

DoNFT Protocol Description

Introduction

DoNFT is a decentralized protocol that allows

- modification of NFT with the help of different algorithms, for example neural networks. These algorithms can be created by anyone and incorporated in our service.
- bundling a few NFTs to a single NFT.

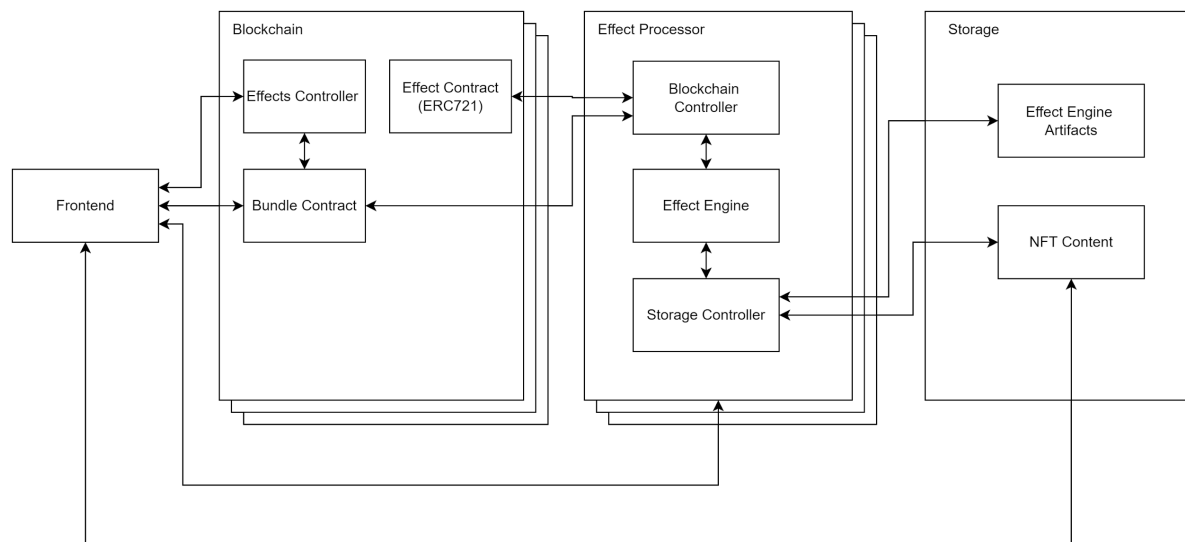
Terms

Bundling - a process of combining several NFTs into 1 resulting NFT

Unbundling - a process of separating bundled NFT into original ones.

Architecture

For other terms we would first refer to a scheme below that describes how DoNFT works



There are four main blocks in the DoNFT protocol:

1) Blockchain block

Effect Contract - ERC721 (or analogue on another blockchain) standart smart contract that performs management of tokens with modifier information (content for modification for example).

Effect Controller - smart contract that manages a list of Effect Contracts and the process of connection Effect Contracts with Effect Processors.

Structure of record

```
{
    "id":<string>,
    "modifiersContract":<address>,    //address of contract with modifiers NFTs
    "owner":<address>,
    "serverUrl": <string>,              //url of EffectProcessor
    "originalContract":<address>,      //address of contract with collection of NFTs to
                                        //which the modifier can be applied
    "type":<string>
}
```

Bundle Contract - a smart contract that performs logic of bundling and unbundling NFT tokens and their management. Original NFT and modifier NFT (or bundled NFTs in case without modification) are locked into smart contract and after unbundling are received back to the users wallet.

Structure of bundle token

```
{
    "tokenId":<string>,
    "metadataUrl":<string>,
    "owner":<address>,
    "bundles": <List of ERC721 tokens>,
    "approvals":<List of address>,
    "royalty":<List of address and percents>
}
```

Additional methods

function bundleWithTokenURI(NFT[] _tokens, string tokenURI) returns(uint256)

Create new bundle token.

function unbundle(uint256 _tokenId)

Burn bundle token and return original NFTs to owner.

function removeNFTsFromBundle(uint256 _tokenId, NFT[] _tokens)

Remove tokens (sent to owner) from existing bundle token.

function addNFTsToBundle(uint256 tokenId, NFT[] _tokens)

Add tokens to existing bundle token.

function bundleTokensOf(uint256 _tokenId) public view returns (NFT[] memory)

Return list of NFTs which are stored in bundle token.

function allowToChanges(uint256 tokenId, address editor) returns(bool)

Give permission for changing bundle token to another address.

