ANASTASIA LABS

Linear Vesting

Project Design Specification

Contents

1. Getting Started with Linear Vesting

1.1. Table of Contents

Contents

2. Introduction

The Linear Vesting contract provides a mechanism for releasing Cardano Native Tokens gradually over a specified timeframe, with customization options to fit different requirements. It's useful for token compensation, DAO treasury vesting, and installment payment plans.

3. Getting Started

3.1. Prerequisites

Before you begin, ensure you have Nix installed on your system. Nix is used for package management and to provide a consistent development environment. If you don't have Nix installed, you can do so by running the following command:

3.1.1. Official option

Nix

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

3.1.2. Preferred option

Determinate Systems text curl –proto '=https' –tlsv1.2 -sSf -L https://install.determinate.systems/nix | sh -s – install

Make sure to enable Nix Flakes by editing either /.config/nix/nix.conf or /etc/nix/nix.conf on your machine and add the following configuration entries: text experimental-features = nix-command flakes ca-derivations allow-import-from-derivation = true

Optionally, to improve build speed, it is possible to set up binary caches by adding additional configuration entries: text substituters = https://cache.nixos.org https://cache.iog.io https://cache.zw3rk.com trusted-public-keys = cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY= hydra.iohk.io:f/Ea+s+dFdN+3Y/G+FDgSq+a5NEWhJGzdjvKNGv0/EQ= loony-tools:pr9m4BkM/5/eSTZlkQyRt57Jz7OMBxNSUiMC4FkcNfk=

To facilitate seamlessly moving between directories and associated Nix development shells we use direnv and nix-direnv: Your shell and editors should pick up on the .envrc files in different directories and prepare the environment accordingly. Use direnv allow to enable the direnv environment and direnv reload to reload it when necessary. Otherwise, the .envrc file contains a proper Nix target which will be used with the nix develop –accept-flake-config command. To install both using `nixpkgs`: `nix`

```
profile install nixpkgs#direnv profile install nixpkgs#nix-direnv
```

4. Building and Testing

Once Nix is installed, you should be able to seamlessly use the repository to develop, build and run packages. Download the Git repository: text git clone https://github.com/Anastasia-Labs/linear-vesting.git

Navigate to the repository directory: text cd linear-vesting

Activate the development environment with Nix: text nix develop –accept-flake-config

Or text make shell

Please be patient when building nix development environment for the first time, as it may take a very long time. Subsequent builds should be faster. Additionally, when you run nix run `.#help` you'll get a list of scripts you can run, the Github CI (nix flake check) is setup in a way where it checks the project builds successfully, haskell format is done correctly, and commit message follows conventional commits. Before pushing you should run cabal run, nix run `.#haskellFormat` (automatically formats all haskell files, including cabal), if you want to commit a correct format message you can run cz commit Build: text make build

Execute the test suite: text make test

5. Understanding the Linear Vesting Contract

text pvalidateVestingScript :: Term s (PVestingDatum :–> PVestingRedeemer :–> PScriptContext :–> PUnit)

The linear vesting validator is not a parameterized one. All its customization needs are fulfilled by the datum (VestingDatum) of the locked UTxO.== Datum text data VestingDatum =

VestingDatum { beneficiary :: Address , vestingAsset :: AssetClass , totalVestingQty :: Integer , vestingPeriodStart :: Integer , vestingPeriodEnd :: Integer , firstUnlockPossibleAfter :: Integer , totalInstallments :: Integer }

Field	Type	Description
beneficiary	Address	The address of the recipient who will receive the vested assets.
vestingAsset	AssetClass	Identifies the specific asset being vested, typically represented by a policy ID and asset name.
totalVestingQty	Integer	The total quantity of the asset to be vested over the entire vesting period.
vestingPeriodStart	Integer	The timestamp marking the beginning of the vesting period.
vestingPeriodEnd	Integer	The timestamp marking the end of the vesting period.
firstUnlockPossibleAfter	Integer	The earliest timestamp at which the first portion of the vested assets can be claimed.
totalInstallments	Integer	The number of separate releases or “unlocks” over which the total vesting quantity will be distributed.

