Fractional NFT

Solidity Contract Design

V1.0

Red Date Technology

December 2022

Table of Contents

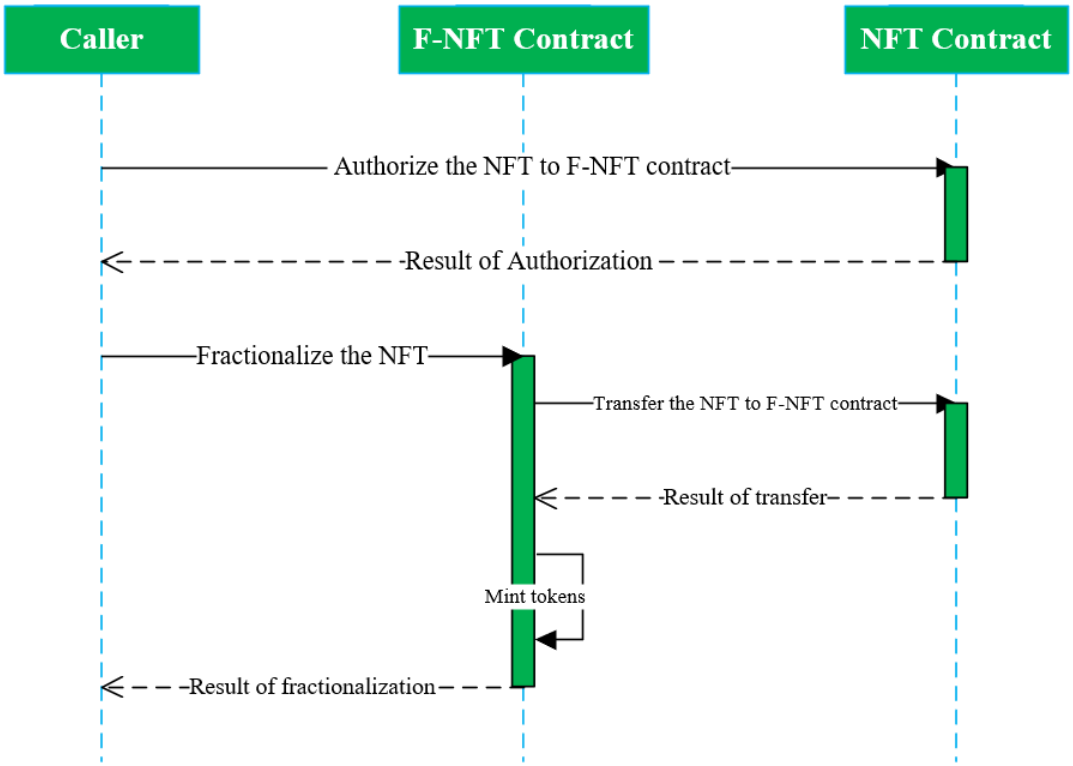
1. Purpose

By calling the Fractional NFT smart contract (F-NFT contract), ERC-721-based NFT works can be fractionalized into multiple ERC-20/ERC-1155 tokens, and their ownership is also split for easy transfer and trading purposes.

