Milestone 2 Documentation
Final Version

AI.COREBLOCK zkTelemetry Proof Submissions via Kurier (zkVerify)

Author: Serkan Sahin (Development)
Project: AI.COREBLOCK - zkTelemetry → Kurier → zkVerify Mainnet
Document: Milestone 2 Technical Documentation
Date: 2026-02-08

1. Overview

Milestone 2 delivers a reliable, high-throughput proof submission pipeline from AI.COREBLOCK's zkTelemetry system to **zkVerify mainnet** via the **Kurier** submission layer. The milestone focuses on two core requirements:

1. **Durable job persistence**: Every proof submission is recorded and can be resumed or reconciled without losing correlation to Kurier jobs or on-chain transactions.
2. **Traceable on-chain verification**: For each submitted proof, we persist the zkVerify **extrinsic hash** and a direct **explorer URL**, enabling auditability and independent verification.

This milestone also documents the practical engineering process: the pipeline did not work perfectly from the start; we iterated through failures and edge cases and hardened the worker until it became stable at scale.

2. Architecture & Data Flow

2.1 Components

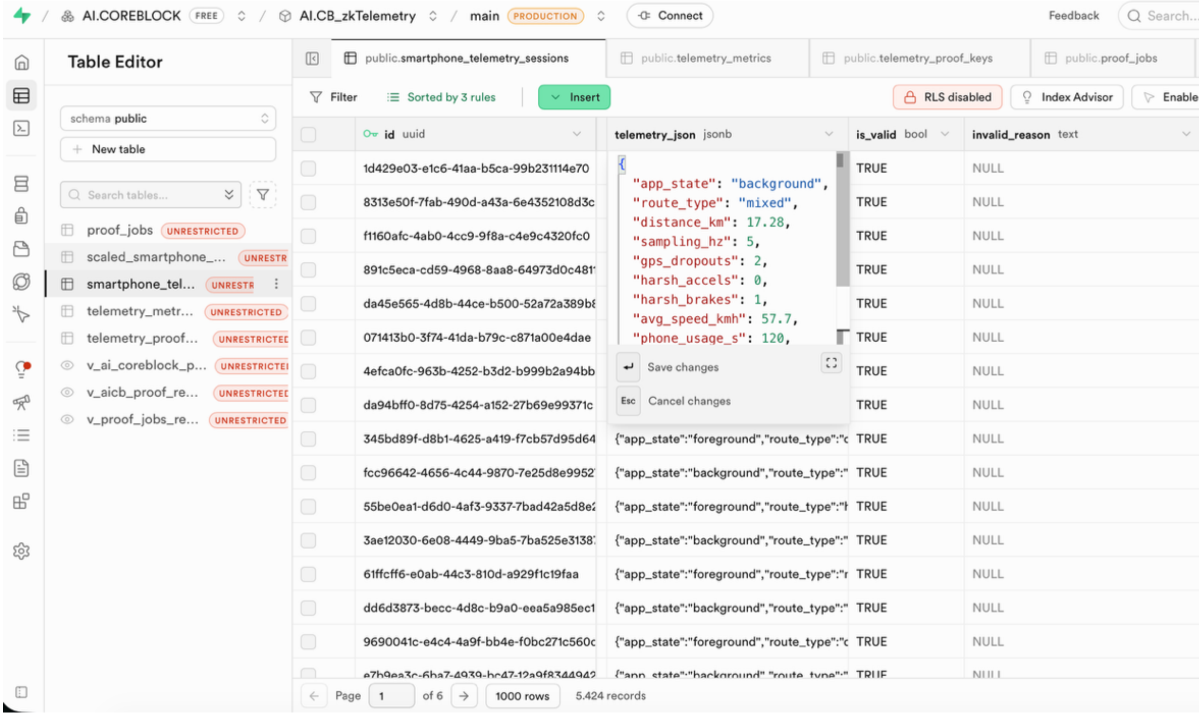
- **Supabase (PostgreSQL)**: stores telemetry sessions and proof job state.
- **Proof Submission Worker (Node.js / TypeScript)**: pulls queued proof jobs, submits them to Kurier, and persists results.
- **Kurier API**: job submission + job status tracking (finalization + tx hash).
- **zkVerify mainnet / Subscan explorer**: public on-chain record of proof submissions.

2.2 End-to-End Flow

1. Telemetry data is ingested from smartphones and stored in Supabase.
2. Proof jobs are created in Supabase with required proof inputs (e.g., commitment hash, proof data).
3. Worker pulls jobs in batches and submits them to Kurier.
4. Kurier returns a job identifier (jobId / requestId).
5. Once final, Kurier provides the zkVerify extrinsic hash and explorer link.
6. A sync script backfills tx hash / explorer URL for submitted jobs.

