Cubics Blockchain Handbook

Documentation for Modules and Programs

Version 0.8.1
23rd December, 2021
https://cubics.com

Address

Explanation

Every wallet, token, and pool has an address, which can be used to transfer funds to it. Addresses are automatically generated for new tokens and pools, while wallets generate a public key upon creation, derived from its private key. Addresses are simply public keys that can receive and send transfers.

To receive transfers, let the sender know your public address, and they will be able to send tokens to that address. To send transfers, visit the Cubics network app. Then, connect your wallet by clicking on the top right button. Select "new transfer" and insert the recipients' address into the to-field, alongside specifying the token and amount value.

Intro

An address represents an account on Cubics and is either a token, pool or private wallet. Every public key is automatically an address that can receive funds. Users can use their address to represent a profile (e.g. an entity, a collection, an account)

Use Cases:
Token and pool addresses
Personal wallet address (equivalent to a public key)

Functionality

When a user creates a managed or self-managed wallet, a public key is automatically generated, serving as a public wallet address. This address is public and can receive permissionless transfers from anyone.

Allowance

Explanation

Allowances allow other people to spend your fungible tokens, up to a specified amount. You can create an allowance by visiting the Cubics network app and connecting your wallet by clicking on the "connect" button in the top right corner. After you've connected your wallet, you can select it and choose "new allowance". Here, you can simply define the spender, the fungible token, and the amount which the spender-address should be able to spend on your behalf.

To spend an allowance that has been granted to you, connect your wallet and select "new transfer" under wallet actions. Fill in the required transfer fields such as to-address, token, and amount. At the bottom of the form, you will find an optional from-field, which you can use to specify the grantee address. This should be the public address of the person or entity who created an allowance for you. By specifying a from-address, your transfer will be sent on their behalf and will withdraw tokens from their wallet and send them to the specified recipient.

Intro

An allowance can be given to another address, which is then allowed to spend a particular fungible token amount on behalf of the creator.

Use Case: Allowing someone to spend funds on your behalf

Functionality

A user sets an allowance, granting another spender the right to spend a particular fungible token for a certain amount. The spender then can create transfers by specifying the from address to be the address that granted the allowance. The allowance creator can increase and decrease their given allowances at any time. The spender can make transfers until the creator's allowance is depleted or adjusted to zero.

Methods

Update Allowance

Allows a spender to spend your tokens for the specified amount.

Requirements: `Token balance`
Outputs: `Allowance to spend`
Inputs:
`Spender` - Public address of the spender
`Token` - Fungible-token address
`Amount` - Allowance amount

Balance

Explanation

A balance on Cubics represents an amount of fungible tokens that an address holds. Balances are debited and credited when someone makes a transfer with a fungible token. When you transfer a token amount to another address, your token balance will be debited, while their token balance will be credited, or vice versa if they send tokens to you.

Claim

Explanation

Claims represent a promise or contract that is made between two parties. The issuing party issues a claim to someone who then holds the claim and has the right to redeem that claim based on the terms specified by the claim issuer. Terms for a claim can be the length of time, amount, or even a custom-written agreement.

When a person or entity issues a claim, the fulfillment of the claim, even upon redemption and valid terms, is not guaranteed. Any person who issues a claim to someone is held publicly accountable with their integrity and trust, however, they might fail to fulfill the promise made.

On the other hand, claims issued by pools that execute verifiable contracts are always guaranteed to resolve successfully, as long as the claim conditions are met, and the pool owns sufficient tokens to resolve the claim.

To create a claim, visit the Cubics network app, connect your wallet by clicking on the "connect" button on the top right and selecting "new claim". You will be able to assign an owner for the claim, setting an associated claim token as well as claim amounts. You will be able to specify when the claim should be able to unlock or when it should expire.

Most commonly, claims represent tickets for something that you contribute to a pool. For example, you might submit a vote to a voting pool or a bid to an auction pool. Alternatively, you might deposit tokens to a swap-liquidity or staking pool or participate in a lottery pool. In all of these cases, you receive a claim, which enables you to redeem it at a later point in time, based on the pools' resolution mechanism.

Intro

Claims represent a promise or contract that is made between two parties. A claim creator issues a claim to someone representing an option that can be redeemed based on specified terms such as timeframe or amount. The claim owner can redeem the claim once the condition of the claim is met. The fulfillment of

promises made by claims is not guaranteed and is dependent on the issuers' integrity and trust. However, if a pool with a verifiable program issues the claim, the claim resolution can be guaranteed through code.

Use Cases:
Claims between two users (credit claims, ownership claims, contracts, promises)
Claims between a pool and a user (liquidity claim, deposit claim, vote claim, lottery ticket claim, etc.)

Functionality

A claim creator creates a claim based on a specific agreement and issues it to a user who has the right to claim once the conditions are met. Most frequently, pools use claims as a ticket to the user, who can redeem the ticket. Created claims can be updated by the creator at any time until resolution. Claim creators can also mark claims as resolved, at which point the claim becomes settled and cannot be updated any longer. Owned claims can be transferred as long as they are not frozen.