5.1. Redeemer

text data VestingRedeemer= PartialUnlock | FullUnlock

5.1.1. Partial Unlock

When using the PartialUnlock redeemer, the following conditions and actions apply: Timing: The transaction can only be validated after firstUnlockPossibleAfter and before vestingPeriodEnd. Claim Amount: The beneficiary can claim a portion of the vested assets. The claimable amount is proportional to the time elapsed since vestingPeriodStart. Remaining Assets: Any unclaimed assets must be returned to the validator address. Datum Preservation: The original datum must remain intact for the returned assets.==== Validation Requirements: The transaction must occur within the specified time window, between firstUnlockPossibleAfter and vestingPeriodEnd. The claimed amount must not exceed the proportionally vested amount. Unclaimed assets must be properly returned with the original datum. By using the PartialUnlock redeemer, the beneficiary initiates a partial claim of their vested assets while ensuring the integrity of the vesting schedule and the security of any remaining assets.=== Full Unlock When the FullUnlock redeemer is used: It allows the beneficiary to claim the entire remaining balance of vested assets from the UTxO. This redeemer is only valid for transactions submitted after the vestingPeriodEnd.==== Validation Requirements: The current time is greater than vestingPeriodEnd. The entire remaining balance of vested assets is being claimed. The beneficiary’s address matches the one specified in the datum. Using the FullUnlock redeemer signals the intent to complete the vesting process and withdraw all remaining assets, but it can only be successfully applied once the vesting period has concluded.= Exporting Plutarch Scripts Once the script is compiled, locate the generated serialised .json file in your project’s output directory. This file contains the bytecode that will be deployed on the Cardano blockchain. You can copy and paste the contents of this file into offchain directory.= Building and Testing OffChain== Install package text npm install `@anastasia-labs/direct-offer-offchain`

