SDG BLOCKCHAIN ACCELERATOR

Debugging and Testing Report

Atlas Ledger

1. Project Information

- **Project Name:** Atlas Ledger
- **Challenge & UNDP Office:** Reforestation projects with UNDP Burkina Faso
- **Report Version:** V1.0

2. Testing Approach

The V6 admin smart contract implementation follows a comprehensive testing strategy with both unit tests and integration testing on the Cardano blockchain.

Unit Testing: The Aiken smart contract has test coverage with 16 total tests, all passing. The test structure covers:

- Basic payout functionality (smoke tests)
- Happy path scenarios including extra outputs, multiple inputs, and higher milestone thresholds
- Comprehensive failure scenarios including missing admin signatures, underpaying beneficiaries, invalid outputs back to script, wrong references, missing datums, wrong beneficiary addresses, and insufficient milestone progress
- Advanced V6 features including split payments to beneficiaries, multi-input exact total payments, and beneficiary validation

Integration Testing: The smart contract integrates with a full NestJS backend service and Next.js frontend. The backend includes services for:

- V6 contract deployment and management
- Blockchain synchronization and monitoring (still improving)
- Transaction building and execution
- UTxO management and milestone tracking
- Smart donation features with automatic payout calculation

Integration tests conducted on Cardano Preprod Testnet:

- **Contract Deployment Tests:**
  - Successful deployment of V6 contracts with various milestone configurations
  - Datum construction and validation on live blockchain
  - Address derivation using real project stake keys
  - Transaction fee calculation and UTxO management
- **Payout Execution Tests:**
  - Single milestone payouts with admin signature verification
  - Multi-input transaction handling (collecting multiple UTxOs)
  - Split payment distributions to beneficiaries
  - Milestone progression validation (reached_up_to >= milestone_ix)
- **Blockchain Synchronization Tests:**
  - UTxO monitoring and status updates
  - Transaction confirmation tracking
  - Blockfrost API integration with rate limiting
  - Real-time contract balance updates
- **End-to-End Workflow Tests:**
  - Complete project lifecycle: deploy → fund → milestone completion → payout
  - Smart donation calculations with immediate vs. contract deposit splits ○ Frontend component integration with live blockchain data

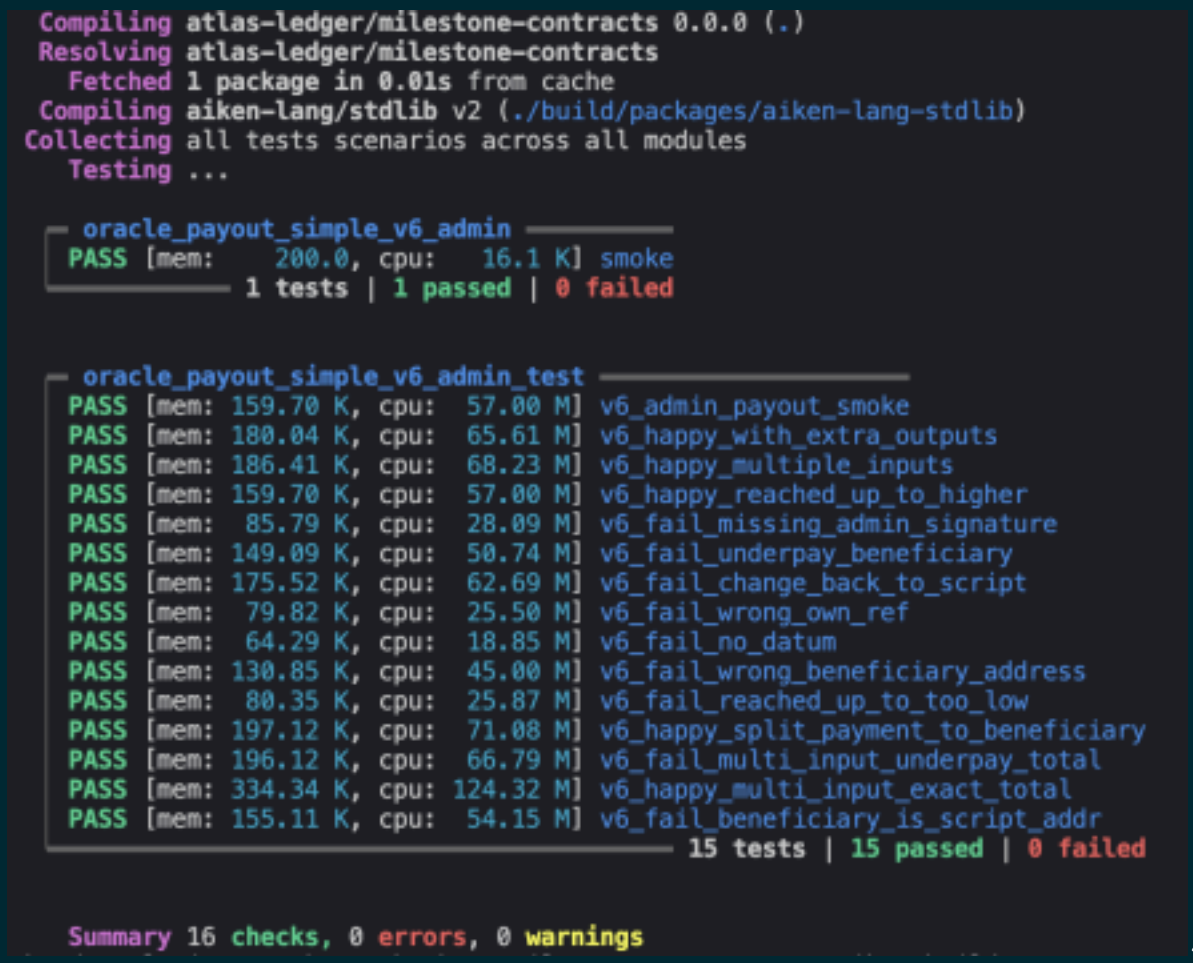
- Admin panel operations on real contracts