2.3 Smartphone Fleet in Vehicles

During Milestone 2 we operated a real-device pilot using **14 smartphones installed in vehicles** to collect telemetry under real driving conditions. This fleet was used to generate the input data that feeds the proof pipeline and to validate stable end-to-end execution at higher volume.



id uuid	telemetry_json jsonb	is_valid bool	invalid_reason text
1d429e03-e1c6-41aa-b5ca-99b231114e70		TRUE	NULL
8313e50f-7fab-490d-a43a-6e4352108d3c		TRUE	NULL
f1160afc-4ab0-4cc9-9f8a-c4e9c4320fc0		TRUE	NULL
891c5eca-cd59-4968-8aa8-64973d0c481		TRUE	NULL
da45e565-4d8b-44ce-b500-52a72a389b8		TRUE	NULL
071413b0-3f74-41da-b79c-c871a00e4dae		TRUE	NULL
4efca0fc-963b-4252-b3d2-b999b2a94bb		TRUE	NULL
da94bff0-8d75-4254-a152-27b69e99371c		TRUE	NULL
345bd89f-d8b1-4625-a419-f7cb57d95d64	{"app_state":"foreground","route_type":"c	TRUE	NULL
fcc96642-4656-4c44-9870-7e25d8e9952	{"app_state":"background","route_type":"	TRUE	NULL
55be0ea1-d6d0-4af3-9337-7bad42a5d8e	{"app_state":"foreground","route_type":"h	TRUE	NULL
3ae12030-6e08-4449-9ba5-7ba525e3138	{"app_state":"background","route_type":"	TRUE	NULL
61ffcff6-e0ab-44c3-810d-a929f1c19faa	{"app_state":"foreground","route_type":"r	TRUE	NULL
dd6d3873-becc-4d8c-b9a0-eea5a985ec1	{"app_state":"background","route_type":"	TRUE	NULL
9690041c-e4c4-4a9f-bb4e-f0bc271c560c	{"app_state":"foreground","route_type":"c	TRUE	NULL
e7b9ea3c-6ba7-4939-bc47-12a9f8344942	{"app_state":"background","route_type":"	TRUE	NULL

```
{
  "app_state": "background",
  "route_type": "mixed",
  "distance_km": 17.28,
  "sampling_hz": 5,
  "gps_dropouts": 2,
  "harsh_accels": 0,
  "harsh_brakes": 1,
  "avg_speed_kmh": 57.7,
  "phone_usage_s": 120,
```

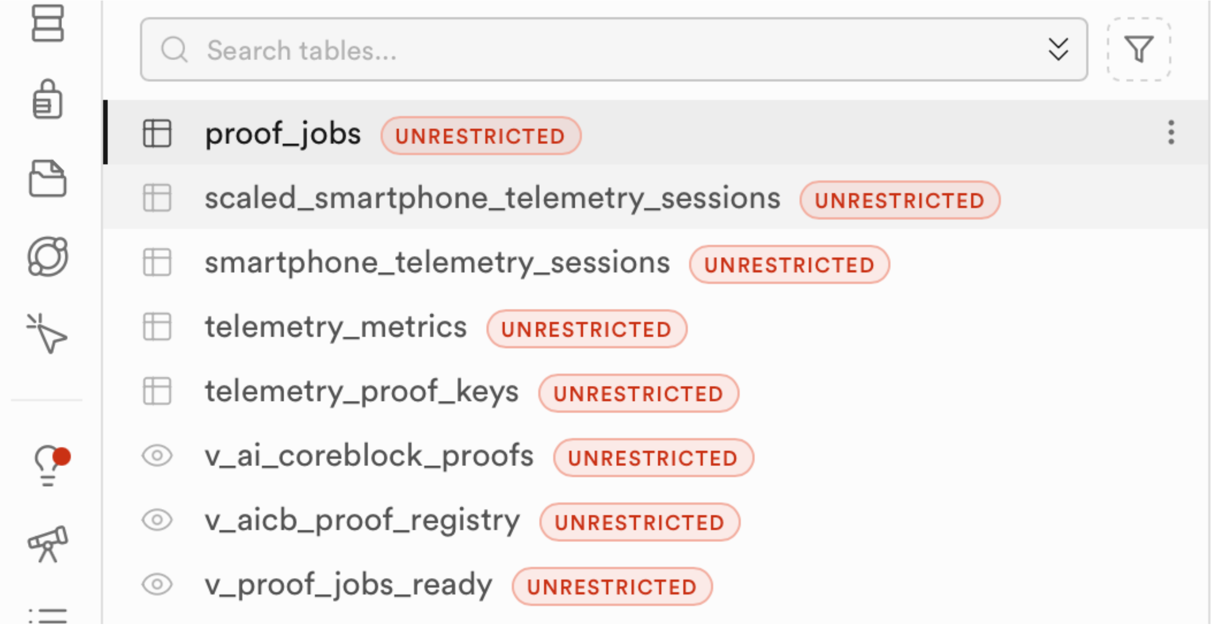
Figure 1. Example view of the telemetry dataset captured from the smartphone fleet (Supabase table view).

This setup ensured that the system was tested against a broader range of real-world conditions (e.g., different routes and driving patterns) while keeping the proof submission flow fully traceable from Supabase job records to on-chain zkVerify extrinsics.

3. Supabase Schema & Job State Model

3.1 Core Tables (Milestone 2 scope)

- smartphone_telemetry_sessions
- telemetry_metrics
- proof_jobs
- (optional views) v_proof_jobs_ready, v_ai_coreblock_proofs, etc.



