YeeZ

Protodao Contract Development

Version:1.0

YeezTech
https://yeez.tech
Yulong Zeng

2023.11

Contents

1 Terms

- SubNodes: Basic Unit of SemiOS Governance. A SubNodes belongs to a SeedNodes.
- SeedNodes: A collection of SubNodes, representing a governance series.
- Builder: The work uploader. Multiple works may share a same builder. A builder belongs to a SubNodes.
- Starter:SubNodes Creator. When creating SubNodes, an existing SeedNodes can be specified, or a new SeedNodes can be created for affiliation.
- Work: Users (callder Builders) can upload pictures to SemiOS website for a SubNodes, which are available for purchase by other users. When a work is purchased it generates an (on-chain) NFT. This purchase process is also called minting an NFT, and the buyer is also referred to as the Minter.
- NFT: Non-Fungible Token, is facilitated by the ERC721 contract which provides a standard framework. In SemiOS, a NFT is assosiate with aSubNodes and a Builder. A NFT identifier includes a ERC721 address and a tokenId.
- Minter: The user that mints a NFT, who is also the NFT's owner.
- ERC20: Fungible Token.
- OutputToken: An endogenous ERC20 token associated with a SeedNodes, as assets generated by this governance series.
- InputToken: An exogenous token associated with a SeedNodes, as external assets attracted by the SeedNodes, usually uses ETH or other known ERC20-tokens.
- NodesAssetPool: A fund pool associated with a SubNodes.
- RedeemPool: A special fund pool associated with a SeedNodes, providing the function of exchanging OutputTokens for InputTokens.
- Treasury: A special fund pool associated with a SeedNodes, used to store initial OutputTokens.
- RoyaltyFee: Fee charged on secondary trades of an NFT.
- RoyaltySplitter: A contract for distributing Royalty Fees.
- ProtocolFeePool: An Address to collect fees for the SemiOS project.

- Pass Card: Reserved NFTs under a SubNodes.
- Uri: A string type, usually containing a link. SubNodes, NFT, and Builder each have their corresponding Uri.
- Fixed Price/Unified Fixed Price: Refers to the use of a fixed price for minting NFTs.
- Block: Assosiate to a SubNodes, also known as a block, it is a time window unit for reward distribution and price change strategies.
- Active Block: A block in which any NFT has been minted in the accosiate SubNodes.
- MintWindowDuration: The duration time of one round, counted by the block number.
- MintFee (NFT price): The token amount to pay for minting an NFT, usually uses InputToken.
- Block Reward: Assets flowing out of a NodesAssetPool in a round.
- TopUp Mode: A special SubNodes mode for fundraising purposes.
- Lottery Mode: A special SubNodes mode where non-active rounds' earnings accumulate.
- Infinite Mode: A special SubNodes mode with infinite rounds, where each round's total block reward equals the total assets of the NodesAssetPool.
- OutputToken Payment Mode: A special SubNodes mode where MintFees are paid by OutputToken.
- Selectable InputToken Mode: A special SeedNodes mode where the ETH used for MintFee is replaced with an ERC20 token chosen by the user.
- Import OutputToken Mode: A special SeedNodes mode where OutputToken is an existing ERC20 token imported by the creator, instead of SeedNodes generated ERC20 token.
- TopUp Account: An account used by users for investment in TopUp mode.
- Maker: The owner of a TopUp account.

2 SemiOS Introduction

The SemiOS, previously known as DAO For Art and later ProtoDao, has been ongoing over two years since its first commit on June 15, 2022. It is an integrated product that encompasses NFT uploading, minting, governance, and reward distribution. This section will provide a brief overview of the project's functionalities based on version 1.3. Features from new versions are also introduced. Starting from version 1.6, the project name was changed to SemiOS.

2.1 SubNodes

A SubNodes is the governance unit used in this project. Users can create a SubNodes and engage in activities such as NFT minting, reward claim, and parameter modification based on this SubNodes. The user who create the SubNodes is called Starter, also known as SubNodes owner, who has authority to set SubNodes attributes and is one of the roles that receive profits. Each SubNodes is identified in the contract by a unique *daoId* (of types bytes32).

2.2 SubNodes Associated Contracts

2.2.1 ERC721 Contract

When each SubNodes is created, an ERC721 contract is also established and associated with it. This contract provides functionalities related to the minting of NFTs.

2.2.2 NodesAssetPool Contract

Each SubNodes, upon creation, is associated with a fund pool contract known as the NodesAssetPool. This contract is used to store the InputTokens and OutputTokens of the SubNodes.

2.2.3 RoyaltySplitter Contract

After an NFT is minted and begins circulating on the secondary market, major exchanges transfer a portion of the transaction fees from secondary sales to a specific address, as stated in the ERC721 contract. The information includes the fee percentage and the recipient address. In SemiOS, the fee percentage set by the Starter is restricted within system-defined maximum and minimum limits. This part of the fee is known as the RoyaltyFee. Note that this action is not mandatory, but exchanges generally adhere to this rule voluntarily. This rule does not apply to peer-to-peer transfers between users.

