directory

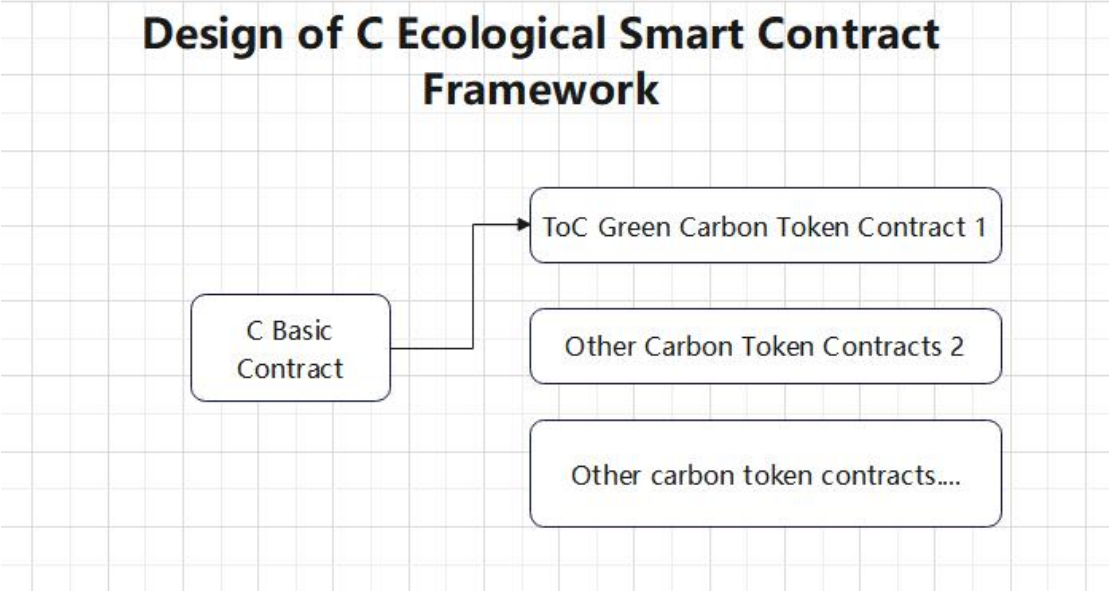
version	Released	content	author
1.00	2024.6.20	Carbon ecological smart contract design	Zoning

Smart contract design on the carbon ecological chain

1. Overview

In order to ensure that the number of tokens on the chain matches the actual carbon rights, it is required that each minting must have a complete minting source in the smart contract, providing query and total traceability, and the carbon ecology is managed by different contracts according to different carbon types, and all carbon token contracts are controlled and managed by the basic contract deployer.

二、Carbon ecological architecture design



Base Contract:

1、Realize the management and operation of the entire carbon ecology

2、Control the number of token types in the carbon ecosystem

3、Provide permission management authorization for various carbon ecological tokens

Different entities will have their own on-chain carbon ecology and have their own independent and controllable C-based contracts, through which they can freely develop various types of carbon credit assets

Ecological Management Strategy:

1、On-chain manager voting mechanism

2、The off-chain threshold signature mechanism manages the private key of a manager

Minting Management Strategy:

1、On-chain manager voting mechanism

2、The off-chain threshold signature mechanism manages the private key of a certain minter

三、Environmental description

Programming language: solidity

Compile Version: 0.8.1

Test Environment: T-chain development network (see the T-link port document for deployment test methods)

Formal Operating Environment: T-chain

四、Definition of the underlying contract

1、Specify the management pool address

The contract publisher only has one chance to operate, and once the management pool address is specified, it will lose the interface operation permission

```
/*
addrs manages address pools
names manages the name of the address
*/
function InitManger(address[] memory addrs,bytes32[] memory names)public returns (bool)
```

Processing logic: 1. The sender must be the contract deployer, 2. The management pool is not initialized

2、Increase managers

To increase the number of managers, it needs to be unanimously approved to actually join

```
/*
    address addrs new admin
    Bytes32 named name
*/
function AddManger(address addrs,bytes32 name)public returns (bool)
```

Processing logic: It is necessary to check whether the sender is the administrator, not to return directly, if so, record the votes of the current manager on the address, and if all pass, add the address to the management pool

3、Delete the manager

To delete a manager, you need to vote unanimously (except for the destination address) to actually delete the manager

```
/*
    address addrs Managers to be removed from
*/
function RemoveManger(address addrs)public returns (bool)
```

Processing logic: It is necessary to check whether the sender is the administrator, not to return directly, if so, record the votes of the current manager of the address, and if all pass,

delete the address from the management pool

4、Query the list of management pools

Returns a list of addresses for the management pool that is currently valid

function ShowManager() public returns(bool,address[],bytes32[])

5、Query the list of management pools to be confirmed

Returns the list of management pool addresses in the current voting stage, including two states: increase (0) and culling (1).