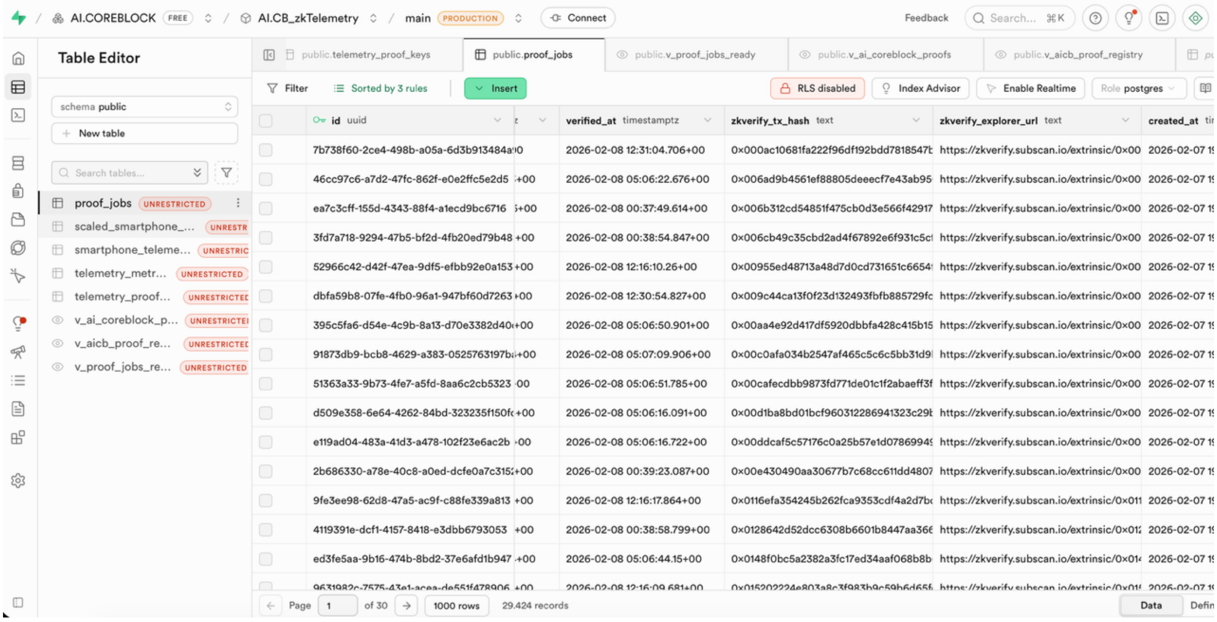
- proof_jobs UNRESTRICTED
- scaled_smartphone_telemetry_sessions UNRESTRICTED
- smartphone_telemetry_sessions UNRESTRICTED
- telemetry_metrics UNRESTRICTED
- telemetry_proof_keys UNRESTRICTED
- v_ai_coreblock_proofs UNRESTRICTED
- v_aicb_proof_registry UNRESTRICTED
- v_proof_jobs_ready UNRESTRICTED

Figure 2. Supabase schema overview showing telemetry and proof-related tables/views.

3.2 proof_jobs (Key Columns)

Each proof job is tracked with the following essential fields:

- status — lifecycle state (queued, submitting, submitted, verified, failed)
- attempts, last_error — resilience & debugging
- commitment_hash — unique proof reference (input-level identity)
- kurier_request_id — Kurier job identifier (critical for correlation)
- kurier_response — raw response payload snapshot for debugging/recovery
- zkverify_tx_hash — on-chain extrinsic hash
- zkverify_explorer_url — direct explorer link for verification
- created_at, updated_at, submitted_at, verified_at



AI.COREBLOCK FREE
AI.CB_zkTelemetry
main PRODUCTION
Connect
Feedback
Search... ⌘K
Table Editor
schema public
New table
Search tables...
proof_jobs UNRESTRICTED
scaled_smartphone_... UNRESTR
smartphone_teleme... UNRESTRIC
telemetry_metr... UNRESTRICTED
telemetry_proof... UNRESTRICTED
v_ai_coreblock_p... UNRESTRICTE
v_aicb_proof_re... UNRESTRICTE
v_proof_jobs_re... UNRESTRICTED
public.telemetry_proof_keys
public.proof_jobs
public.v_proof_jobs_ready
public.v_ai_coreblock_proofs
public.v_aicb_proof_registry
Filter
Sorted by 3 rules
Insert
RLS disabled
Index Advisor
Enable Realtime
Role postgres
Page 1 of 30
1000 rows
29.424 records
Data
Defin