or text pnpm install `@anastasia-labs/direct-offer-offchain`

5.2. Local Build

In main directory text pnpm run build

5.3. Test framework

https://github.com/vitest-dev/vitest

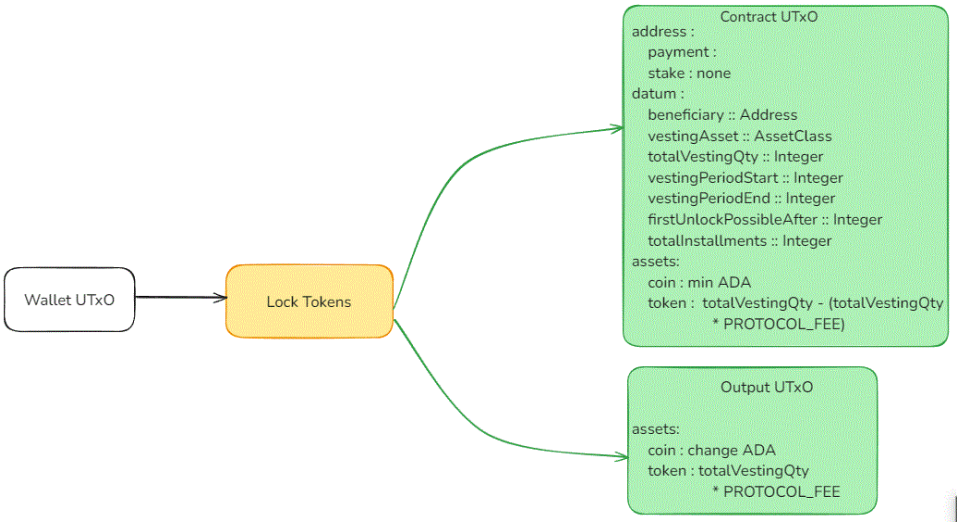
5.4. Running Tests

text pnpm test

6. Transaction Building

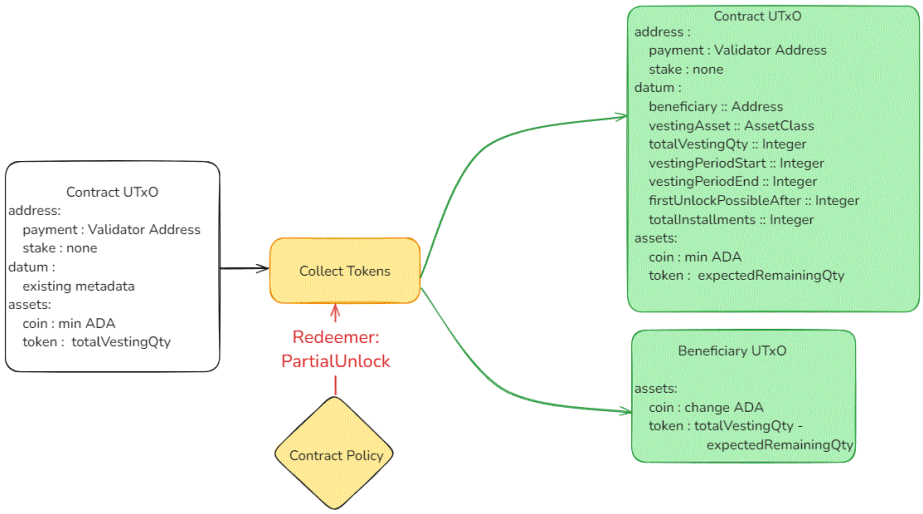
6.1. Lock Assets at Vesting Contract

A predefined amount of tokens is sent to smart contract. This is total amount to be vested. The start date (e.g., vestingPeriodStart) , end date (e.g., vestingPeriodEnd) of vesting period , the number of installments (e.g., totalInstallments), first installment date (e.g., firstUnlockPossibleAfter), and beneficiary's address is specified which is only address that can claim vested tokens. All these fields are specified in vestingDatum.



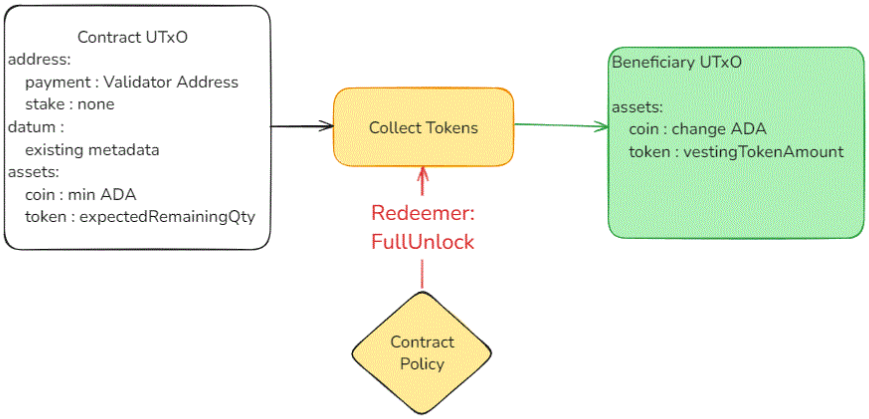
6.2. Collect token with Partial-Unlock Redeemer

With PartialUnlock redeemer after firstUnlockPossibleAfter and before vestingPeriodEnd, beneficiary can claim vested asset in proportion to time that has passed after vestingPeriodStart. The remaining assets needs to be sent back to validator address with original datum intact otherwise validation will fail.



6.3. Collect all tokens with Full-Unlock Redeemer

After vestingPeriodEnd, beneficiary can claim entire remaining balance at UTxO using FullUnlock redeemer.



7. Additional Resources

7.1. Demeter Workspace

To provide seamless experience for running and trying out our application click workspace button below this will start workspace in Demeter with our repository code automatically cloned. Code in Workspace There is also GitBook available for more details.