ANASTASIA LABS

Proof of Achievement - Milestone 5

Testing and Optimization for SanchoNet Features

Project Close-out Report

Project Number	1100024
Project Manager	Jonathan Rodriguez

ANASTASIA LABS

Contents

Project Name: Lucid Evolution: Redefining Off-Chain Transactions in Cardano
URL: Catalyst Proposal

Introduction

Lucid evolution from its inception had the goal to become the go-to off chain library, creating a streamlined development process for offchain on Cardano. We aimed to make this a reality by creating Lucid Evolution library and work on it 24/7. Everyday we help developers fix their problems, answer questions and also learn ourvelselves from them.

We had an amazing experience watching the growth of the library, hearing that our rejuvenation of the legacy Lucid library major Cardano dApps and entities rely on the enhanced Lucid Eolution for their operations.

We have crossed the 1000+ commits mark some time ago now, and watch the continous increase of dApps powered by the Evolution library.

Evidence Definition

Provided documentation and reports detailing the testing process conducted on SanchoNet features integrated with Lucid is available at https://github.com/Anastasia-Labs/lucid-evolution

SanchoNet Feature Implementation and Testing

Test Cases Overview

For this overview we will display what our testing suite covers, and how we ourselves are always on the lookout for upgrades / bugs / undiscovered minor issues and these is possible to track

Our testing suite for SanchoNet features is extensive and includes direct on-chain execution of tests. This approach shows that our transaction builder library is reliable in real-world scenarios. We can group these tests under:

DRep Operations
- Register DRep
- Deregister DRep
- Update DRep

Voting Delegation
- Delegate vote to DRep (Always Abstain)
- Delegate vote to DRep (Always No Confidence)
- Delegate vote to Pool and DRep

Combined Registration and Delegation

- Register and delegate to Pool
- Register and delegate to DRep
- Register and delegate to Pool and DRep

Script-based DRep Operations
- Register Script DRep
- Deregister Script DRep

Effect Library Integration

We have rewritten Lucid from scratch, incorporating the Effect library to provide developers with improved error managemen.

Through this we see better error handling via Effect library which allows developers for more precise operations as increased error messages handling make troubleshooting easier for everyone.

Our test suite, including the SanchoNet feature tests, is regularly executed as part of our continuous integration process.

Individual Test Cases Displayed

These test cases can be found in our `onchain-preview.test.ts`, `onchain-preprod.test.ts` and specifically the `governance.ts` files in our library.

Each test implements proper error handling through `Effect.orDie` and includes retry logic via `withLogRetry` to ensure reliable test execution. Our suite is designed to work across both `Preview` and `Preprod` networks, with appropriate network-specific configurations.:

DRep Operations

Register DRep

Tests the creation of new DReps by registering a reward address as a DRep with optimized fee calculations

```
export const registerDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .register.DRep(rewardAddress)
    .setMinFee(200_000n)
    .completeProgram();
  return signBuilder;
}).pipe(
  Effect.flatMap(handleSignSubmit),
  Effect.catchTag("TxSubmitError", (error) => Effect.fail(error)),
  withLogRetry,
  Effect.orDie,
);
```

Deregister DRep

Ensures proper removal of DRep credentials from the system

```
export const deregisterDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .deregister.DRep(rewardAddress)
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Update DRep

Validates the ability to modify existing DRep metadata and credentials

```
export const updateDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .updateDRep(rewardAddress)
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Voting Delegation

Delegate vote to DRep (Always Abstain)

Tests delegation to a DRep with an “always abstain” voting pattern

```
export const voteDelegDRepAlwaysAbstain = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .delegate.VoteToDRep(rewardAddress, {
      __typename: "AlwaysAbstain",
    })
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Delegate vote to DRep (Always No Confidence)

Validates delegation with “always no confidence” voting behavior

```
export const voteDelegDRepAlwaysNoConfidence = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .delegate.VoteToDRep(rewardAddress, {
      __typename: "AlwaysNoConfidence",
    })
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Delegate vote to Pool and DRep