The SemiOS system wants the RoyaltyFee to be directed to two addresses: one being the NodesRedeemPool and the other the project's address, known as the ProtocolFeePool. However, the standard for RoyaltyFee in the ERC721 only allows for one declared recipient address. Therefore, a Royalty Splitter contract is introduced to facilitate the automatic distribution of RoyaltyFees.

Each SubNodes, upon creation, sets up a Royalty Splitter contract, which is declared in the ERC721 contract as the recipient of the RoyaltyFee. When this contract receives ETH, it automatically distributes the received ETH to specific split addresses according to predefined proportions, through the fallback() function.

- If the SubNodes has not enabled the Import OutputToken mode, the addresses for the split payment are the ProtocolFeePool and the NodesRedeemPool.
- If the SubNodes has enabled the Import OutputToken mode, the addresses for the split payment are the ProtocolFeePool and the NodesAssetPool, as the import ERC20 does not support the Redeem function.
- Currently, the ProtocolFeePool collects a 2.5% share of the RoyaltyFee.

The RoyaltySplitter contract also supports the distribution of ERC20 tokens, but this must be triggered manually or via a Chainlink Keeper.

2.2.4 GrantAssetPoolNFT Contract

In version 1.6, a new feature was added involving an ERC721 contract created with each SubNodes. When a user makes a grant to the NodesAssetPool, this contract mints an NFT as proof of the donation.

2.3 SeedNodes

SeedNodes is a collection of SubNodes which belong to the same governance series. When creating a SubNodes, an existing SeedNodes should be specified for affiliation.

If no association is made, a new SeedNodes is created for affiliation. In this case, the SubNodes Starter is the owner of the SeedNodes.

2.4 SeedNodes Associated Contracts

SeedNodes Associated Contracts means that all associate SubNodes share the same contract.

2.4.1 ERC20 Contract

Each SeedNodes is associated with an ERC20 contract created concurrently with its inception. This contract serves as the governance token for the series, known as the OutputToken. All SubNodes within the series share the same OutputToken.

New in 1.3: Upon creation, a SeedNodes can opt to associate with an existing ERC20 address, known as the Import OutputToken mode. In this mode, there is no need to create a new ERC20 contract.

2.4.2 NodesRedeemPool Contract

Each SeedNodes is also associated with an additional fund pool contract known as the NodesRedeemPool upon its creation,. This contract is used to store a portion of the InputToken revenues from all SubNodes in the series and provides users with the functionality to exchange their OutputTokens for InputTokens.

2.4.3 Treasury Contract

This feature, added in 1.6, involves each SeedNodes creating an additional fund pool upon its establishments. Additionally, a preset total number of DaoTokens is generated for the system. The default total number is set at 1 billion.

2.4.4 GrantTreasuryNFT Contract

Additionally, a similar feature was introduced in version 1.6 for each MainDao. When a user makes a grant to the Treasury, this ERC721 contract mints an NFT as a certificate of the transaction.

In conclution, SeedNodes and SubNodes have following relationships:

- All SubNodes shares the same OutputToken (as well as InputToken) as its associated SeedNodes, but SubNodes still maintains its own ERC721 contract. As of version 1.3, whether a SubNodes adopts Import OutputToken mode is consistent with its associated SeedNodes.
- All SubNodes shares the same NodesRedeemPool with its associated SeedNodes, but still possesses its own NodesAssetPool.
- Compared to SubNodes starters, SeedNodes creators have more permissions, mainly in setting treasury assets and rights. These rights will be represented as NFTs starting from version 1.6.

2.5 Builder

Builder represents the uploader of work and is identified in the contract by a unique *canvasId* (of type bytes32). Each work belongs to a Builder, and each Builder is associated with a SubNodes. The *canvasId* is generated off-chain and must be provided to the contract at the time of minting the NFT. Builder is also one of the beneficiaries.

When a SubNodes is created, the first Builder is automatically generated, which is the SubNodes Starter. All reserved NFTs (pass cards) of the SubNodes belong to this Builder.

In the current version, the NFT pricing strategy is associated with the Builder. Future versions simplify to s̃n.

2.6 SubNodes Block

The SubNodes Block serves as the minimum calculation period for NFT pricing strategies and reward distribution strategies. Many rules are based on rounds. For example, rewards distributed within a block are calculated and disbursed when the round is over. Unique rules apply to NFT pricing strategies within a round, which will be discussed further below.

Each SubNodes can specify a parameter called Block duration, which represents the duration of a block counted by Ethereum block number. Each time the MintWindowDuration is reset, the current round is recalculated.

If any works are minted within a SubNodes during a block, that block is considered active.

Each SubNodes has an attribute called remaining blocks. With each passing block, SubNodes' remaining blocks decrease by one. The remaining rounds of a SubNodes can be set by owner. When a SubNodes' remaining blocks reach zero, the SubNodes enters a dead state, during which no NFT can be minted. The owner can revive a dead SubNodes by setting a non-zero remaining blocks (or switching to infinite mode), after which all data on active blocks and pricing strategies are reset and recalculated.

