SDG BLOCKCHAIN ACCELERATOR

Thallo Technical Architecture Document

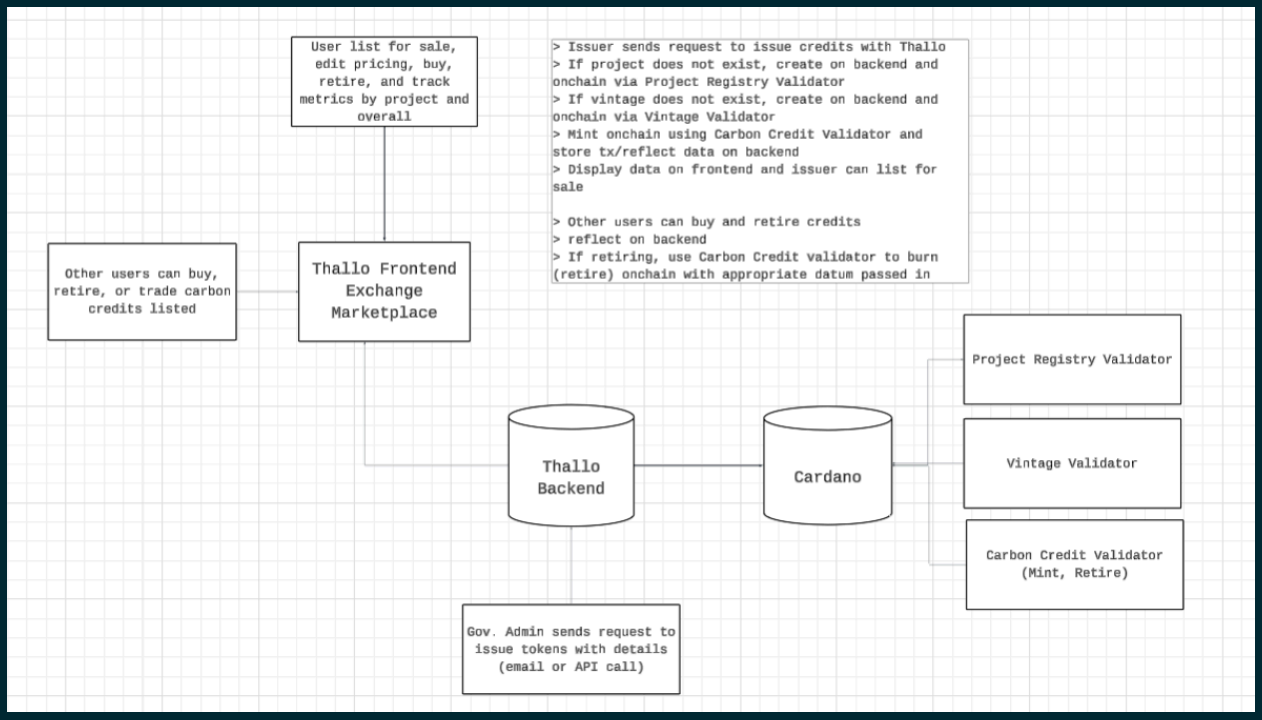
1. Project Information

- **Project Name:** Thallo Carbon Credit Marketplace & Exchange
- **Challenge & UNDP Office: Tanzania**
- **Document Version:** 1.01

2. Overview

Thallo is an all in one carbon credit issuance, marketplace, and exchange platform. Thallo allows carbon credit issuers and registries to custody and mint onchain or issue their carbon credits directly onchain. Credits can then be listed for sale and traded and/or retired on Thallo's marketplace. In the background, all carbon credit actions are reflected onchain.

3. System Architecture Diagram



4. Blockchain Design

Thallo utilizes 3 different cardano smart contracts/validators

1. Project Registry Validator
    a. Creates projects with a *datum* that contains basic information about the project. Can consider expanding *datum* to store more information onchain

b. Enforces the only valid transactions are signed by the protocol owner (Thallo)

```
pub type ProjectDatum {
  project_id: ByteArray,
  project_name: ByteArray,
  project_owner: VerificationKeyHash,
  is_active: Bool,
}

pub type ProjectAction {
  CreateProject
  UpdateProject
  DeactivateProject
}
```

2. Project Vintage validator
    a. Creates vintages associated with a specific projectId using a *VintageDatum*. Can also update vintage information. Can also be expanded if necessary.
    b. Enforces only valid transactions are signed by the protocol owner (Thallo)

```
18
19  pub type VintageDatum {
20    project_id: ByteArray,
21    vintage_year: Int,
22    methodology: ByteArray,
23    verification_body: ByteArray,
24    project_owner: VerificationKeyHash,
25  }
26
27  pub type VintageAction {
28    CreateVintage
29    UpdateVintage
30  }
```

3. Carbon Credit Validator

a. Handles the minting and retirement of carbon credit assets using a *mint* validator. Takes a *MintAction* which is either *MintCredits* or *RetireCredits*. Both take a *CarbonCredit* type which specifies the projectId, vintage, and how many to mint or retire.

b. Utxo output is enforced that cannot retire credits that do not exist / don't have
c. Enforced that only the protocol owner (Thallo) can mint credits

```
32
33  pub type CarbonCredit {
34    project_id: ByteArray,
35    vintage: Int,
36    amount: Int,
37  }
38
39  pub type MintAction {
40    MintCredits { credits: CarbonCredit }
41    RetireCredits { credits: CarbonCredit }
42  }
43
```

5. Data Flow & Transaction Lifecycle

1. Carbon credit issuer contacts Thallo to issue carbon credits for a specific project and vintage.

2. If necessary, project and vintage are created on both Thallo's backend and on Cardano. The project registry validator checks the signature and datum for project and vintage creation if necessary.