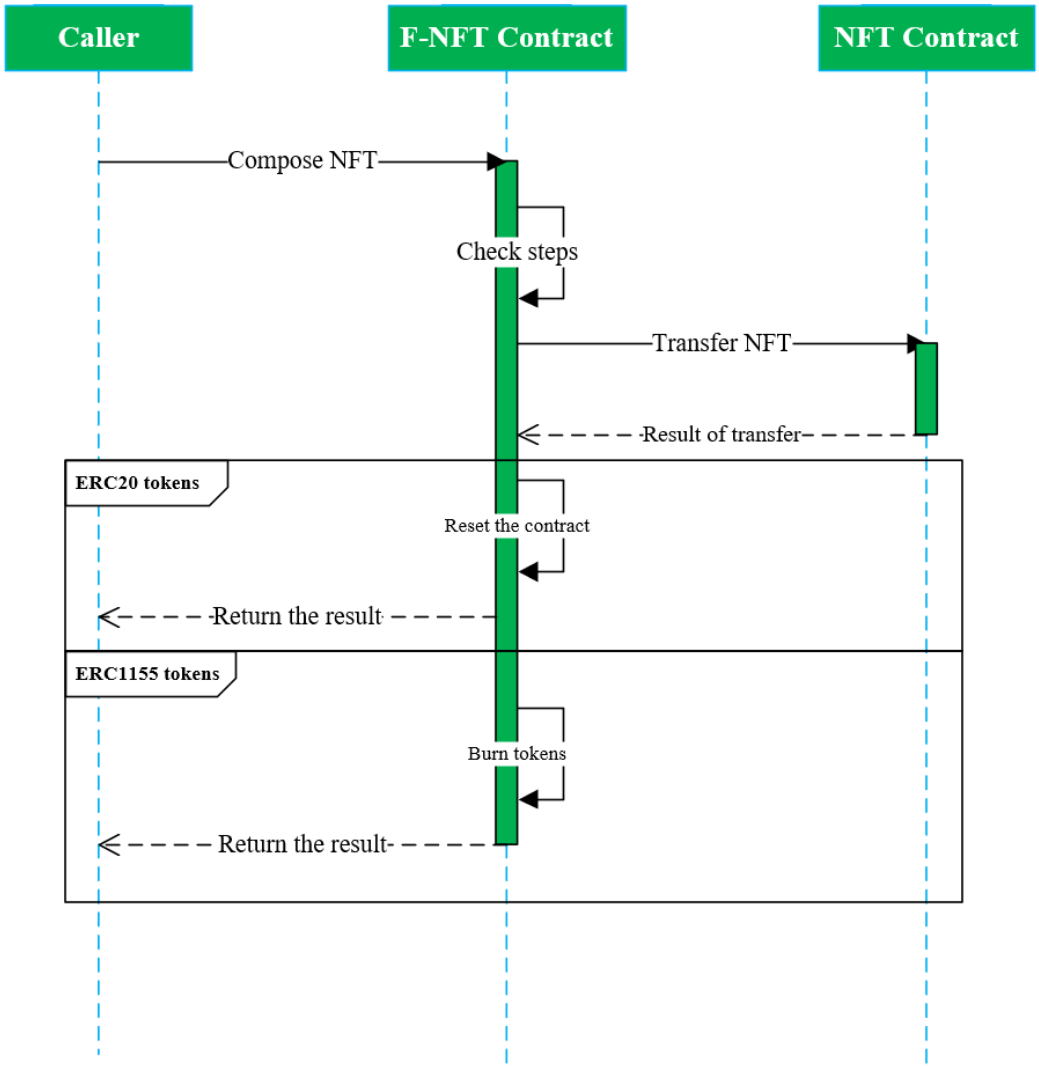
In short, the NFT fractionalization process is equivalent to "asset redistribution", but this process changes the standard of the asset (from ERC-721 to ERC-20/ERC-1155)

2. Overall Design

2.1 NFT Fractionalization Timing Diagram



NFT fractionalization



NFT composition

2.2 Security Design Description

The contract is deployed and initialized to run completely autonomously according to the established logic and does not accept manual management, upgrade, or destruction by any account.

2.3 Contract Upgrade Design

This contract does not allow upgrade.

3. Contract Design

3.1 ERC-20 Standard Fractional NFT Contract

3.1.1 Function Description

The ERC-721 NFT owner authorizes the NFT to this contract and generates ERC-20 tokens. A contract can only support the fractionalization of one NFT, and after the NFT is composed, the contract is reset and can accept new NFT fractionalization operations. The contract contains following roles:

1. Contract deployer: the contract owner. This is the the person who deploys or owns the contract. The ownership of the contract can be transferred to others. The contract owner has the authority to initialize the contract, i.e., to fractionalize other NFTs. Contract initialization can only accept NFTs owned by that owner and transfer the ERC-20 tokens (NFT fractions) to the contract owner.

2. User: All other user accounts that can use the standard methods in ERC-20 contract as well as the compose method in this contract.

3.1.2 Data Structure

➢ Contract parameters

No.	Field name	Field	Type	Remarks
1.	Contract owner	_owner	address	Contract owner
2.	ERC721 contract	_erc721	address	Address of ERC721 contract
3.	Token Id	_tokenId	uint256	Token ID

➢ User account

No.	Field name	Field	Type	Remarks
1.	Account address	key	address	User’s account address

2.	Account balance	value	uint32	Use's account balance

3.1.3 API Definition

3.1.3.1 Contract Initialization

Initialization method of contract deployment

- Input parameter: name, symbol;
- Output parameter: none;
- Method name: recharge;
- Method example: constructor(string memory name_, string memory symbol_);
- Event: none;
- Core logic:
  - Set the name and symbol of ERC-20 contract before deployment.
- Business scenario:
  - Contract initialization

3.1.3.2 NFT Fractionalization

Contract owner calls this method to fractionalize the NFT.