Methods

Create

Enables a pool or user to issue a claim to someone.

Requirements: `Claim token balance/ownership`
Outputs: `Claim assigned to specified owner`
Inputs:
`Owner` - Holder who receives the claim
`Token` - Claimable token
`TokenAmount` - Claimable token amount
`CubicsAmount` - Claimable Cubics amount
`Expires` - Expiry date for the claim
`Unlocks` - Date at which the claim becomes claimable
`Frozen` - Forbids claim transfers, if set to true
`Category` - Claim category
`Meta` - Custom metadata as a JSON object
`Answer` - Claimed answer
`Number` - Claimed number

Transfer

Enables the claim owner to transfer claim ownership.

Requirements: `Unfrozen claim`
Outputs: `None`
Inputs:

`Claim` - Claim hash
`To` - To address

Update

Enables the claim creator to update specific claim values.

Requirements: `Claim creator`
Outputs: `None`
Inputs:
`Claim` - Claim hash
`TokenAmount` - Claimable token amount
`CubicsAmount` - Claimable Cubics amount
`Expires` - Expiry date for the claim
`Unlocks` - Date at which the claim becomes claimable
`Frozen` - Forbids claim transfers, if set to true
`Category` - Claim category
`Meta` - Custom metadata as a JSON object
`Answer` - Claimed answer
`Number` - Claimed number

Resolve

Enables the claim creator to close a claim and tag it as resolved.

Requirements: `Claim creator`
Outputs: `None`
Inputs:
`Claim` - Claim hash

Pool

Explanation

Pools are autonomous programs that execute a set of verifiable instructions. Pools are generic wrappers for tokens and enable decentralized functionalities such as auctions, launches, lending, locks, loots, lotteries, royalties, staking, swaps, and votes. Pools can hold tokens, make transfers, and create claims based on their underlying program logic. In short, pools can do everything you can do with a Cubics wallet, except that pools are verifiable and trustless.

To create a pool, visit the Cubics network app, click on connect wallet on the top right and select "new pool". There, you will be able to specify a program, a pool token, and configure the details of your pool as required by your selected program. Programs usually provide detailed documentation, which explains individual configuration values and how these affect the pool life cycle. Once a pool is created, other users can interact with it, according to the selected program.

Intro

Pools are autonomous accounts that execute a set of programmed instructions. Pools are generic wrappers for assets to enable decentralized functionalities such as auctions, launches, lending, locks, loots, lotteries, royalties, staking, swaps, and votes. Pools can hold tokens, make transfers, and create claims based on their underlying and specified program chosen at pool creation.

Use Cases: Reusable autonomous wrappers which execute programmable instructions (equivalent to smart contracts), enabling DeFi functionalities like auctions, launches, lending, locks, loots, lotteries, royalties, staking, swaps, and votes

Functionality

A user creates a pool for a particular token while specifying the desired pool program. This program can be specified as auction (creating an NFT auction), launch (launch-pads for new tokens), lending (allowing others to borrow tokens against CUBIC collateral), locks (locking/vesting tokens), loot (randomized token drops), lottery (token lotteries), royalty (rewards for NFTs), swap (decentralized exchange), vote (voting/betting mechanisms) or any custom program as written in Cubics's Turing-incomplete programming language.

During pool creation, the user might configure the pool using predefined parameters and modify how program instructions are executed (for example, by specifying a minAmount, the pool might only accept transfers with a certain amount).

Methods

Create

Create a new pool that executes instructions based on the specified program for a specified token.

Requirements: `Token balance/ownership`
Outputs: `Pool for the specified token`
Inputs:
`Token` - Token for the pool
`Program` - Pool program id
`Name` - Name of the pool
`Description` - Description of the pool
`Mechanism` - Mechanism, as required by specific programs
`Candidates` - List of candidates
`Rate` - Rate between 0-1
`Percentage` - Percentage, between 0-1
`Number` - Custom number as required by specific programs
`Answers` - List of answers
`Meta` - Custom metadata as a JSON object
`Expires` - Datetime of expiry
`MinAmount` - Min. amount for transfers to the pool
`MaxAmount` - Max. amount for transfers to the pool
`MinTime` - Min. time of claims created by the pool
`MaxTime` - Max. time of claims created by the pool
`TransfersLimit` - Max. number of transfers to the pool
`ClaimsLimit` - Max. number of claims by the pool
`TokenTarget` - Token target balance of the pool
`CubicsTarget` - Cubics target balance of the pool
`TokenLimit` - Token limit balance of the pool
`CubicsLimit` - Cubics limit balance of the pool

Profile

Explanation

Profiles enable users to customize addresses with values like name, description, metadata, previews, and other info. By using profiles, users can turn their public address into a profile representing their NFT collection or any type of account.

To turn your regular address into a profile with custom information, visit the Cubics network app and connect your wallet by clicking "connect wallet" in the top right corner. Click on your wallet and select "view address" to open your address page. On the address page, you will see an update button, which will present you with the option to update all your profile info.

Intro

A user can customize their address profile with name, description, links, image previews, and other info. This enables users to represent entities, collections, and accounts.