2.7 Work and NFT

Builders upload works to the website. Before being minted in the contract, it has nothing to do with the blockchain. When minters chooses to purchase a work, an on-chain NFT is minted. Each NFT is assigned to a specific Builder (*canvasId*) and SubNodes (*daoId*).

Users can mint an NFT under an existing Builder, or choose to create a new Builder and mint an NFT belongs to it simultaneously, depending on whether the specified *canvasId* already exists.

Once an NFT is minted, its owner becomes the initiator of the minting transaction, and it can subsequently be traded on the secondary market.

2.7.1 NFT Price

When the NFT minting function is called, a payment called MintFee in InputToken is required (added in version 1.4: in OutputTokenpayment mode, OutputToken is used for MintFee) as the price for minting the NFT. Typically, the InputToken is ETH or well-known third-party ERC-20 tokens such as USDT and USDC.

The pricing for NFTs can follow these models:

- Unified Fixed Price

  After a SubNodes activates this mode, its owner will set a fixed price. In this case, all NFTs under this SubNodes are minted at this price, and minting does not require a signature. This mode cannot be switched.

  The unified fixed price can be set to zero initially. If so, it can not be set to a non-zero value. If initial unified fixed price is not zero, owner can set it to any non-zero value at any time.

- Fixed Price

When the unified fixed price is disabled, the Builder can designate a fixed price for his work by providing a signature. When users mint NFT for purchase this work, they must upload the Builder's signature on the price, thus ensuring the price data cannot be falsified.

- Non-fixed price (system pricing) mode

  When the unified fixed price is disabled, the Builder can designate his work as system pricing item by providing a signature. When users mint NFT for purchase this work, they must upload the canvasCreator's signature on system pricing mode, thus ensuring the price mode cannot be falsified.

Note: The attribute that asset used for minting NFT has been changed from ETH to a user-defined InputToken is introduced in version 1.7, which is SeedNodes attribute.

2.7.2 System Pricing Strategy

The system pricing strategy is quite complex and supports the introduction of new templates. However, generally, it follows these rules:

- Each SubNodes has a floor price to be set. Typically, the first NFT minted in a SubNodes is at this floor price. The floor price cannot be zero.
- If an NFT under a certain Builder is minted multiple times within a round, each minting price will be twice the price of the previous minting (w.r.t. exponential pricing template, the factor of 2 can be set to other coefficients) or increased by a fixed amount. (w.r.t. linear pricing template) (hereafter, multiplying or dividing by 2 is used to represent price step changes).
- When a block ends, the minting price of a NFT is halved, but the price reduction stops at the base price.
- If the prices of all Builders' NFTs under the SubNodes are reduced to the floor price, the minting price for the NFTs in the next round will be half the floor price (this part is not affected by the template). During this period, if any work under the SubNodes is minted, the prices of all NFTs in that SubNodes revert to the floor price.

The factor of 2 mentioned above can be defined by templates into other pricing methods. Currently, the project supports exponential and linear pricing templates, and it is possible to set the pricing coefficients. For the exponential pricing template, price changes by multiplying or dividing by the pricing coefficient; for the linear pricing template, price changes by adding or subtracting the pricing coefficient; in all cases, the price cannot drop below the floor price.

The pricing strategy and floor price can be set by owner.

2.7.3 Pass Card

When a SubNodes is created, a certain number of NFTs can be reserved as special NFTs. Their tokenIds range from 1 to the reserved quantity, and they have a specific URI prefix. The price of a Pass card, when the unified fixed price is not activated, is the floor price of the DAO. Pass cards belong to the Builder that is automatically created at the time of the SubNodes creation.

The tokenIds of all non-special NFTs start from the reserved quantity + 1.

2.8 MintFee Distribution Rules

When a user (minter) mints an NFT, they must pay InputToken for MintFee. MintFee will be distributed to the following address for the first time, where the distribution ratio is set by the owner.

- ProtocolFeePool, the fund pool of the SemiOs project. The proportion is specified by the SemiOS admin. In version 1.7 it is changed to 0.
- Builder, the uploader of the work (NFT mint for purchase).
- NodesRedeemPool, the redemption pool (OutputToken–>InputToken) for the SeedNodes.
- NodesAssetPool, the affiliated fund pool of the SubNodes.

Except the ProtocolFeePool ratio, other ratios can be set with two different values for system pricing and non-system pricing repsecitvely.

In TopUp mode, no diversion occurs: all MintFee go directly into the Minter's TopUp account.

2.9 Block Reward Issuance Rules

When an NFT is minted within a SubNodes for the first time in a block, the block becomes active and triggers a block reward issuance. This means that all assets of the NodesAssetPool, including OutputToken and InputToken, are distributed according to the following rules:

2.9.1 Total Block Issuance Determination

Whether the assets are InputToken or OutputToken, the total amount of block reward issuance is always determined according to the following rules:

If the SubNodes has not activated the Lottery mode:

$$\text{Amount issued for current block} = \frac{\text{Total amount in NodesAssetPool}}{\text{Remaining blocks}}$$

The sources of assets in NodesAssetPool can be from the redirection of MintFee, the block rewards from other SubNodes, or assets directly injected by users. Notably, if the asset is OutputToken, after version 1.6, the SeedNodes owner can inject treasury funds into the NodesAssetPool.