3. Transaction is created to issue carbon credits for specific project and vintage. The carbon credits validator validates this transaction. If valid, the tokens are minted and UTxO is updated on Cardano. Data is also reflected in Thallo's backend.

4. All data and metrics are visible on Thallo's frontend. If listed for sale, other users will see tokens available for purchase from the issuer.

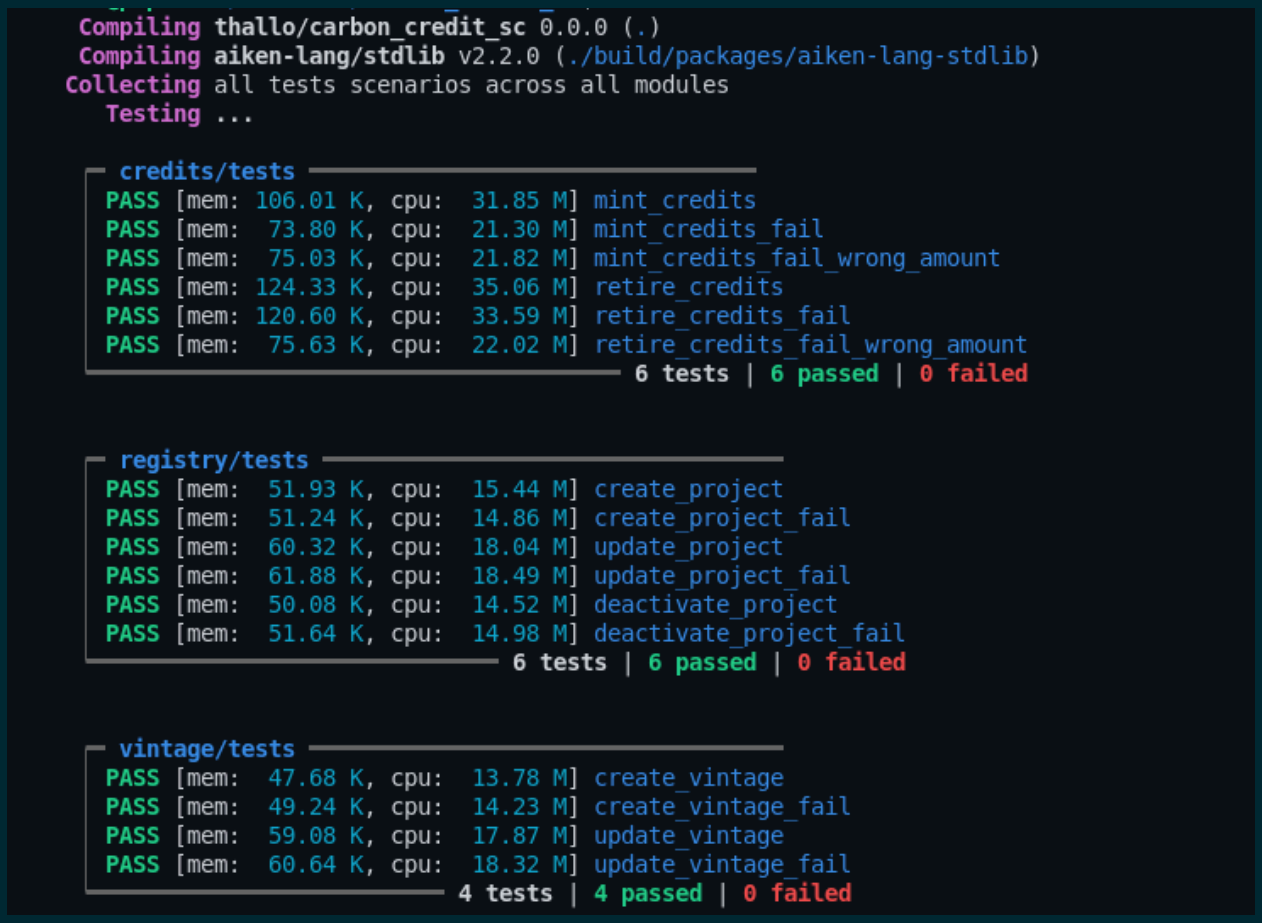
5. When retired, the carbon credits validator is used to validate the retirement request against the datum and signatures. If valid, tokens are burned (retired) onchain and reflected in Thallo's backend as well as displayed on the frontend.

6. Off-chain Components

1. Thallo backend (database and AWS) which mirrors carbon credit and project/vintage information. Also stores extra data required for the frontend like historical metrics, login information, etc. Basically all the information required for the frontend to display and function properly. The backend is what ultimately interfaces with Cardano.

2. Thallo frontend exchange marketplace and admin portal. UI for users to issue, buy, sell, and retire carbon credits. Also a UI/UX for Thallo's admin portal. Interfaces to the backend.

7. Sandbox/Testnet Results



Each of the 3 validators contains unit tests that test the proper amounts are enforced upon retirement/minting and that the protocol owner (Thallo) must be the signatory to create projects, vintages and issue and retire carbon credits.

https://preprod.cardanoscan.io/address/addr_test1vpcpquku64qz3ltmhjfn8ya3p2pykfp0cnk95vfh85aet4qe9drpy

Is the testnet address used. Successful preprod transactions can be seen there.

8. Tools and Environments Used

- Aiken compiler v1.1.19

- Aiken stdlib v2.2.0
- Blockfrost (RPC for preprod)
- Lucid for typescript scripting, deployment, testing, blockchain calls

9. Remaining Considerations / Next Steps

- Additional integration testing and stress testing on cardano preprod using Blockfrost and Lucid
- Possibly explore expanding UTxO model for enforcement/data storage of creating projects and vintages
- Figure out scripting deserialization with lucid-evolution for validator datums