id uuid	z	verified_at timestamptz	zkverify_tx_hash text	zkverify_explorer_url text	created_at ti
7b738f60-2ce4-498b-a05a-6d3b913484a	0	2026-02-08 12:31:04.706+00	0x000ac10681fa222f96df192bdd7818547b	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
46cc97c6-a7d2-47fc-862f-e0e2ffc5e2d5	+00	2026-02-08 05:06:22.676+00	0x006ad9b4561ef88805deeecf7e43ab95	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
ea7c3cff-155d-4343-88f4-a1ecd9bc6716	+00	2026-02-08 00:37:49.614+00	0x006b312cd54851f475cb0d3e566f42917	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
3fd7a718-9294-47b5-bf2d-4fb20ed79b48	+00	2026-02-08 00:38:54.847+00	0x006cb49c35cbd2ad4f67892e6f931c5c	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
52966c42-d42f-47ea-9df5-efbb92e0a153	+00	2026-02-08 12:16:10.26+00	0x00955ed48713a48d7d0cd731651c6654	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
dbfa59b8-07fe-4fb0-96a1-947bf60d7263	+00	2026-02-08 12:30:54.827+00	0x009c44ca13f0f23d132493fbfb885729fc	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
395c5fa6-d54e-4c9b-8a13-d70e3382d40	+00	2026-02-08 05:06:50.901+00	0x00aa4e92d417df5920dbbfa428c415b15	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
91873db9-bcb8-4629-a383-0525763197b	+00	2026-02-08 05:07:09.906+00	0x00c0afa034b2547af465c5c6c5bb31d9	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
51363a33-9b73-4fe7-a5fd-8aa6c2cb5323	00	2026-02-08 05:06:51.785+00	0x00cafecdbb9873fd771de01c1f2abaeff3f	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
d509e358-6e64-4262-84bd-323235f150f	+00	2026-02-08 05:06:16.091+00	0x00d1ba8bd01bcf960312286941323c29b	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
e119ad04-483a-41d3-a478-102f23e6ac2b	00	2026-02-08 05:06:16.722+00	0x00ddcaf5c57176c0a25b57e1d0786994	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
2b686330-a78e-40c8-a0ed-dcfe0a7c315	+00	2026-02-08 00:39:23.087+00	0x00e430490aa30677b7c68cc611dd4807	https://zkverify.subscan.io/extrinsic/0x00	2026-02-07 1
9fe3ee98-62d8-47a5-ac9f-c88fe339a813	+00	2026-02-08 12:16:17.864+00	0x0116efa354245b262fca9353cdf4a2d7b	https://zkverify.subscan.io/extrinsic/0x011	2026-02-07 1
4119391e-dcf1-4157-8418-e3dbb6793053	+00	2026-02-08 00:38:58.799+00	0x0128642d52dcc6308b6601b8447aa36	https://zkverify.subscan.io/extrinsic/0x01	2026-02-07 1
ed3fe5aa-9b16-474b-8bd2-37e6afd1b947	+00	2026-02-08 05:06:44.15+00	0x0148f0bc5a2382a3fc17ed34aaf068b8b	https://zkverify.subscan.io/extrinsic/0x01	2026-02-07 1
9631982c-7575-43e1-acea-de551f478906	+00	2026-02-08 12:16:09.681+00	0x015202224e803a8c3f983b9c59b6d65f	https://zkverify.subscan.io/extrinsic/0x01	2026-02-07 1

Figure 3. Supabase proof_jobs table showing persisted zkverify_tx_hash and zkverify_explorer_url.

3.3 Job Status Semantics

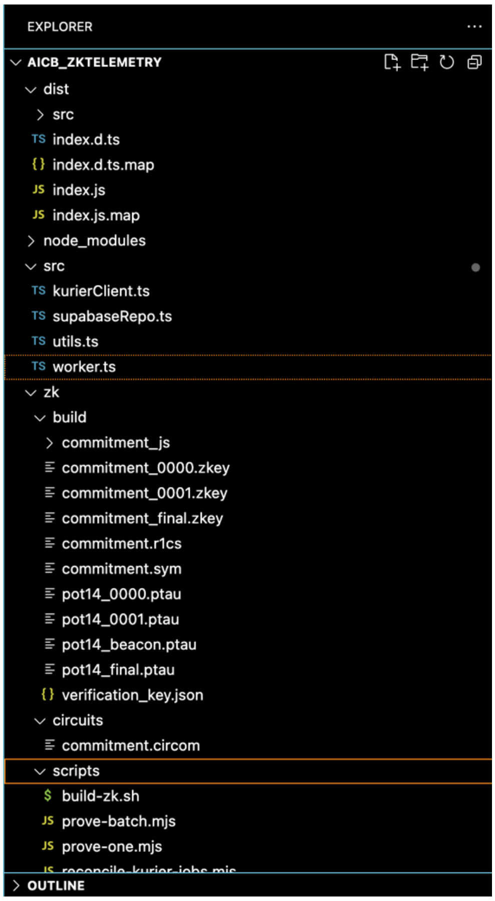
- **queued**: ready to be submitted
- **submitting**: reserved by worker, submission in progress
- **submitted**: Kurier accepted job; awaiting on-chain finalization or tx enrichment
- **verified**: job confirmed verified/finalized (if applicable in the flow)
- **failed**: permanent failure after retries or non-retryable error

4. Worker: Submission, Persistence & Reliability

4.1 Worker Responsibilities

The Proof Submission Worker is responsible for:

- fetching queued jobs in batches
- moving them to submitting
- submitting proofs to Kurier
- extracting and persisting:
  - kurier_request_id
  - raw response into kurier_response
  - (if available immediately) zkverify_tx_hash and zkverify_explorer_url
- updating final state (submitted / verified / failed) with retries and backoff logic



```
AICB_ZKTELEMETRY
  dist
    src
    index.d.ts
    index.d.ts.map
    index.js
    index.js.map
  node_modules
  src
    kurierClient.ts
    supabaseRepo.ts
    utils.ts
    worker.ts
  zk
    build
      commitment_js
      commitment_0000.zkey
      commitment_0001.zkey
      commitment_final.zkey
      commitment.r1cs
      commitment.sym
      pot14_0000.ptau
      pot14_0001.ptau
      pot14_beacon.ptau
      pot14_final.ptau
      verification_key.json
    circuits
      commitment.circom
    scripts
      build-zk.sh
      prove-batch.mjs
      prove-one.mjs
      reconcile-kurier-jobs.mjs
```

Figure 4. Project repository structure showing worker + Kurier sync scripts.

