besiwims /
plutus-tx-template

Code Pull requests Actions Projects Wiki Security Insights

Plutus vs Solidity : Property Smart Contract

Jump to bottom

Bernard Sibanda edited this page on Sep 10, 2024 · 1 revision

Solidity and Plutus Smart Contracts

Ethereum (Solidity) and Cardano (Plutus)

Combined Example: Fractionalizing Property

1. Solidity Smart Contract (Ethereum)

Key Features:

- Written in Solidity.
- Uses the ERC20 token standard.
- Functions for issuing and transferring tokens.
- Simple ownership model.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract PropertyToken is ERC20, Ownable {
    address public propertyOwner;
    uint256 public totalTokens;

    event SharesIssued(address indexed to, uint256 amount);

    constructor(string memory name, string memory symbol, uint256 _totalTokens) ERC20(name, s
        propertyOwner = msg.sender;
        totalTokens = _totalTokens;
        _mint(propertyOwner, totalTokens);
    }
    function issueShares(address to, uint256 amount) public onlyOwner {
        require(amount > 0, "Amount must be greater than 0");
        require(balanceOf(propertyOwner) >= amount, "Insufficient balance to issue shares");
        _transfer(propertyOwner, to, amount);
        emit SharesIssued(to, amount);
    }

    function sellShares(address from, address to, uint256 amount) public onlyOwner {
        require(amount > 0, "Amount must be greater than 0");
        require(balanceOf(from) >= amount, "Insufficient balance to sell shares");
        _transfer(from, to, amount);
    }
}
```

2. Plutus Smart Contract (Cardano)

Key Features:

- Written in Haskell.
- Uses Plutus for token management and smart contract execution.
- Functions for minting and transferring tokens.
- Involves Plutus data types and on-chain validation.

```
{-# LANGUAGE DataKinds           #-}
{-# LANGUAGE NoImplicitPrelude   #-}
{-# LANGUAGE OverloadedStrings   #-}
{-# LANGUAGE ScopedTypeVariables #-}
{-# LANGUAGE TemplateHaskell     #-}
{-# LANGUAGE TypeApplications    #-}
{-# LANGUAGE TypeFamilies        #-}
{-# LANGUAGE FlexibleContexts    #-}

module PropertyToken where

import           PlutusTx
import           PlutusTx.Prelude
import           Ledger
import           Ledger.Value as Value
import           Ledger.Typed.Scripts as Scripts
import           Ledger.Constraints as Constraints
import           Playground.Contract
import           Plutus.Contract

-- Define the currency symbol and token name
propertyTokenSymbol :: CurrencySymbol
```

```haskell
propertyTokenSymbol = "propertyToken"

propertyTokenName :: TokenName
propertyTokenName = "Property"

-- Define the schema for the contract
data PropertyTokenSchema =
    MintToken Integer | TransferToken Integer PubKeyHash deriving (Show, Eq)

PlutusTx.unstableMakeIsData ''PropertyTokenSchema

-- Define the contract
mkPropertyToken :: Integer -> Contract w s Text ()
mkPropertyToken amount = do
    let token = Value.singleton propertyTokenSymbol propertyTokenName amount
    let lookups = Constraints.mintingPolicy (mintPolicy amount)
    let tx = Constraints.mustMintValue token
    ledgerTx <- submitTxConstraintsWith @Scripts.MintingPolicy lookups tx
    awaitTxConfirmed $ getCardanoTxId ledgerTx
    logInfo @String $ "Minted " ++ show amount ++ " tokens"

transferToken :: Integer -> PubKeyHash -> Contract w s Text ()
transferToken amount recipient = do
    let token = Value.singleton propertyTokenSymbol propertyTokenName amount
    let lookups = Constraints.ownPaymentPubKeyHash
    let tx = Constraints.mustPayToPubKey recipient token
    ledgerTx <- submitTxConstraintsWith @Scripts.MintingPolicy lookups tx
    awaitTxConfirmed $ getCardanoTxId ledgerTx
    logInfo @String $ "Transferred " ++ show amount ++ " tokens to " ++ show recipient

mintPolicy :: Integer -> MintingPolicy
mintPolicy amount = Scripts.wrapMintingPolicy $ \_ _ _ -> if amount > 0 then () else error ()

-- Boilerplate code for Plutus Contract
endpoints :: Contract () PropertyTokenSchema Text ()
endpoints = do
    logInfo @String "Property Token Contract started"
    awaitPromise
        (mintToken `select` transferToken)
    where
        mintToken = endpoint @"mint" >>= mkPropertyToken
        transferToken = endpoint @"transfer" >>= \(amount, recipient) -> transferToken amount

mkSchemaDefinitions ''PropertyTokenSchema

mkKnownCurrencies []
```

Key Differences

1. **Language**:
   - **Solidity**: Uses a contract-oriented language specifically designed for Ethereum.
   - **Plutus**: Uses Haskell and the Plutus smart contract framework tailored for Cardano.
2. **Token Management**:
   - **Solidity**: Uses ERC20 standard functions like `mint`, `transfer`, and `approve`.
   - **Plutus**: Uses Plutus-specific constructs like `Value.singleton` and custom minting policies.
3. **Contract Deployment**:
   - **Solidity**: Contracts are deployed and interacted with through transactions on the Ethereum network.
   - **Plutus**: Contracts are written in Haskell and deployed on the Cardano blockchain, involving on-chain validation and custom minting policies.
4. **Ownership and Permissions**:
   - **Solidity**: `Ownable` modifier controls who can call certain functions.
   - **Plutus**: Uses on-chain validation and constraints, and transactions are validated based on the script's logic.
5. **Minting and Transfer**:
   - **Solidity**: Functions are straightforward with checks performed in the contract logic.
   - **Plutus**: Minting and transferring involve building and submitting transactions with specific constraints and policies.
6. **Testing and Verification**:
   - **Solidity**: Typically tested using frameworks like Truffle or Hardhat.
   - **Plutus**: Tested using the Plutus Playground or Cardano's testnet, focusing on Haskell-based testing.

This comparison should give a clear idea of how property fractionalization is approached differently in Ethereum and Cardano ecosystems. Each platform has its unique features and methodologies for handling smart contracts and tokens.

▾ Pages 16

Find a page...

Clone this wiki locally

```
https://github.com/besiwims/plutus-tx-template.wiki.git
```