2) Effect processor block

Effect processor is a service that performs logic of new content creation. It contains 3 parts:

Blockchain Controller performs reading and writing information on and from a blockchain.

Effect Engine - an algorithm (neural networks, graph networks, machine learning algorithms or any other kinds of algorithms) that can modify the content of an original NFT and generates new content with a modification applied

Storage Controller performs interactions with different types of storages (for example IPFS)

When the Effect processor is up Effect Engine Artifacts are uploaded from the Storage.

Typical API

GET /api/healthcheck

used to check the status of the service

POST api/effects/applyEffect

Body

```
{
  "original": {
    "contract": <address>,
    "tokenId": <string>,
    "contentUrl": <url>
  },
  "modifier": {
    "contract": <address>,
    "tokenId": <string>,
    "contentUrl": <url>
  },
  "sender": <address>
}
```

Response

Url of file with new content

3) Storage block

Effect Engine Artifacts - objects that are needed for Effect Engine work (for example ML models and their weights) are uploaded from storage in the moment of launching the Effect Processor.

NFT Content - all the data that is stored in NFT (metadata, images e.t.c)

4) Frontend block

Frontend - dApp for logic of interaction with Smart Contracts, Effect Processors and Storages. it includes all the following skills

- shows a list of all NFT tokens in a user's wallet
- performs UI a logic of interaction with Bundle Contract (methods bundle, unbundle and some other typical NFT interaction methods)
- interacts with Storage
- interacts with Effect Processors
- interacts with Effect Controller

Implementation

Nowadays MVP of protocol supports EVM-like blockchains Ethereum and Polygon. We are working on deploying on Harmony, Solana and NEAR. To connect a wallet user has to choose a chain and a wallet type.

As a source of user's NFTs we are using the Rarible protocol (if it supports selected blockchain) and direct reading from blockchain (for this purpose we implemented "manage collection" logic).

Effect processor is implemented as a python service inside a docker container.

IPFS is used as a Storage.

User Stories

1) NFT Modification

- [Frontend]
User chooses an original NFT that is to be modified and a modifier NFT that was bought on any marketplace.
- [Frontend] -> [Effect Controller]
Finding Effect Processor which is related with the chosen modifier NFT (frontend finds an Effect Processor URI in Effect Controller).
- [Frontend] -> [Effect Processor]
Contents of the original NFT and the modifier NFT are sent to the Effect Processor.

- [Effect Processor]
Storage controller downloads artifacts which are required for Effect Engine's work and wasn't downloaded before.
- [Effect Processor]
Blockchain Controller checks that chosen NFTs are on the user's address
- [Effect Processor] -> [Frontend]
Effect Engine generates new content and returns it to Frontend.
- [Frontend] -> [Storage]
New content is uploaded to Storage along with generated metadata (also users can add yours data) for bundled NFT.
- [Frontend] -> [Bundle Contract]
Calling the "Bundle" method of Bundle Contract.
- [Bundle Contract]
Transfer ownerships of original NFT and modifier NFT to new NFT. Assigning a metadata URL to a new NFT. Minting new NFT.

2) Bundling several NFTs

- [Frontend]
Users choose all the NFTs they want to bundle together without any modification.
- [Frontend] -> [Storage]
Generating metadata (also users can add yours data) for bundled NFTs and uploading it to Storage.
- [Frontend] -> [Bundle Contract]
Calling the "Bundle" method of Bundle Contract.
- [Bundle Contract]
Transfer ownerships of selected NFTs to new NFT. Assigning a metadata URL to a new NFT. Minting new NFT.

3) Unbundling NFTs.

- [Frontend]
Choosing previously bundled NFT
- [Frontend] -> [Bundle Contract]
Calling the "Unbundle" method of Bundle Contract
- [Bundle Contract]
Transfer ownerships of bundled NFTs to the previous owner. Burning bundle NFT.

Monetization

- fee of using Bundle Contract
- selling Modifier NFTs
- hosting Effect processor
- outsourcing of creating special Effect processor
- hosting Storage