4.2 Running the Worker (High Throughput Mode)

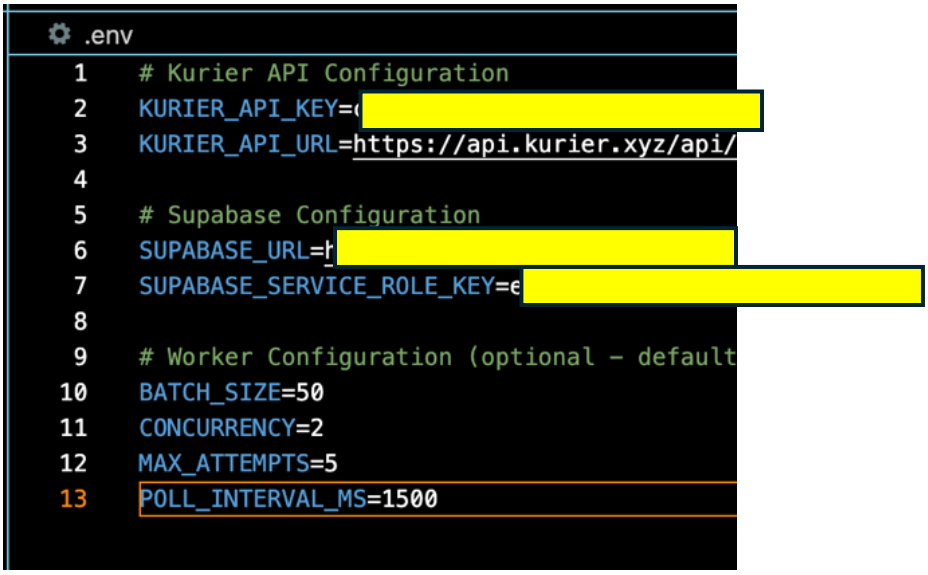
To run the worker directly from TypeScript (avoiding ESM build resolution issues), we start it via tsx: BATCH_SIZE=100 CONCURRENCY=20 npm run dev

This starts the worker and continuously processes proof jobs from Supabase.

4.3 Configuration

The system is configured through environment variables (.env) including:

- SUPABASE_URL
- SUPABASE_SERVICE_ROLE_KEY
- KURIER_API_URL
- KURIER_API_KEY
- optional worker parameters:
    - BATCH_SIZE
    - CONCURRENCY
    - MAX_ATTEMPTS
    - POLL_INTERVAL_MS



.env

```
# Kurier API Configuration
KURIER_API_KEY=[redacted]
KURIER_API_URL=https://api.kurier.xyz/api/

# Supabase Configuration
SUPABASE_URL=[redacted]
SUPABASE_SERVICE_ROLE_KEY=[redacted]

# Worker Configuration (optional - default
BATCH_SIZE=50
CONCURRENCY=2
MAX_ATTEMPTS=5
POLL_INTERVAL_MS=1500
```

Figure 5. Example .env configuration (API keys redacted).

4.4 Throughput & Load Testing

We intentionally tested the worker across a wide operational range—from near real-time confirmations (single-device submissions where proofs appear almost immediately after a smartphone session completes) up to **large batch submissions (e.g., ~1,000 proofs at once)** to stress-test end-to-end throughput.

This testing was essential: it helped us identify practical limits and bottlenecks across Kurier and zkVerify (rate limits, delayed transaction metadata availability, and optimal concurrency/batch sizing). Based on these findings, we calibrated the system to operate within a stable "sweet spot" that balances **speed, volume, and reliability**, while maintaining full traceability from Supabase job records to on-chain zkVerify extrinsics.

5. Kurier Job Tracking & tx Hash Synchronization

5.1 Why Sync Is Needed

In high-volume operation, Kurier job submission can return quickly while the on-chain transaction hash becomes available slightly later. To ensure complete auditability, we synchronize and backfill missing on-chain fields for submitted jobs.

5.2 Sync Script

The milestone includes a sync script:

- zk/scripts/sync-kurier-status.mjs

It identifies submitted jobs where zkverify_tx_hash is missing but kurier_request_id exists, fetches the status from Kurier, and updates the database accordingly.

Because the script is typically run from a shell, .env needs to be loaded:
set -a; source .env; set +a; node zk/scripts/sync-kurier-status.mjs

