0 |
| failed_equal | true |
| R2_equal | true |
| system_gas_used | 0 |
| second_route_3121915_equal | true |
| user_txs | 653008422 |

| Check | Value |
|-----------------------------|--------------------|
| system_txs | 62710451 |
| failure_rate_user_fraction | 39994683/217669474 |
| failure_rate_user_pct_label | 18.374 |

PART 9 — THE MEASURED TABLES

9.1 Failed transactions and the fees they paid, by block range (P1–P5)

| Statement | Blocks | failed | fee_failed_wei |
|-----------|--------------------------|-----------|------------------------|
| P1 | 0 .. 19,999,999 | 32594667 | 212060710844437370000 |
| P2 | 20,000,000 .. 34,999,999 | 38719737 | 178878750922993954000 |
| P3 | 35,000,000 .. 43,999,999 | 13524416 | 45585008478580212000 |
| P4 | 44,000,000 .. 52,999,999 | 16117253 | 356169485660238432000 |
| P5 | 53,000,000 .. the pin | 19027976 | 758386365413005492000 |
| Sum | 0 .. the pin | 119984049 | 1551080321319255460000 |

9.2 User and system transactions (P6)

| user_txs | user_failed | system_txs | system_failed | system_gas_used |
|-----------|-------------|------------|---------------|-----------------|
| 653008422 | 119984049 | 62710451 | 0 | 0 |

9.3 The failure rate

| | |
|-------------------------------|--------------------|
| user_failed / user_txs, exact | 39994683/217669474 |
| Rounded for display (%) | 18.374 |

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 | PP006_r1 | pas s | 2026-09-25T14:28:03Z | 3a8a1838fdb4a0e5ffd0bb491b596a50234998944307004bbf4637edca12203 | 8f3556308c71318fc34e7cddc8a8ccb951292ceeb702631365dab8a7aaf122d1 |
| P2 | PP006_r2 | pas s | 2026-09-25T14:28:06Z | 9bb7065fc9a3151dd8d0baa4994a9b236858d451a1638aaf0ba1338bb5a2d1e6 | 06d2f396fe4a6c057b55ebf59a0b8d896f49407ccf2f1f2b7901438941387c01 |

| State ment | Name | Ar m | Run at (UTC) | Re-run hash (the question) | Result SHA-256 (the answer) |
|------------|------------------|---------|----------------------|---|---|
| P3 | PP006_r3 | pas s | 2026-09-25T14:28:11Z | c20040b9050994bc02dc6ada4a5ca6244d22f63694d3647824a8a7f7d4ffe705 | 85837330e3332835766ad030644dda6a0d88690d1daef0d49c0cd8b2f254a001 |
| P4 | PP006_r4 | pas s | 2026-09-25T14:28:12Z | afe1cfc145f1b79a7ebcfee8243de57bd3ed8a79ae320e09260cca150a66db98 | 0f5e069714de71138eb8c6e82c91ec0ea027244d56ee89c88f0ed107bf32e31d |
| P5 | PP006_r5 | pas s | 2026-09-25T14:28:14Z | 669cfdcf23a9c7418bec2fd84a70f28f0434e192fb171fc7c56d70373c589c74e | 6d0b2d508e1c1a4f70df701bba2d2d81438a4d63bb20a2a1557175548e3f8cdc |
| P6 | PP006_u ser | pas s | 2026-09-25T14:28:18Z | 91c1eaf41047134087c1ef1d434e926197df8c5e27c2d1e54f3e1561333fdb a3 | 8d0259ad70b0ca1bf09c5b129e69a3c fb2518a4450dc618bf87de6efd5a31126 |
| R1 | PP006_r efute | ref ute | 2026-09-25T14:28:16Z | 02dd95e4d55ef659ec1d49d5c06a788993ef3899a983d43d959b00278edf84c3 | 6b2e2eaec50f8649facf07b3d00e62a167efb52404b9072795de2b58d6891e34 |
| R2 | PP006_fe esample | ref ute | 2026-09-25T14:28:18Z | 4c21c093f48db57e02203c9d781cafa bdfc455d96bf14cd3f8cba188e5dbd8b5 | ecce8cd4e0de481e92c908bd75f65130099034fd98e84011ca0f15239dc784dc |

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 (§S.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

- 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.
- The method section of this Full Proof (§2) and its queries (§3), used verbatim.** Paraphrasing a query is how results drift.
- Integer arithmetic of at least 128 bits, 256 preferred,** in your query engine or language.