Remaining blocks include the current block.

If the SubNodes has activated the Lottery mode:

$$X = Y * \frac{Z}{M + Z - 1}$$

- X: Amount issued for current block.
- Y: Total Asset in NodesAssetPool.
- Z: Lottery rounds.
- M: Remaining rounds.

The lottery periods Z is the current block serial number minus the serial number of the last active block (if there is no active period, then it is 0), which is the total number of block for the accumulation of rewards (including the current period).

If the SubNodes activates the Infinite mode, then the total amount issued X will be the total assets.

2.9.2 Distribution Rules

After determining the total amount for current block reward issurance, the assets are allocated for the following purposes, according to ratios set by SubNodes owner (Note: different ratios can be set for InputToken and OutputToken separately):

- Transfer to the NodesAssetPool of several other SubNodes under the same SeedNodes, meaning that this SubNodes can designate several SubNodes (within the same SeedNodes) as the recipient SubNodes.
- Transfer to the NodesRedeemPool, which can only be set for InputToken.
- Used for internal distribution within this round.
- The portion not used in block reward and remains in the NodesAssetPool of this SubNodes, called reserved.

If the SubNodes activates the TopUp mode, then 100% of the total OutputToken block issuance is used for internal reward distribution, and all InputToken in the NodesAsset-Pool(under normal circumstances, neither MintFee nor block rewards enter the asset pool in TopUp mode) are transferred into the NodesRedeemPool.

2.10 Internal Distribution Rules

The portion of the block rewards used for internal distribution will be allocated to the following roles according to the set distribution coefficients and weights:

- The NFT minters in this round.
- The Builders of the NFTs minted in this round.
- The Starter (SubNodes creator).
- The ProtocolFeePool, in version 1.7 this ratio is set to be 0.

Among these, the distribution coefficient is a SubNodes parameter, and roles include the Minter, Builder, and Starter. Coefficients can be set separately for these roles as well as for InputToken and OutputToken. For a single NFT minting, the added weight for a role is the amount of the MintFee that flows into the NodesAssetPool multiplied by the role coefficient (when the role is Minter or Builder, only the case that user mints the NFT or is the Builder of the NFT is included); the total weight increase is the amount of the MintFee that flows into the NodesAssetPool. It means that an NFT's contribution is related to its MintFee that distributes to the NodesAssetPool.

The final distribution ratio is the user's weight divided by the total weight.

$$X = Y * \frac{\Sigma_{\mathtt{Z}}(C * F)}{\Sigma_R F}$$

- X: User's claimable reward quantity for the block.
- Y: Total internal distribution for the block.
- Z: NFTs minted in the block.
- C: Role coefficient.
- R: All NFTs in the round.
- F: Amount of MintFee flowing into the NodesAssetPool.

If the SubNodes activates the TopUp mode, then by default, all internal distributions are allocated to Minter (i.e., the Minter role coefficient is 100%, and the formula remains unchanged). Moreover, the aforementioned amount of minting fees flowing into the NodesAssetPool(definition of F) is changed to the total MintFee, because under TopUp mode, InputToken does not flow into the NodesAssetPool.

If the SubNodes' global fixed price is set to zero, then the aforementioned F is regarded as 1, meaning that all NFTs contribute the same.

2.10.1 Reward Collection

The rewards from internal distribution for the current block can only be collected after the block ends. The contract supports a one-click collection where users can collect earnings from all roles, across all SubNodes, for all blocks where rewards have not yet been claimed.

The diagram below summarizes the basic process of asset flow within the DAO.