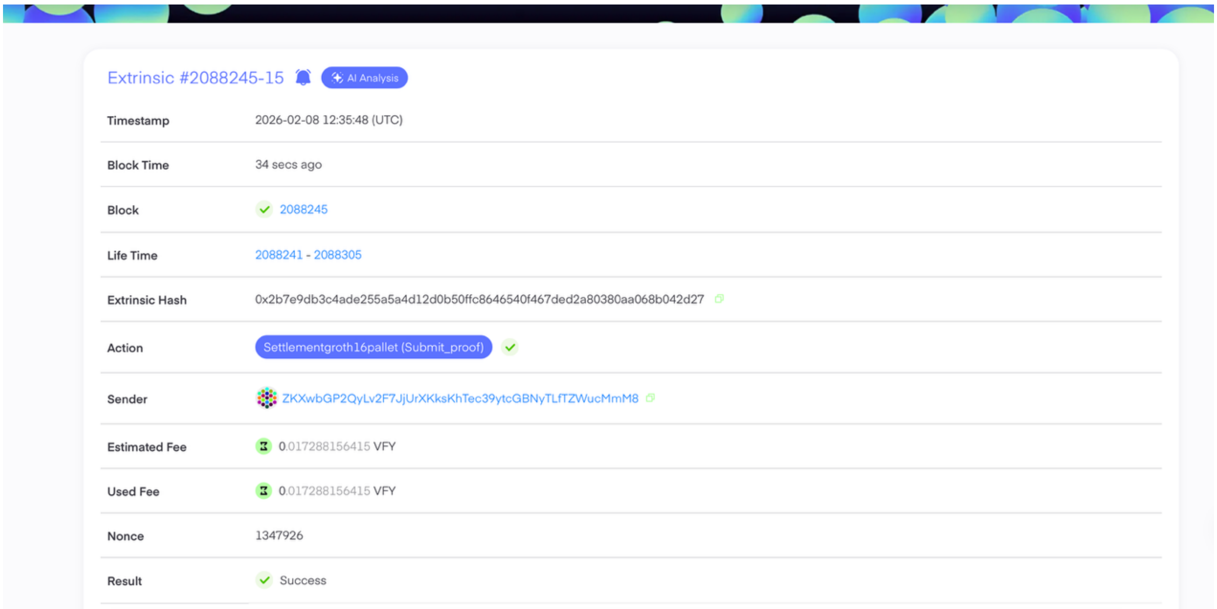
6. On-chain Verification (zkVerify Explorer)

6.1 Explorer Evidence (Extrinsic Details)

Each submitted proof is verifiable on zkVerify mainnet via the extrinsic hash. The system stores both:

- zkverify_tx_hash
- zkverify_explorer_url

This allows direct linking from Supabase proof job to the public on-chain transaction.



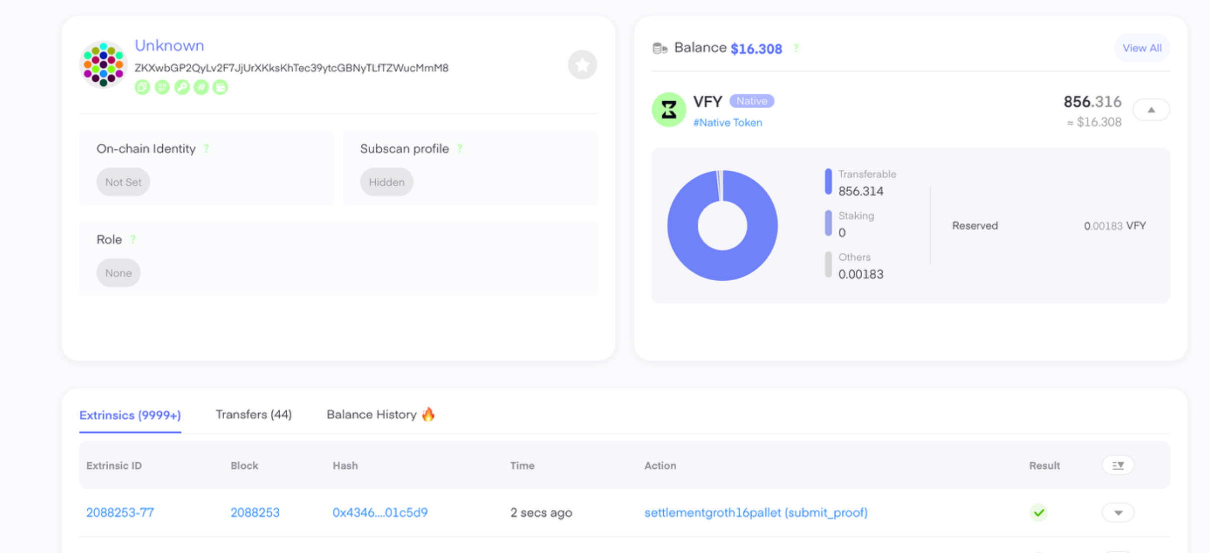
Extrinsic #2088245-15

Timestamp	2026-02-08 12:35:48 (UTC)
Block Time	34 secs ago
Block	✓ 2088245
Life Time	2088241 - 2088305
Extrinsic Hash	0x2b7e9db3c4ade255a5a4d12d0b50ffc8646540f467ded2a80380aa068b042d27
Action	Settlementgroth16pallet (Submit_proof) ✓
Sender	ZKXwbGP2QyLv2F7JjUrXKksKhTec39ytcGBNyTLfTZWucMmM8
Estimated Fee	0.017288156415 VFY
Used Fee	0.017288156415 VFY
Nonce	1347926
Result	✓ Success

Figure 6. zkVerify Subscan extrinsic details showing Submit_proof with success status.

6.2 High-volume Evidence (Account Activity)

At scale, proof submissions appear as high-frequency extrinsics (e.g., submit_proof) on the explorer.



Unknown
ZKXwbGP2QyLv2F7JjUrXKksKhTec39ytcGBNyTLfTZWucMmM8

On-chain Identity ?
Not Set

Subscan profile ?
Hidden

Role ?
None

Balance $16.308 ?
View All

VFY Native
#Native Token
856.316
≈ $16.308

Transferable
856.314
Staking
0
Others
0.00183

Reserved 0.00183 VFY

Extrinsics (9999+) Transfers (44) Balance History 🔥

Extrinsic ID	Block	Hash	Time	Action	Result
2088253-77	2088253	0x4346....01c5d9	2 secs ago	settlementgroth16pallet (submit_proof)	✓

Figure 7. Subscan account page showing high-volume extrinsics and proof submission activity.

