ANASTASIA LABS

Proof of Achievement - Milestone 4

Keeping Pace with Cardano Evolution

Project Number 1100024
Project Manager Jonathan Rodriguez

ANASTASIA LABS

Contents

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

Introduction

Our team has been hard at work, crafting a toolkit that makes Cardano's new governance capabilities accessible and intuitive for developers and users. This report dives into the nuts and bolts, showcasing how we've translated Cardano's complex governance model into a developer-friendly library and how our extensive testing modules cover all endpoints and succesfully run both on Preview and Preprod networks with every update we push

SanchoNet Feature Implementation and Testing

Document with test cases covering various aspects of SanchoNet features, such as functionality, performance, and integration:

Test Cases Overview

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

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:

DRep Operations

Register DRep

```
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

```
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
```

```
9       .deregister.DRep(rewardAddress)
10      .completeProgram();
11    return signBuilder;
12  }).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Update DRep

```
1   export const updateDRep = Effect.gen(function* ($) {
2     const { user } = yield* User;
3     const rewardAddress = yield* pipe(
4       Effect.promise(() => user.wallet().rewardAddress()),
5       Effect.andThen(Effect.fromNullable),
6     );
7     const signBuilder = yield* user
8       .newTx()
9       .updateDRep(rewardAddress)
10      .completeProgram();
11    return signBuilder;
12  }).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Voting Delegation

Delegate vote to DRep (Always Abstain)

```
1   export const voteDelegDRepAlwaysAbstain = Effect.gen(function* ($) {
2     const { user } = yield* User;
3     const rewardAddress = yield* pipe(
4       Effect.promise(() => user.wallet().rewardAddress()),
5       Effect.andThen(Effect.fromNullable),
6     );
7     const signBuilder = yield* user
8       .newTx()
9       .delegate.VoteToDRep(rewardAddress, {
10        __typename: "AlwaysAbstain",
11      })
12      .completeProgram();
13    return signBuilder;
14  }).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Delegate vote to DRep (Always No Confidence)

```
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

```
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",
    })
```

```
18       completeProgram();
19   return signBuilder;
20 }).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Combined Registration and Delegation

Register and delegate to Pool

```
1  export const registerAndDelegateToPool = Effect.gen(function* ($) {
2    const { user } = yield* User;
3    const networkConfig = yield* NetworkConfig;
4    const poolId =
5      networkConfig.NETWORK == "Preprod"
6        ? "pool1nmfr5j5rnqndprtazre802glpc3h865sy50mxdny65kfgf3e5eh"
7        : "pool1ynfnjspgckgxjf2zeye8s33jz3e3ndk9pcwp0qzaupzvvd8ukwt";
8
9    const rewardAddress = yield* pipe(
10     Effect.promise(() => user.wallet().rewardAddress()),
11     Effect.andThen(Effect.fromNullable),
12   );
13   const signBuilder = yield* user
14     .newTx()
15     .registerAndDelegate.ToPool(rewardAddress, poolId)
16     .completeProgram();
17   return signBuilder;
18 }).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Register and delegate to DRep

```
1  export const registerAndDelegateToDRep = Effect.gen(function* ($) {
2    const { user } = yield* User;
3    const rewardAddress = yield* pipe(
4      Effect.promise(() => user.wallet().rewardAddress()),
5      Effect.andThen(Effect.fromNullable),
6    );
7    const signBuilder = yield* user
8      .newTx()
9      .registerAndDelegate.ToDRep(rewardAddress, {
10       __typename: "AlwaysAbstain",
```

```
    .completeProgram();
  return signBuilder;
}).pipe(Effect.flatMap(handleSignSubmit), withLogRetry, Effect.orDie);
```

Register and delegate to Pool and DRep

```
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

```
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

```
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

Effect Library Integration

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

Through this we provide better error handling via Effect library which allows developers for more precise error messages.

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

Test Execution Results

Our testing methodology involves executing over 100 CI tests directly on the Cardano preprod and preview networks. This approach allows for real-world validation by running tests on actual Cardano networks, as we ensure that our library functions correctly in production-like environments.

As part of our CI pipeline, these tests are run regularly, making sure of ongoing compatibility and reliability with each new version released of `lucid-evolution`.

Proof Video

Example test execution of all our tests in our testing suite passing (from `onchain-preprod.test.ts`):

Proof video

VIDEO

Testing of Conway Era functions

To ensure consistency across different Cardano environments, we run our test suite on both the Preview and Preprod networks:

- `packages/lucid/test/onchain-preview.test.ts`
- `packages/lucid/test/onchain-preprod.test.ts`

GIF Testrun

Succesful Test Result

This GIF, **a onchain test** running in terminal, showcases
that our test packages are working as intended by running succesful test cases

Bug Identification

Ongoing Development

The governance features in Lucid Evolution are in a state of continuous development and refinement. As the Conway era and its associated governance mechanisms are still evolving, our library adapts to keep pace with the latest community discussions and protocol changes.

Our codebase is designed to be easily adaptable to changes in the governance protocol. We maintain an extensive suite of on-chain tests to ensure reliability across different scenarios. We closely follow constitutional workshops and governing body elections to align our features with the community's needs.

Optimizations

We made further enhancements to our governance-related endpoints and made sure they are all live and functioning with each of our releases on the Cardano Network for developers to utilize.

We've implemented a range of DRep (Delegate Representative) operations, including registration, deregistration, and updates. Added support for various voting delegation scenarios, including "Always Abstain" and "Always No Confidence" options. Easy convenience functions to streamline functions that allow users to register and delegate in a single transaction and the extended TxBuilder including governance-specific methods, to enabing creation of highly efficient governance era transactions.

After our implementations we took a look back at how our Transaction builder (`TxBuilder`) was functioning and made iterative design choices to increase performance in all our endpoints.

How?

By strategically making IO-bound computations optional by allowing preset protocol parameters during Lucid initialization and preset wallet UTXOs during transaction building. As a result of this design choice we can increase the quality of UX for Cardano Governance era - where all of us are most focused at the moment. This will be followed by more deep dives into how we can make the library smoother and easier to use for all developers on Cardano.

To not crowd this report with big code blocks, we present the direct change visible between these two states of the evolution library governance endpoints:

- Previous version (Unoptimized)
- Newest version (Optimized)

Performance Benchmark

Build times represented in miliseconds

Previous Governance Endpoint

- registerDRep: 2234ms
- updateDRep: 2161ms
- deregisterDRep: 2261ms
- registerScriptDRep: 2198ms
- deregisterScriptDRep: 2195ms

Optimized Governance Endpoint

- registerDRep: Slashed from 2234ms to a mere 11ms
- updateDRep: Improved from 2161ms to just 9ms
- deregisterDRep: Enhanced from 2261ms to 9ms
- registerScriptDRep: Reduced from 2198ms to 19ms
- deregisterScriptDRep: Down from 2195ms to 19ms

A direct comparison between the legacy lucid library and the evolution library can not be provided as governance features are not supported on Lucid 0.10.7. The optimized results are a comparison between previos implementation and its current state in Lucid Evolution library.