- Input parameter: ERC-721 contract address, token Id, total ERC-20 tokens;
- Output parameter: none;
- Method name: fractional;
- Method example: function fractional (address addr, uint256 tokenId, uint256 amount);
- Event: event Fractional (address indexed erc721, address indexed spender, uint256 tokenId, uint256 amount);
- Core logic
  - Check whether the call is the contract owner;
  - Input the address of ERC-721 contract, transfer the ownership of the NFT to be the address of ERC-20 contract;
  - Check whether the owner of the NFT is the ERC-20 contract;

- Mint the ERC-20 tokens to the contract owner;
- Trigger the event.

3.1.3.3 NFT Composition

The account that owns all the ERC-20 tokens can call this method to compose the NFT and transfer it to the dedicated account, and then reset the contract.

- Input parameter: dedicated account;
- Output parameter: none;
- Method name: compose;
- Method example: compose (address to);
- Event: Compose (address indexed sender, address indexed to);
- Core logic:
  - Check whether the caller owns all the ERC-20 tokens;
  - Transfer the ERC721 NFT to the dedicated account;
  - Trigger the event;
  - Reset the contract.

3.1.3.4 Query NFT

Query the address of the ERC-721 contract that mints the NFT and the NFT token ID.

- Input parameter: none;
- Output parameter: ERC721 contract address, NFT token ID;
- Method name: token;
- Method example: token () returns(address , uint256);
- Event: none;
- Core logic:
  - Return the address of ERC-721 contract and the NFT token ID;
- Business rule: this method is only called after the initialization;

3.1.3.5 Query Name

Return the token name.

- Input parameter: none;
- Output parameter: string;
- Method name: name;
- Method example: function name() public view returns (string);
- Event;
- Core logic:
  - Return the token name;

3.1.3.6 Query Symbol

Return the token symbol.

- Input parameter: none;
- Output parameter: string;
- Method name: symbol;
- Method example: function symbol() public view returns (string);
- Event;
- Core logic:
  - Return the token symbol;

3.1.3.7 Unit of Token

Return the unit of token.

- Input parameter: none;
- Output parameter: uint8;
- Method name: decimals;
- Method example: function decimals() public view returns (uint8);
- Event;
- Core logic:
  - Return the unit of token;

3.1.3.8 Token Quantity

Return the total number of tokens.

- Input parameter: none;
- Output parameter: uint256;
- Method name: totalSupply;
- Method example: function totalSupply() public view returns (uint256);
- Event: none;
- Core logic:
  - Return the total number of tokens.

3.1.3.9 Query the Balance

Return the balance of an account

- Input parameter: account address;
- Output parameter: balance;
- Method name: balanceOf;
- Method example: function balanceOf (address _owner) public view returns (uint256 balance);
- Event;
- Core logic:
  - Get the balance of an account.

3.1.3.10 Transfer Tokens

Transfer tokens to the dedicated account.

- Input parameter: account address, the number of tokens to be transferred;
- Output parameter: transfer result;
- Method name: transfer;
- Method example: function transfer (address _to, uint256 _value) public returns (bool success);
- Event: event Transfer (address indexed _from, address indexed _to, uint256 _value);
- Core logic:
  - Check whether the balance of the sender account is greater than value

- Deduct the value of the sender account;
- Increase the balance of “to” account with the amount of value;

3.1.3.11 Transfer Tokens between Accounts

Transfer the tokens from the authorized account to the dedicated account.

- Input parameter: account that sends tokens, account that receives tokens, the number of transferred tokens;
- Output parameter: transfer result;
- Method name: transferFrom;
- Method example: function transferFrom (address _from, address _to, uint256 _value) public returns (bool success);
- Event: event Transfer (address indexed _from, address indexed _to, uint256 _value);
- Core logic:
  - Check whether from account has been authorized;
  - Check whether the number of authorized tokens is greater the number of tokens to be transferred;
  - Check whether the number of tokens held by from account is greater than the number of tokens to be transferred;
  - Deduct tokens in from account;
  - Increase tokens in to account.

3.1.3.12 Account Authorization

Authorize the account.