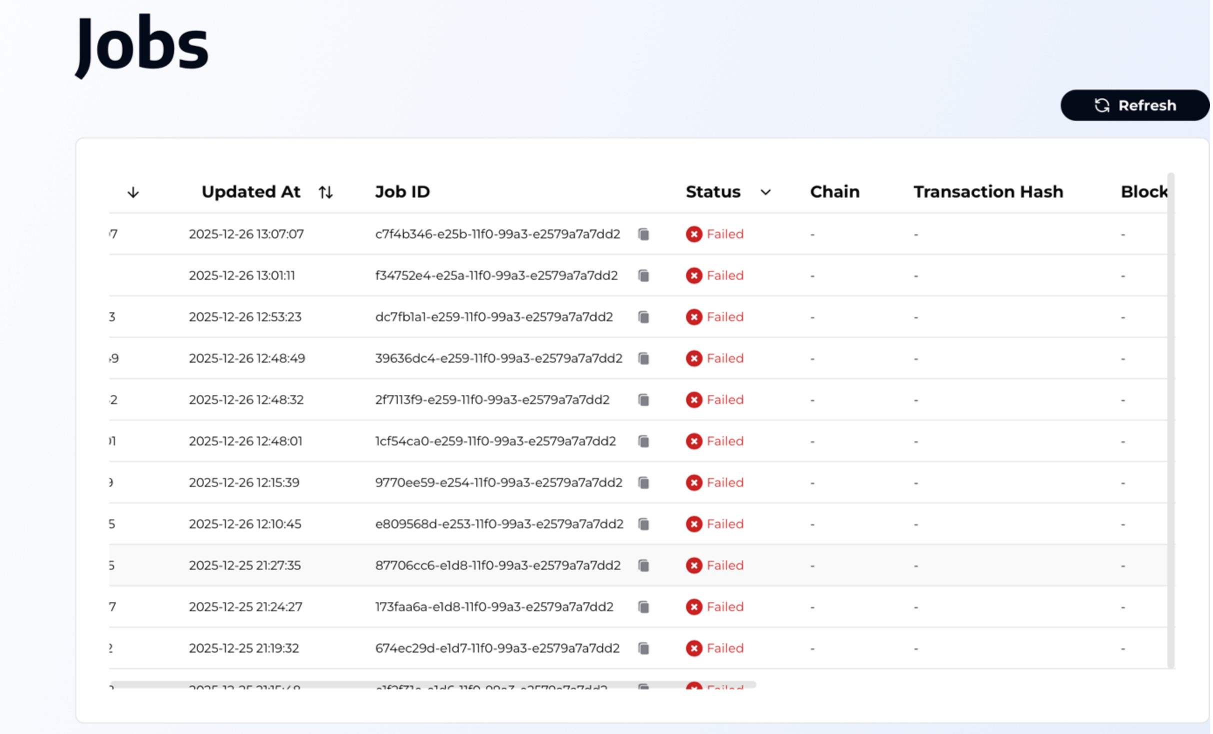
7. Engineering Iteration & Hardening (What We Learned)

Engineering Note

Milestone 2 was achieved through iterative hardening. Early job runs revealed edge cases (e.g., missing correlation fields and delayed transaction hash availability at high throughput). These findings directly informed the final resilience measures (robust response parsing, strict persistence checks, and a reconciliation/sync path), resulting in stable high-volume operation.

We addressed these by improving:

- **robust response parsing** (extract job ID from multiple response shapes)
- **persistence guarantees** (ensuring Kurier job ID is stored)
- **sync & reconciliation paths** (ability to recover missing transaction metadata)
- **runbook clarity** (standardized worker and sync commands)