Edge Cases: The testing covers critical edge cases:

- Invalid datum handling (None datum rejected)
- Unauthorized signatures blocked
- Insufficient collateral (underpayment scenarios)
- Double spend prevention via change-back restrictions
- Multi-input transaction validation to prevent double-satisfaction attacks
- Beneficiary address validation to prevent impossible payouts

3. Error Logs

No critical errors found in the current implementation. All tests pass successfully:



4. Resolved Issues

Issue ID	Description	Root Cause	Resolution	Status
001	Multi-input underpay vulnerability	Original validator only checked individual outputs against individual inputs	Implemented total-sum comparison: sum of all script inputs vs. total paid to beneficiary	✅ Fixed
002	Split payment restriction	Validator required single output payment	Added paid_total_to() helper to sum payments across multiple outputs to same beneficiary	✅ Fixed

003	Double-satisfaction attack vector	Multiple script UTxOs could be satisfied by same beneficiary payment	Enhanced validation logic to require total_from_script <= paid_total	✅ Fixed
004	Beneficiary validation gap	No check preventing beneficiary == script address	Added bene_not_script guard to prevent impossible self-payouts	✅ Fixed

5. Optimization Notes

Smart Contract Optimization

- Review validator structure for micro-optimizations: consolidate repeated fold or sum operations into pre-computed accumulators where feasible.
- Simplify conditional branches in the validate() function to minimize script size and reduce CPU cost per execution.
- Investigate use of reference scripts for stable validator deployments to lower per-transaction fees.

Transaction Construction Optimization

- Introduce off-chain pre-aggregation logic to reduce the number of inputs required in large payout batches.
- Optimize collateral management by caching reusable key hashes and address derivations.
- Apply dynamic batching based on EvaluateTx results to ensure safe execution within ex-unit thresholds.

Backend and Off-Chain Services

- Implement asynchronous UTxO synchronization using event-based updates to minimize API calls to Blockfrost.
- Integrate transaction caching for read-only blockchain queries to reduce latency and API cost.
- Improve retry and back-off logic in transaction submission routines to handle network congestion gracefully.

Monitoring and Performance Evaluation

- Collect runtime metrics (CPU, memory, ex-units) from Preprod execution to build a cost-prediction model for mainnet deployment.
- Continuously profile validator execution via aiken eval and cardano-cli evaluate-transaction to identify future optimization opportunities.

6. Tools and Environments Used

Aiken CLI v1.1.19+e525483 (aiken check, aiken build)

Development Environment:

- Cardano preprod testnet integration

Blockchain Integration:

- Blockfrost API for blockchain queries (with intelligent rate limiting)
- Lucid Evolution v0.4.29 for transaction building and signing (@lucid-evolution/lucid)
- Lucid Evolution Utils v0.1.66 (@lucid-evolution/utils)
- Lucid Evolution BIP39 v0.2.11 (@lucid-evolution/bip39)

7. Remaining Issues / Next Steps

Current Status: Full test suite passing, deployed on the preprod net

Issue: Many-Donors Scaling

Problem (in the validator):

The current V6 validator executes once per donor UTxO and, on each run,

recomputes: ○ the sum of all script inputs in ctx.inputs, and

○ the sum paid to the beneficiary in ctx.outputs.

This makes every execution scan the whole transaction. As the number of donor inputs m grows, CPU and memory per input increase, and total ex-units grow even faster. This is intrinsic to the “pay the total once” rule implemented on-chain.

Consequence:

Payout transactions with many donors can approach per-transaction CPU limits

even when individual code paths are simple.

Proposal to solve it:

- **Structured on-chain flow:**
  Add a consolidation path (new redeemer branch) that merges many donor UTxOs into one script UTxO for the same milestone; the regular Payout
  then spends just one input.

- **Off-chain mitigation:**
  Chunk payouts into batches (e.g., ≤ 20 donors/tx) and EvaluateTx each built transaction; shrink the batch if steps approach your safety

threshold. **Deployment Status:**

- V6 admin smart contract successfully deployed and operational on Cardano Preprod testnet
- Contract addresses are derived dynamically per project using: validator_hash + project_stake_vkh
- Backend services fully integrated with deployed validator
- Frontend components operational with V6 contract interactions

Preprod Testnet Verification:

- Live contract addresses (HEX):
  - 106c8c2497ab7173c16a389f9948aa798a22d2d476e1785b0706dfa7af534b62ad660488b3c51fd524faeed907f5612f6682b37f7ae48393d8
  - 108c22570eeb82089ada1539e4709841e3d3b740fde3577ad700080e90b19bd26b60b052cd99d01cac3cdbf3ed4531d6483d019c6e49998ffd

Next steps for mainnet readiness:

1. Monitor real transaction performance on Preprod testnet
2. Validate multi-milestone payout scenarios with actual users
3. Security audit review of live contract behavior

4. Prepare mainnet deployment strategy

Ongoing monitoring priorities:

- Transaction success rates and error patterns
- UTxO management efficiency in multi-input scenarios
- Smart donation feature performance with milestone calculations
- Blockchain synchronization reliability