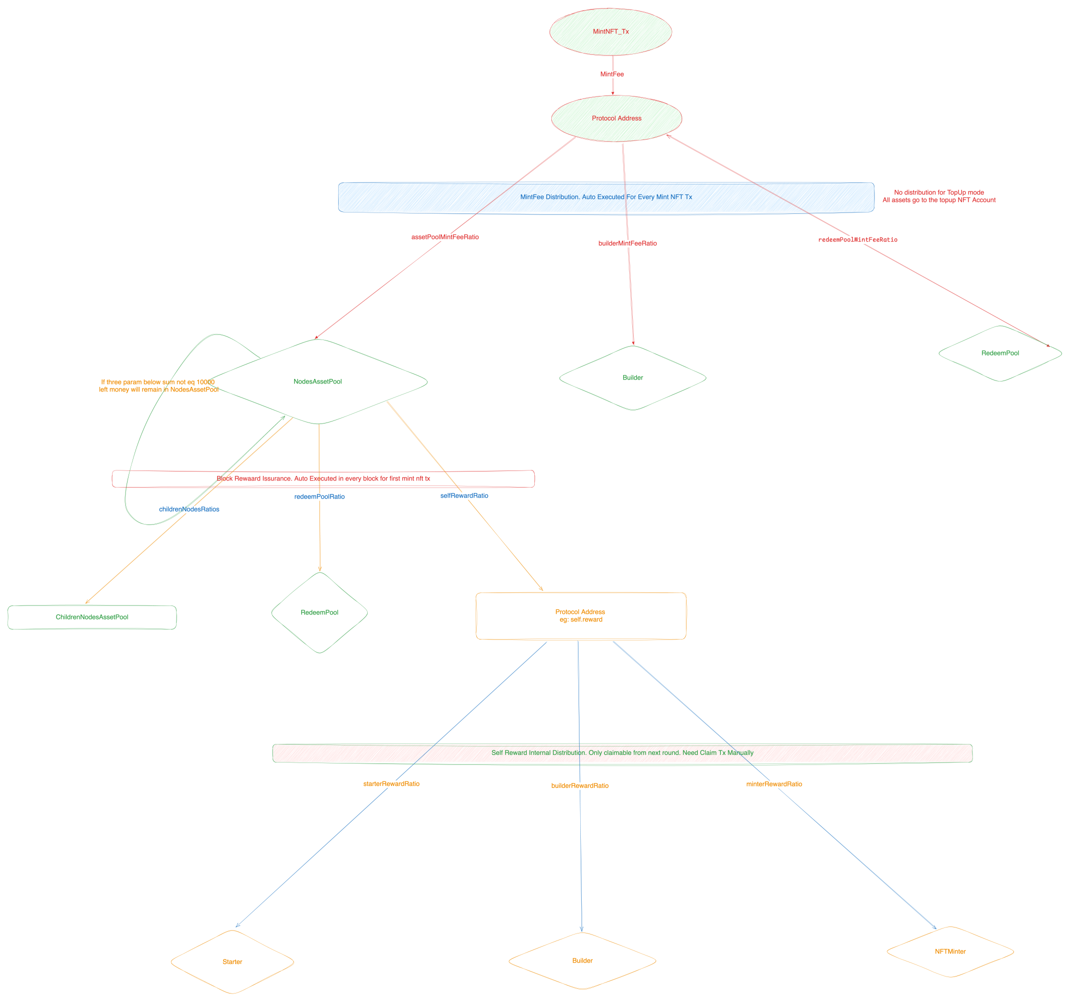
MintNFT_Tx
MintFee
Protocol Address
MintFee Distribution. Auto Executed For Every Mint NFT Tx
No distribution for TopUp mode
All assets go to the topup NFT Account
assetPoolMintFeeRatio
builderMintFeeRatio
redeemPoolMintFeeRatio
If three param below sum not eq 10000
left money will remain in NodesAssetPool
NodesAssetPool
Builder
RedeemPool
Block Rewaard Issurance. Auto Executed in every block for first mint nft tx
childrenNodesRatios
redeemPoolRatio
selfRewardRatio
ChildrenNodesAssetPool
RedeemPool
Protocol Address
eg: self.reward
Self Reward Internal Distribution. Only claimable from next round. Need Claim Tx Manually
starterRewardRatio
builderRewardRatio
minterRewardRatio
Starter
Builder
NFTMinter

Figure 1: SubNodes asset flow

2.11 Redeem Rule

Users can exchange the received OutputToken for InputToken. The redemption rules do not apply to Import OutputTokenMode.

The circulating amount of OutputToken is defined as the total supply of OutputToken minus the OutputToken balance in NodesAssetPools of all series SubNodes, then minus the OutputToken balance in the NodesRedeemPool.

After version 1.6, the circulating amount of OutputToken need to be further reduced by the OutputToken balance in the treasury.

The amount of redeemed InputToken equals the number of OutputToken used for redemption multiplied by the InputToken balance in the NodesRedeemPool, then divided by the circulating amount of OutputToken.

2.12 Granting the Asset Pool

This is a new logic added in version 1.6.

Users can make grants to the NodesAssetPool or the treasury.

- If the choice is to make a grant to the NodesAssetPool, users can opt to use funds from their own wallet or from the treasury. Only the owner the SeedNodes can use the latter option. After making the payment, the user receives a “grantAssetPoolNft” as proof of payment.
- If the choice is to make a grant to the treasury, users must use funds from their own wallet. After making the payment, the user receives a ”grantTreasuryNft” as proof of payment.

2.13 Permission Judgement

2.13.1 NFT Minting Permission Judgement

A SubNodes has complex settings for determining whether a user has the permission to mint NFTs. The overall process follows these steps:

- BlockMintCap (SubNodes attribute): the maximum number of NFTs that can be minted each block.
- Blacklist (SubNodes attribute): if a user’s address is on this blacklist, their mint request is denied.
- NodesMintCap (SubNodes attribute): SubNodes global mint limit. If the Whitelist attribute is not enabled, this requires that the total number of mints of a user cannot exceed this limit. If met, minting is allowed.
- Whitelist (SubNodes attribute): If activated, it is divided into minter Whitelist, NFT whitelist and a ERC721 Whitelist.
- If a user is on the Minter Whitelist:
- UserMintcap[] (SubNodes attribute): each whitelist user’s mint limit (which can be infinite). In this case, the total number of mints a user makes cannot exceed the UserMintcap[the user] value. If met, minting is allowed. Note: Merkle tree system is used to determine whether a user is in minter Whitelist.