R.2 PIN BEFORE YOU RUN

- 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-00002.pdf` - 455576 bytes, SHA-256 `a58603fe7366c4025f8b6cd58f1aa922375b04dc532c3b1b46c59d4817e60495`.

Its exact source, whose fingerprint the document prints, `FP-006-v2-source.md` (56858 bytes, SHA-256 `6e2ef2d8b20d6cd2976a56b00e53d3428da57ab75a5683ad74f26a904a6db96e`) is embedded in this PDF as a file attachment.

The sealed document `JFD-20260928-FP-00002.pdf` (455576 bytes, SHA-256 `a58603fe7366c4025f8b6cd58f1aa922375b04dc532c3b1b46c59d4817e60495`) is embedded in this PDF as a file attachment.

The anchor proof as stamped at sealing `JFD-20260928-FP-00002.pdf.ots` (786 bytes, SHA-256 `e83aa03a623284d322941b1a0eebe3e5ffd8666a14ecc98f1b0866ae16b6e66`) 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 seal record from which the certificate is built `seal-record-JFD-20260928-FP-00002.json` (7255 bytes, SHA-256 `64266b8fc352ba1b44238ac69034b0f0f49d4a393cc331eacc48bcb26a668bc5`) is embedded in this PDF as a file attachment.

The complete anchor proof, carrying the Bitcoin attestation, `JFD-20260928-FP-00002.pdf.anchored.ots` (2580 bytes, SHA-256 `dbe3182b5bf948760a441a2636308ae44e4be4d3c0cc362c9b320b16cb8f8783`) 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-00002.json` (2582 bytes, SHA-256 `2bb3e14fc1f6166a369189d41090d6b498d715be8271093dcf3855528a5a62cf`) 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 of Authenticity on the Turing plate, built from the same seal record and signed; then the page "How to verify this without the Issuer, in six steps", which carries the record, both witnesses and the anchor in full, and the steps that verify all of it without the Issuer.

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 3205882 | → ledger event 3205882 |
| signed (UTC) | → event 3205882 | → event 3205882 |
| 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 3205882 | → event 3205882 |

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 3205882 · <https://gh.jetfyul.com/v/JFD-20260928-FP-00002>

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-00002>

GLASS HULL

CERTIFICATE OF AUTHENTICITY · JFD-20260928-FP-00002

JFD-20260928-FP-00002.pdf · 455576 bytes · sealed 2026-09-28T05:44:15Z

Robinhood Chain failed transactions

Certificate JFD-20260928-CERT-00012

a58603fe7366c4025f8b6cd58f1aa922375b04dc532c3b1b46c59d4817e60495

Ledger event 3205882 (kind-15)

A9 0123ADA0711D8597EE · A10 0123DC797ACC1195EE

Bitcoin block 968952 · 2026-09-28T05:46:19Z

a58603fe7366c402 · ANCHORED · <https://gh.jetfyul.com/v/JFD-20260928-FP-00002>

K. L. Phillips
Chief Executive Officer

Glass Hull · FP-Template 01-926 · Alan Turing
Matchstick Made © 2026 · Jet Fyul Dynamics LLC

How to verify this without the Issuer, in six steps

Robinhood Chain failed transactions - JFD-20260928-FP-00002

The record

Document control number: JFD-20260928-FP-00002

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

a58603fe7366c4025f8b6cd58f1aa922375b04dc532c3b1b46c59d4817e60495

Sealed file: JFD-20260928-FP-00002.pdf, 455576 bytes

Source fingerprint printed inside the document:

6e2ef2d8b20d6cd2976a56b00e53d3428da57ab75a5683ad74f26a904a6db96e

Ledger event (kind-15): 3205882, event hash

ce8dbcc9830883eb533bcccb93d400e9fb49c9a23e84c054ad334444d71b67c4

Notary authorisation (kind-3): 3205881

Seal request signed by both witnesses: SHA-256

bfd14ce95723e96f58178ab4389ab1e676b73e16e1cf88f450960d4f53f3cf3d

Anchor: OpenTimestamps, **ATTESTED in Bitcoin block 968952** - block hash

00000000000000000000d994a468367aa49a494af15e29737ac5c1b788f02a8f, block time 2026-09-28T05:46:19Z.

Calendars that attested: <https://finney.calendar.eternitywall.com> (block 968952);

<https://alice.btc.calendar.opentimestamps.org> (block 968953). The earliest attested block is the anchor.

Anchor check: the committed value equals block 968952's merkle root

ecfeaa519852da74d3b675ab34ca4df4645a0f929510338682bba6968e3d0b0b, and mempool.space and blockstream.info agree on that merkle root and on the block hash (checked 2026-09-28T06:51:29Z).

Proofs: the proof as stamped at sealing, JFD-20260928-FP-00002.pdf.ots and the complete proof JFD-20260928-FP-00002.pdf.anchored.ots, which carries the Bitcoin attestation itself.

Verification address: <https://gh.jetfyul.com/v/JFD-20260928-FP-00002>

The two hardware witnesses

Witness 1 - host A9, a9-atecc608b-slot0, serial 0123ADA0711D8597EE, curve ECDSA-P256 (secp256r1), signed 2026-09-28T05:44:12.718569+00:00

Public key: d62d03f8dc3813fb5b74f2b64339d185742aabcae1bdcfd2f7a7ad59f1dd3c5d50e57effa4a3c888f58c3eca9d274bd7f0b8d263fa38bbf0bc6fbff6035c2411

Signature: 84baf89c3e2cbf218d7b236cac700a7725b86f2030c58cb3cfb2ba240cef223bcd6dc272bd6e41fff d38af39bc728fe37aa8a5ebefcbf8c0baa33bb1095abc5a

Witness 2 - host A10, a10-atecc608b-slot0, serial 0123DC797ACC1195EE, curve ECDSA-P256 (secp256r1), signed 2026-09-28T05:44:14.986868+00:00

Public key: 853e69fc700990303abfe61358bfa456160eaae1529dbc74805d1639cfc88e78497bbe060e7ae5d54466b2ce0a2753e14204a21eed4a132a61ace3294f9beba1

Signature: 0fd836e1021de641b9701f1d62644f25aec84146436405191498787d255db9de3ef596ecbd4c39f846619071ea6daaba044981c4be6be8a7b11b988b1606437f

The six steps

- (1) Extract JFD-20260928-FP-00002.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 3205882 by its id. Its payload must carry that fingerprint as doc_sha256 and JFD-20260928-FP-00002 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 bfd14ce95723e96f58178ab4389ab1e676b73e16e1cf88f450960d4f53f3cf3d.
- (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 968952; 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-00002> and confirm it shows the same fingerprint, ledger event, witnesses and anchor.

This certificate supersedes an earlier one

This certificate supersedes certificate **JFD-20260928-CERT-00003** (ledger event 3206933, SHA-256 92951d88434e1606bff5fd4f3346e71c28d99056b5a492acde5249a4aff72792). That certificate was issued before the certificate design of record was applied. Nothing about the certified document changed: the same file, the same fingerprint, the same sealing event and the same two witness signatures. The earlier certificate remains a valid seal of its own bytes; it is no longer the certificate of record.

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.