Tests the complex scenario of simultaneous stake pool and DRep delegation

```
export const voteDelegPoolAndDRepAlwaysAbstain = Effect.gen(function* ($) {
  const { user } = yield* User;
  const networkConfig = yield* NetworkConfig;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const poolId =
    networkConfig.NETWORK == "Preprod"
      ? "pool1nmfr5j5rnqndprtazre802glpc3h865sy50mxdny65kfgf3e5eh"
      : "pool1ynfnjspgckgxjf2zeye8s33jz3e3ndk9pcwp0qzaupzvvd8ukwt";

  const signBuilder = yield* user
    .newTx()
    .delegate.VoteToPoolAndDRep(rewardAddress, poolId, {
      __typename: "AlwaysAbstain",
    })
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Combined Registration and Delegation

Register and delegate to Pool

Tests simultaneous stake registration and pool delegation

```
export const registerAndDelegateToPool = Effect.gen(function* ($) {
  const { user } = yield* User;
  const networkConfig = yield* NetworkConfig;
  const poolId =
    networkConfig.NETWORK == "Preprod"
      ? "pool1nmfr5j5rnqndprtazre802glpc3h865sy50mxdny65kfgf3e5eh"
      : "pool1ynfnjspgckgxjf2zeye8s33jz3e3ndk9pcwp0qzaupzvvd8ukwt";

  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .registerAndDelegate.ToPool(rewardAddress, poolId)
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Register and delegate to DRep

combined DRep registration and voting delegation

```
export const registerAndDelegateToDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const signBuilder = yield* user
    .newTx()
    .registerAndDelegate.ToDRep(rewardAddress, {
      __typename: "AlwaysAbstain",
    })
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Register and delegate to Pool and DRep

Tests the most complex scenario of registering and delegating to both a pool and DRep in a single transaction

```
export const registerAndDelegateToPoolAndDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const rewardAddress = yield* pipe(
    Effect.promise(() => user.wallet().rewardAddress()),
    Effect.andThen(Effect.fromNullable),
  );
  const networkConfig = yield* NetworkConfig;
  const poolId =
    networkConfig.NETWORK == "Preprod"
      ? "pool1nmfr5j5rnqndprtazre802glpc3h865sy50mxdny65kfgf3e5eh"
      : "pool1ynfnjspgckgxjf2zeye8s33jz3e3ndk9pcwp0qzaupzvvd8ukwt";
  const signBuilder = yield* user
    .newTx()
    .registerAndDelegate.ToPoolAndDRep(rewardAddress, poolId, {
      __typename: "AlwaysAbstain",
    })
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Script-based DRep Operations

Register Script DRep

Tests the registration of script-based DReps, including proper script attachment and validation

```
export const registerScriptDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const { rewardAddress, script } = yield* AlwaysYesDrepContract;
  const signBuilder = yield* user
    .newTx()
    .register.DRep(rewardAddress, undefined, Data.void())
    .attach.Script(script)
    .completeProgram();
  return signBuilder;
}).pipe(
  Effect.flatMap(handleSignSubmit),
  Effect.catchTag("TxSubmitError", (error) => Effect.fail(error)),
  withLogRetry,
  Effect.orDie,
);
```

Deregister Script DRep

Ensures proper cleanup of script-based DRep credentials

```
export const deregisterScriptDRep = Effect.gen(function* ($) {
  const { user } = yield* User;
  const { rewardAddress, script } = yield* AlwaysYesDrepContract;
  const signBuilder = yield* user
    .newTx()
    .deregister.DRep(rewardAddress, Data.void())
    .attach.Script(script)
    .completeProgram();
  return signBuilder;
}).pipe(
  Effect.flatMap(handleSignSubmit),
  Effect.catchTag("TxSubmitError", (error) => Effect.fail(error)),
  withLogRetry
  Effect.orDie,
);
```

Committee Certificates Implementation (PR 313)

Following the initial governance features, Lucid Evolution expanded its capabilities to include committee-related operations:

1. Committee Hot Key Authorization: A new method `authCommitteeHot` was added to authorize a hot key for a committee member

```
packages/lucid/src/tx-builder/TxBuilder.ts
startLine: 479
endLine: 490
```

2. Committee Member Resignation: The `resignCommitteeHot` method was introduced to allow committee members to resign their position

```
packages/lucid/src/tx-builder/TxBuilder.ts
startLine: 491
endLine: 499
```

and governance specific `propose` validator operations together with the updated script attachments