function ShowPendingManager() public returns(bool,address[],bytes32[],byte[])

6、Deploy carbon-type tokens

To increase the carbon type token, it needs to be unanimously approved to actually join

/*

address addrs Contract address of the Carbon token

Bytes32 named :name of the token

Uint type Type of the token

*/

function AddToken(address addrs,bytes32 name,uint token_type)public returns (bool)

Processing logic: It is necessary to check whether the sender is the manager, not to return directly, if so, record the vote of the current manager on the token, and if the vote is unanimously passed, the token address corresponding to the type will be updated

7、Upgrade the carbon-type token contract

To upgrade a carbon-type token, it needs to be unanimously approved to be truly upgraded

/*

address addrs Contract Address of the Carbon Token

Bytes32 named Name of the Token

Uint type Type of the Token

*/

function UpdateToken(address addrs,bytes32 name,uint token_type)public returns (bool)

Processing logic: It is necessary to check whether the sender is the manager, not to return directly, if so, record the vote of the current manager on the token, and if the vote is unanimously passed, the token address corresponding to the type will be updated

8、Inquire about the list of tokens

function ShowToken() public returns(bool,address[],bytes32[],uint[])

Processing logic: Returns the list of existing tokens (including contract address, name, and type, returned as an array)

9、Inquire about the list of tokens to be confirmed

function ShowPendingToken() public returns(bool,address[],bytes32[],uint[],uint[])

Processing logic: Returns the list of existing tokens (including contract address, name, and type, returned as an array)

10、Add minting address

When the minting address is minted on the carbon credit chain, a security check will be carried out, and only the legal address has the minting authority on the type of carbon coin, this interface is to add the legal minting address, and the submission address needs to be unanimously approved by the management pool in order to add the minting address on the corresponding token.

```
/*
    address addrs:              New Convict
    address subcontractaddr     Carbon-type contract address
    Bytes32 name
*/
```

function SetCoinage(address addrs,address subcontractaddr,bytes32 name)public returns (bool)

Processing logic: To detect whether the sender is the address of the management pool, otherwise it will fail, if so, record the vote of the current manager on the carbon type contract address, if the vote is unanimously passed, the address becomes the minting address of the carbon type contract address.

11、Delete the minting address

```
/*
    address addrs:              New Convict
    address subcontractaddr     Carbon-type contract address
*/
```

function RemoveCoinage(address addrs,address subcontractaddr)public returns (bool)

Processing logic: to check whether the sender is a management pool address, otherwise it will fail, if so, record the vote of the current manager on the carbon contract address, and if it is unanimously approved, the address will be cleared from the minting address pool of the carbon contract address.

12、List of minting addresses

function ShowCoinage()public returns(bool,address[],uint[],uint[])

Processing logic: Returns the corresponding minting address pool for each carbon type token

13、A list of minting addresses to be confirmed

function ShowPendingCoinage()public returns(bool,address[],
uint[],uint[],byte[])

Processing logic: Returns the corresponding minting address pool for each carbon type token

五、toC Green Carbon Token Contract Definition

The token is compatible with all interfaces of ERC20 tokens, and is controlled by the constraints of the underlying contract, and the ToC carbon token is initially set to zero.

1、Compatible with all interfaces of ERC20 tokens

2、Minting interface

This is the only source of the toC token token

```
/*
Bytes32 Business         Merchant code
bytes32 place            Venue code
bytes32 dev              Device encoding
bytes32 targetCode           The name of the item
bytes32 targetName       Item code
bytes32 targetMode       The model number of the item
bytes32 targetType       Item type
uint  targetNums         The amount of carbon in the item
address businessAddr     Business address
address targetAddr       Carbon source address
Byte    percent          Percentage
*/
function  Coinage(bytes32 business,bytes32 place,bytes32 dev,bytes32 targetCode,
            bytes32 targetName,bytes32 targetMOde,bytes32 targetType,uint
            targetNums,address businessAddr,address targetAddr,
    Byte    percent)public returns (bool)
```

Processing logic: first through the basic contract, verify whether the sender is the legitimate minter of the token, if it is not directly return failure, otherwise, record the currency source, and the corresponding merchant address and carbon source address, and increase the corresponding number of tokens according to percent

3、Minting source query

function ShowCoinage(uint begintime,uint endtime)public returns()

Processing logic: The return result form is to be determined

4、Minting source statistics

function StatsCoinage(uint begintime,uint endtime)public returns()

Processing logic: The return result form is to be determined

5、Get token information

function Blanceof()public returns(uint,uint)

Return to sender owns the total amount of tokens and contract tokens

6、

六、**Deploy contract tests**

For details about how to use APIs, see

T-Chain - Development Interface 5.0 (Deployment Contract and Contract Execution Chapter)

Development Network API submission address: http://13816813962.gnway.cc:80