- Input parameter: authorized account, the number of authorized tokens;
- Output parameter: result;
- Method name: approve;
- Method example: function approve(address _spender, uint256 _value) public returns (bool success);

- Event: event Approval (address indexed _owner, address indexed _spender, uint256 _value);
- Core logic:
  - Set the number of tokens to be authorized to the account;

3.1.3.13 Query Authorized Token

Return the number of authorized tokens.

- Input parameter: authorizer, authorized account;
- Output parameter: the number of authorized tokens;
- Method name: allowance;
- Method example: function allowance (address _owner, address _spender) public view returns (uint256 remaining);
- Event;
- Core logic:
  - Check and get the number of authorized tokens.

3.2 ERC-1155 Standard Fractional NFT Contract

3.2.1 Function Description

This contract supports fractionalization of multiple NFTs. The generated ERC-1155 tokens corresponding to the NFTs will be destroyed after the NFT is composed.

The contract contains following roles:

1. Contract deployer: contract owner, the person who deploys or owns the contract. Contract owner can be transferred to others.

2. Users: all other general accounts. They can call “fractional” method in the contract to fractionalize the authorized NFTs and get all NFT fractions in the form of ERC-1155 tokens. Also, they can call the ERC-1155 contract standard method and the NFT “compose” method.

3.2.2 Data Structure

No.	Field name	Field	Type	Remarks
1.	TokenId	_tokenIdCounter	Uint256	Self-increment
2.	ERC721 token structure	_erc721	mapping (uint256 => erc721token)	
3.	Relationship between token and ERC721 contract	_tokens	mapping (address=>mapping (uint256 => uint256))	
No.	**Object**	**erc721 token**		
1.	Contract initialization	inited	Bool	
2.	ERC721 contract address	erc721	IERC721	
3.	NFT Token ID	tokenId	Uint256	
4.	Amount	amount	Uint256	

3.2.3 API Definition

3.2.3.1 NFT Fractionalization

The NFT owner calls this method to fractionalize the authorized NFTs.

- Input parameter: ERC-721 contract address, NFT token ID, the number of NFT fractions, data;
- Output parameter: Result of fractionalization
- Method name: fractional;
- Method example: function fractional (address erc721address, uint256 erc721tokenId, uint256 amount, bytes memory data) public {;
- Event: event Fractional (address indexed erc721, address indexed spender, uint256 erctokenId, uint256 tokenId, uint256 amount);
- Core logic:
  - Check whether the ERC-721 contract has authorized this contract address;

- Generate TokenId
- Transfer NFT to the address of this contract;
- Record the NFT information;
- Generate the ERC-1155 token, set sender as the token owner;
- Trigger the event;

3.2.3.2 NFT Composition

The user that holds all ERC-1155 tokens (NFT fractions) can call this contract to compose the NFT.

- Input parameter: receiver's account address, Token ID;
- Output parameter: none;
- Method name: compose;
- Method example: function compose (address to, uint256 tokenId) public;
- Event: event Compose (address indexed sender, uint256 tokenId, address indexed to);
- Core logic:
  - Check whether the number of ERC-1155 tokens held by the sender equals to the total number of ERC-1155 tokens;
  - Transfer the ERC-721 NFT hosted by the contract address to a dedicated address;
  - Trigger the event;
  - Burn all ERC-1155 tokens held by the sender;

3.2.3.3 Query Total Number of Tokens

Query the total number of ERC-1155 tokens by the token ID.

- Input parameter: Token ID;
- Output parameter: uint256;
- Method name: totalSupply;
- Method example: function totalSupply(uint256 tokenId) external view returns (uint256);

- Event: none;
- Core logic:
  - Return the total number of ERC-1155 tokens;

3.2.3.4 Query NFT

Query the NFT token ID by inputting the address of ERC-721 contract and ERC-1155 token ID.

- Input parameter: NFT ERC-721 contract address, tokenId;
- Output parameter: uint256;
- Method name: NFTOf;
- Method example: function NFTOf (address erc721address, uint256 erc721tokenId) external view returns(uint256);
- Event: none;
- Core logic:
  - Query the NFT token ID by inputting the address of ERC-721 contract and ERC-1155 token ID.

3.2.3.5 Query Token

Query the address of ERC-721 contract and the NFT token ID by inputting the ERC-1155 token ID.

- Input parameter: TokenId;
- Output parameter: address, uint256;
- Method name: tokenOf;
- Method example: function tokenOf (uint256 tokenId) external view returns(address,uint256);
- Event: none;
- Core logic:
  - Query the NFT information by inputting the ERC-1155 token ID.