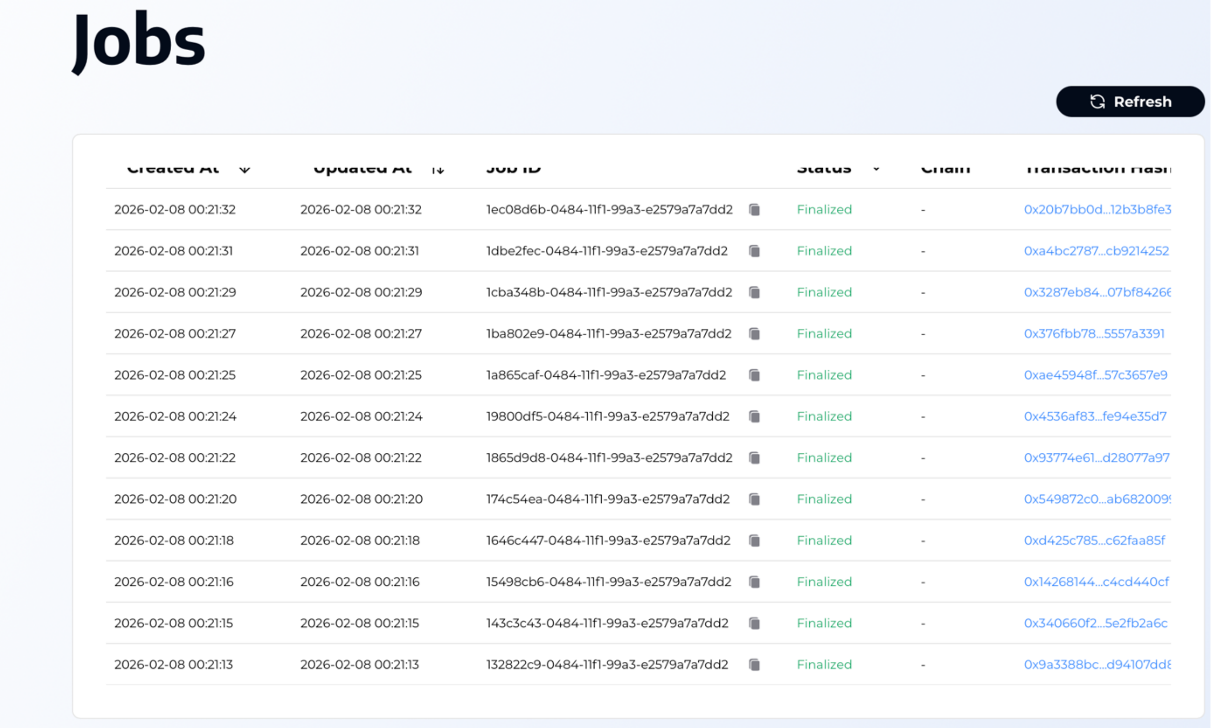


Jobs

Refresh

↓	Updated At ⇅	Job ID	Status	Chain	Transaction Hash	Block
7	2025-12-26 13:07:07	c7f4b346-e25b-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
	2025-12-26 13:01:11	f34752e4-e25a-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
3	2025-12-26 12:53:23	dc7fb1a1-e259-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
9	2025-12-26 12:48:49	39636dc4-e259-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
2	2025-12-26 12:48:32	2f7113f9-e259-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
1	2025-12-26 12:48:01	1cf54ca0-e259-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
	2025-12-26 12:15:39	9770ee59-e254-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
5	2025-12-26 12:10:45	e809568d-e253-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
5	2025-12-25 21:27:35	87706cc6-e1d8-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
7	2025-12-25 21:24:27	173faa6a-e1d8-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-
	2025-12-25 21:19:32	674ec29d-e1d7-11f0-99a3-e2579a7a7dd2	☒ Failed	-	-	-

Figure 8. Kurier “failed jobs” view from early iterations (before hardening).



Jobs

Refresh

Created At ↓	Updated At ⇅	Job ID	Status	Chain	Transaction Hash
2026-02-08 00:21:32	2026-02-08 00:21:32	1ec08d6b-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x20b7bb0d...12b3b8fe3
2026-02-08 00:21:31	2026-02-08 00:21:31	1dbe2fec-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0xa4bc2787...cb9214252
2026-02-08 00:21:29	2026-02-08 00:21:29	1cba348b-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x3287eb84...07bf8426
2026-02-08 00:21:27	2026-02-08 00:21:27	1ba802e9-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x376fbb78...5557a3391
2026-02-08 00:21:25	2026-02-08 00:21:25	1a865caf-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0xae45948f...57c3657e9
2026-02-08 00:21:24	2026-02-08 00:21:24	19800df5-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x4536af83...fe94e35d7
2026-02-08 00:21:22	2026-02-08 00:21:22	1865d9d8-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x93774e61...d28077a97
2026-02-08 00:21:20	2026-02-08 00:21:20	174c54ea-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x549872c0...ab682009
2026-02-08 00:21:18	2026-02-08 00:21:18	1646c447-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0xd425c785...c62faa85f
2026-02-08 00:21:16	2026-02-08 00:21:16	15498cb6-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x14268144...c4cd440cf
2026-02-08 00:21:15	2026-02-08 00:21:15	143c3c43-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x340660f2...5e2fb2a6c
2026-02-08 00:21:13	2026-02-08 00:21:13	132822c9-0484-11f1-99a3-e2579a7a7dd2	Finalized	-	0x9a3388bc...d94107dd

Figure 9. Kurier “finalized jobs” view after stabilization, showing transaction hashes.

8. Note on Attribution: Kurier Sender Accounts

Kurier submits proofs using its own sender infrastructure. Therefore, it is not always reliable to filter on-chain submissions by sender address alone when multiple clients share the same sender accounts. **In practice, we attribute proofs using the per-job kurier_request_id and the corresponding zkverify_tx_hash persisted in Supabase.**

Each AI.COREBLOCK proof is uniquely attributable via:

- kurier_request_id stored in Supabase for every proof job
- zkverify_tx_hash and zkverify_explorer_url backfilled via the sync script
- commitment_hash as the proof-level identifier in our system

Improvement Request
To make filtering easier at high volume, we propose attaching a project-specific identifier (e.g., verification key hash / vkHash, commitment_hash, or a clientReference tag) to each Kurier submission so Kurier-side filtering becomes trivial even when shared senders are used.

9. Conclusion

Milestone 2 successfully establishes a robust, high-throughput proof submission pipeline from AI.COREBLOCK's zkTelemetry system to zkVerify mainnet via Kurier. The solution provides end-to-end traceability by persisting correlation identifiers (kurier_request_id) and on-chain references (zkverify_tx_hash, zkverify_explorer_url) per proof job in Supabase, enabling independent verification through the zkVerify explorer.

In addition to achieving stable operation at higher volumes, the milestone resulted in a clear operational runbook (worker execution, monitoring queries, and transaction backfill via the sync script). The system is therefore ready to scale proof generation further.