- If the user is not on the minter whitelist, go to NFT Whitelist:
- NFT Whitelist (SubNodes attribute, newly added in version 1.7) : a list of NFTs, each with a minting limit (which can be infinite). If a user owns any NFTs in the NFT Whitelist, the total number of mints the user makes cannot exceed the minting limit associate with the NFT. If met, minting is allowed. Note: If the user owns multiple NFTs in the NFT Whitelist, the smallest limit is used for evaluation.
- If the user is not on the NFT whitelist, go to ERC721 Whitelist:
- ERC721 Whitelist (SubNodes attribute): a lists of ERC721 contract addresses, each with a minting limit (which can be infinite). If a user owns any NFTs from ERC721 contracts on the ERC721 Whitelist, the total number of mints the user makes cannot exceed the minting limit associate with the ERC721 contract; if met, minting is allowed; otherwise, it is not allowed. Note: If the user owns multiple NFTs from the ERC721 whitelist, the smallest limit is used for evaluation.
- If none of the above conditions are met, the mint request is denied.

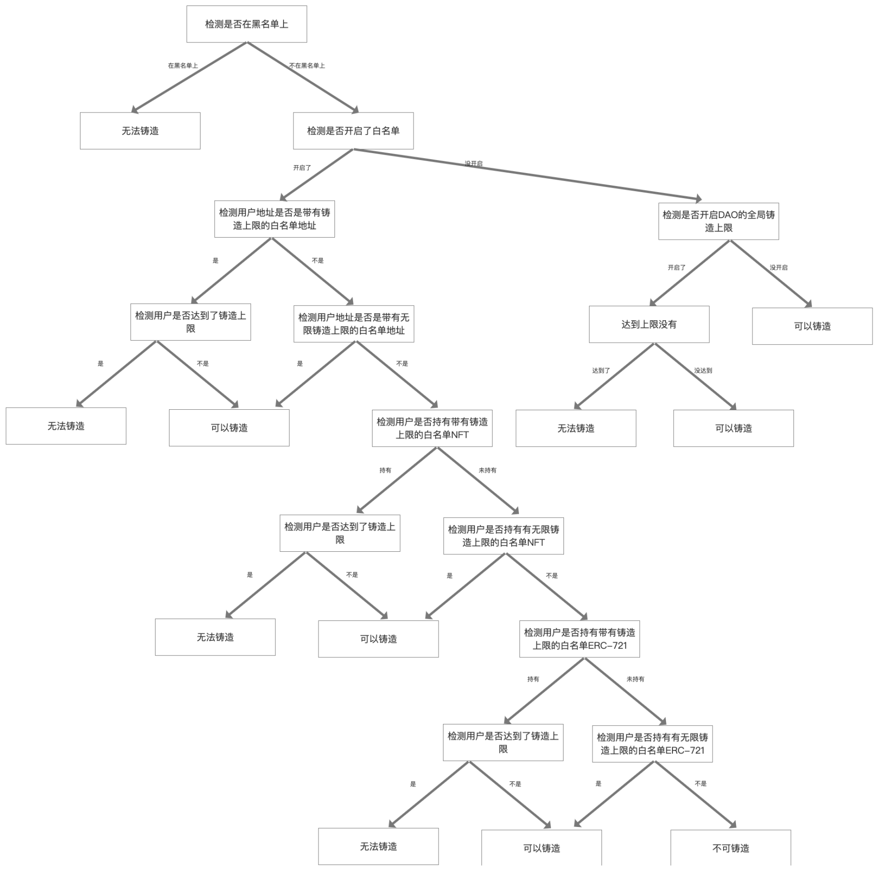
检测是否在黑名单上
在黑名单上
不在黑名单上
无法铸造
检测是否开启了白名单
开启了
没开启
检测用户地址是否是带有铸造上限的白名单地址
检测是否开启DAO的全局铸造上限
是
不是
开启了
没开启
检测用户是否达到了铸造上限
检测用户地址是否是带有无限铸造上限的白名单地址
达到上限没有
可以铸造
是
不是
是
不是
达到了
没达到
无法铸造
可以铸造
检测用户是否持有带有铸造上限的白名单NFT
无法铸造
可以铸造
持有
未持有
检测用户是否达到了铸造上限
检测用户是否持有有无限铸造上限的白名单NFT
是
不是
是
不是
无法铸造
可以铸造
检测用户是否持有带有铸造上限的白名单ERC-721
持有
未持有
检测用户是否达到了铸造上限
检测用户是否持有有无限铸造上限的白名单ERC-721
是
不是
是
不是
无法铸造
可以铸造
不可铸造

Figure 2: minter permission

2.13.2 Builder Permission Assessment

When a new Builder is to be created (identified by minting), the SubNodes assesses the permissions of the Builder's address as follows:

- Builder Blacklist (SubNodes attribute): If the Builder's address is on this blacklist, the minting is denied.
- Builder Whitelist, Builder NFT Whitelist and Builder ERC721 WhiteList (SubNodes attribute). If all attributes are disabled, the minting is allowed.
- If the Builder is on the Builder Whitelist, the minting is allowed. Note: Merkle tree system is used to determine whether a user is in Builder Whitelist.
- If the Builder owns any NFTs in Builder NFT Whitelist (a list of NFTs) or from ERC721 contracts in Builder ERC721 WhiteList, the minting is allowed.
- If none of these criteria are met, the minting is denied.

2.14 DAO Rights Tied to NFT

This is a feature proposed for version 1.6, which has not yet been implemented in the frontend or backend. Specifically, when a DAO is created, its rights can be divided as follows:

- The "Edit Information" right. This right is unrelated to the contract itself and is used to modify parameters displayed on the frontend. The contract only records the owner of this right.
- The right to"Edit On-chain Parameters". This includes modifying on-chain properties of the DAO, such as changing the remaining blocks of the DAO, entering or exiting infinite mode, setting the global NFT minting cap for the DAO, setting the NFT minting cap per round within the DAO, setting the floor price of the DAO, setting a global fixed price for the DAO, adjusting the NFT pricing strategy of the DAO, and setting all receipent DAOs and their correspoinding block reward distribution ratios, including RedeemPool ratio(ETH only), internal distribution ratio, and reserved ratio, as well as setting the distribution coefficients for various roles with respect to internal distribution.
- The right to "Edit Strategies". This involves modifying various blacklists and whitelists of the DAO.

- Revenue rights, which designate the profits that can be collected by someone in the DaoCreator role.

Additionally, if a MainDao is created, it will have the following three rights:

- The right to "Edit SubDao Information". This right is unrelated to the contract and is used to modify parameters displayed on the frontend. This right allows for the editing of information for any DAO within the series.
- The "Treasury Transfer Asset" right. This right allows for the transfer of treasury funds into the DaoAssetPool of any series DAO.
- The right to set the "TopUp Unlock Distribution Ratio", which includes setting a default ratio and setting ratios for one or more series DAOs.

All the aforementioned rights are tied to NFTs, meaning that only the owner of the corresponding NFT possesses these rights (in previous versions, all these rights were owned by the DaoCreator). Additionally, the owner of each right has the option to bind this right to another NFT. When a DAO is initially created, the DAO's ERC721 contract will create an NFT with tokenId 0, and all four rights (seven for a MainDao) are bound to this NFT, which is owned by the DaoCreator. Subsequently, the DaoCreator can distribute different rights to different NFTs by binding them accordingly.

2.15 Supplementary Explanation for Special Mode

2.15.1 Infinite Mode

- In Infinite Mode, the total amount of block reward in each round equals all the assets in the DaoAssetPool.
- After turning Infinite Mode on or off, the data of the pricing strategy remains unchanged, and the round does not reset. For instance, if the price of the last non-fixed price NFT under a certain canvas in the current round is 0.01 ETH, then even if Infinite Mode is toggled, the minting price for the next NFT under that canvas would be 0.02 ETH (assuming an exponential pricing strategy with a pricing coefficient of 2).
- After turning off Infinite Mode, the lottery rounds in the lottery mode are reset. Note: Turning on Infinite Mode is unrelated to the lottery round in the lottery mode, because Infinite Mode distributes all rewards.

- When Infinite Mode is disabled, it is necessary to re-specify the remaining number of rounds for the DAO.

2.15.2 TopUp Mode

New Concept: Savings Account, introduced for DAOs operating in TopUp mode. A user's savings account is shared across the same series of DAOs. A DAO in TopUp mode can be thought of as a fund where LPs inject capital by minting artworks. Using money from the savings account is akin to the fund making an investment, and successful investments unlock DaoTokens as rewards proportionally.

In a DAO operating under TopUp mode, the ETH for MintFee is transferred to their savings account at the end of the current round. The total block reward for each round in a TopUp DAO remains calculated as before, but theoretically, there would be no ETH in the DaoAssetPool. The distribution rule for DaoTokens is 100% for internal allocation. The DaoTokens that users can claim are transferred to their savings account at the end of the round.

Using the savings account: When users mint any NFT in a non-TopUp DAO within the same series, the required ETH is first deducted from their savings account. Each portion of ETH used from the savings account unlocks a corresponding proportion of DaoTokens directly into the user's wallet. For example, if a user's savings account contains 1 ETH and 2000 DaoTokens, spending 0.3 ETH to mint an NFT would result in 600 DaoTokens being transferred into the user's wallet (unlocking 30%).

Minting NFTs under a TopUp DAO does not allow the use of the savings account.

Version 1.4 update: When users use DaoTokens from their savings account to mint artworks (only in DAOs that have enabled DaoToken payment mode), the corresponding proportion of ETH is unlocked (implementing a refund logic).

Abstract Behavior of the Savings Account

- Minting an NFT in a TopUp mode DAO is equivalent to depositing money into the savings account (either ETH or InputToken); the source of funds can only be the user's wallet (cannot use money from another savings account).

- Minting an NFT in a non-TopUp mode DAO is equivalent to withdrawing/spending money from the savings account. Spending ETH can unlock DaoTokens in the saving account, and spending DaoTokens can unlock ETH in the savings account (unlocking means directly transferring to the user's wallet).

Savings Account Bound to NFT This is a new logic added in version 1.5. The savings account is bound to an NFT's owner rather than a user's address, thus allowing the savings account to circulate as an asset package in the secondary market. Only the owner of the bound NFT can use the funds in the corresponding savings account.

When a user deposits money into the savings account by minting artwork in a TopUp mode DAO, they must specify an NFT identifier (including ERC721 address and tokenId) to indicate that the money is to be deposited into the savings account associated with that NFT. If the ERC721 address field of the NFT identifier specified by the user is zero address, then the NFT minted by the user during this transaction becomes the NFT bound to the new savings account, essentially creating a new savings account. When a user spends money from the savings account, they can specify at most one NFT address, indicating that they are spending money from the savings account bound to that NFT, with the requirement that the user is the owner of this NFT. If the ERC721 address field of the specified NFT address is the zero address, it indicates that the user is not using any savings account.

NFT Locking This is new logic added in version 1.5. The owner of an NFT bound to a savings account can lock the NFT, specifying the block number until which it is locked. During the lock period, the savings account bound to the NFT cannot be used for spending, and if the NFT identifier is specified during minting, assets from the user's wallet will be used directly. Deposits into the savings account are still allowed during the lock period.

Savings Account Unlocking Diversion This feature was added in version 1.6. In previous versions, 100% of the unlocked assets from a savings account were transferred directly into the user's wallet. Now, a portion of the unlocked ETH/DaoToken must be transferred to the RedeemPool/DaoAssetPool. The proportion of this transfer is predetermined by the creator of the MainDao and is not specified in the DAO creation parameters.

The creator of the MainDao can set a default proportion for the unlocking diversion, which every newly created SubDao will follow. Additionally, the MainDao creator can set different proportions across multiple SubDaos within the series.

2.15.3 DaoToken Payment Mode

In this mode, the DAO requires users to mint NFTs using DaoTokens. This requires prior approval, or the DaoToken must support the permit method. When minting NFT using the savings account in this mode, a corresponding proportion of ETH is unlocked.

DaoToken payment mode and TopUp mode cannot be enabled simultaneously.

2.15.4 Self-Selected InputToken Mode

This is a new feature introduced in version 1.7, associated with the MainDao. It means that all DAOs in the series use the same InputToken. When creating a DAO, users can choose an ERC20 token as the InputToken to replace ETH needed for minting works. Similarly, the rewards distributed for block production will also be in the form of the selected InputToken. Additionally, the use and unlocking of ETH in TopUp savings accounts, as well as assets obtained through Redeem, will also switch from ETH to the chosen InputToken.

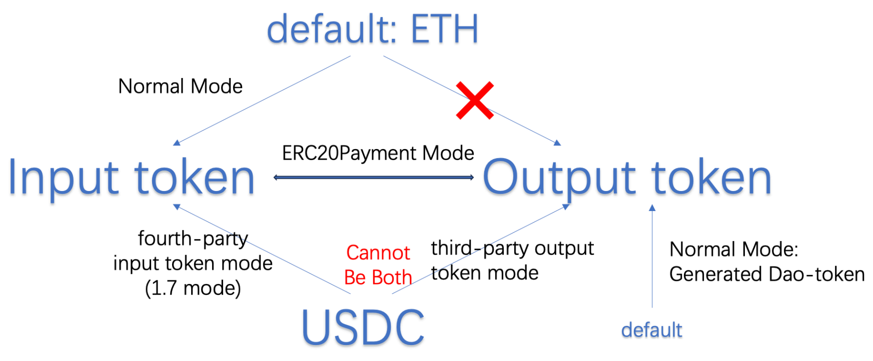
default: ETH
Normal Mode
ERC20Payment Mode
Input token
Output token
fourth-party input token mode (1.7 mode)
Cannot Be Both
third-party output token mode
Normal Mode: Generated Dao-token
USDC
default

Figure 3: different DAO modes

2.16 DAO Default Template

The DAO is currently created with the following default parameters:

- The DAO is currently created with the following default parameters:
- Floor price set at 0.01 ETH.
- Global fixed price set at 0.01 ETH.
- Reserved pass card quantity set at 1000.

- Pricing strategy set to exponential mode, with a pricing coefficient of 2.
- Initially mint 50,000,000 DaoTokens for the DaoAssetPool. After version 1.6, this quantity is set to 0, and the MainDao creator needs to make an additional transaction to fund the DaoAssetPool via the treasury.
- Start time is the current block, with a round duration of one day.
- DaoCreator is added to the minter whitelist, with a minting limit of 5.
- Associated ERC721 is added to the 721 list, with a minting limit of 5 for its holders. (Fission effect)
- All special modes are turned off.
- Other parameters are to be added...