Use Cases:
Custom profiles (such as an address representing an NFT collection)

Functionality

When a user creates a managed or self-managed wallet, a public key is automatically generated, serving as a public wallet address. This address is public and can receive permissionless transfers from anyone. A user can update their connected wallet address to represent a profile or a collection.

Methods

Update Profile

Update an address profile by adding custom info such as name, preview image, etc.

Requirements: `Cubics wallet`
Outputs: `NFT representing profile data`
Inputs:
`Name` - A name for the address
`Description` - A description for the address
`Links` - Profile links
`Meta` - Profile data as a JSON object
`Preview` - Preview image for the address
`Category` - Category of the profile

Token

Explanation

Tokens can represent fungible or non-fungible assets. Fungible tokens are divisible tokens and allow fractional ownership of a project. In contrast, non-fungible tokens represent ownership of a non-divisible asset, such as an image, a domain, an avatar, or things in physical or virtual reality.

To create a token, visit the Cubics network app and click on "connect wallet" on the top right. Click on your connected wallet and select "new token". Here, you will be able to fill in token name, description, and values such as symbol, supply, or reserve for fungible tokens and category, or owner for non-fungible tokens.

After you create a token, it will be minted to your address or to the specified owner address if the token is an NFT. For every fungible token, an automatic swap pool is created in the background, which enables any token holder to swap tokens for CUBIC and vice versa.

Intro

Tokens are fungible (divisible) or non-fungible assets. Fungible tokens are similar to shares and represent fractional ownership of something. Non-fungible tokens represent objects via object URL (IPFS, S3, etc) or content hash (e.g. MD5 of bytes content).

Use Cases:
Fungible tokens (representing fractional ownership of something)
Non-fungible tokens (representing objects via URL or content hash in physical and virtual worlds)

Functionality

A user can mint fungible or non-fungible tokens. Fungible tokens represent fractional shares and have a symbol, supply, and optionally a reserve, which can be used to mint additional tokens at a later point in time. Non-fungible tokens represent objects in physical and virtual spaces by referencing an object URL (e.g. IPFS link) or a content hash. When fungible tokens are minted, a decentralized swap pool is created automatically, enabling the decentralized exchange of tokens.

Methods

Create

Create a fungible or non-fungible token.

Requirements: `None`
Outputs: `Funigble or non-fungible token`

Inputs:
`Name` - Name of the token
`Description` - Description of the token
`Links` - Links for the token
`Meta` - Custom metadata as a JSON object
`Preview` - Preview image for the token, e.g. icon for FT, image for NFT
`Symbol` - Symbol for FT
`Supply` - Supply for FT
`Reserve` - Reserve for FT, mintable at a later point in time
`Whole` - Does not allow fractionalization if true `Owner` - Owner for NFT
`Object` - Object representing NFT
`Content` - Content representing NFT
`Mime` - Mime of NFT object
`Frozen` - Disables transfer of NFT, if true
`Category` - NFT category

Update

Allows the creator to update a fungible or non-fungible token.

Requirements: `Token creator`
Outputs: `None`
Inputs:
`Token` - Token address
`Name` - Update name
`Description` - Updated description
`Links` - Updated links
`Meta` - Updated meta
`Preview` - Updated image preview
`Mime` - Updated object mime
`Frozen` - Frozen NFT status
`Category` - Updated NFT category

Mint

Allows the creator of a fungible token to mint additional tokens from its unallocated reserve.

Requirements: `Token creator`
Outputs: `Token transfer from factory to creator`
Inputs:
`Token` - Token address
`Amount` - Amount to be minted

Transaction

Explanation

Cubics transactions are blocks of instructions that are executed by Cubics validator nodes. Instructions can specify multiple functions and enable batch actions such as minting multiple tokens or sending multiple transfers in one transaction. Transactions have a fee associated with them, based on the number of functions calls, reads, and writes, that each instruction executes.

After a transaction is submitted, it is broadcasted to all validator nodes that process the transaction and return the computed result to the coordinator node. After consensus has been established, a transaction is considered confirmed and is stored permanently on the Cubics blockchain. Cubics transactions have an approximate one-second finality time with an average cost below a tenth of a cent.

Transactions are automatically created for any action you might perform on the Cubics network app. If you send a transfer, for example, it is automatically wrapped and submitted as a transaction by the Cubics client.

Transfer

Explanation

A transfer enables anyone to transfer a token, whether fungible or non-fungible, to someone else's wallet. Fungible token transfers always credit and debit balances with an amount, while non-fungible token transfers simply change the ownership address of the NFT.

To create a transfer, visit the Cubics network app and click on "connect wallet" in the top right corner. Afterward, you will be able to select "new transfer" under wallet actions and submit transfer details such as recipient, token, and optionally a token amount if you wish to transfer a fungible token. Transfers can include a message for the recipient, or specify a from-address to make use of an allowance.

Intro

A transfer allows the transfer of fungible and non-fungible tokens between addresses.

Use Cases:
Transferring an amount of a fungible token
Transferring a non-fungible token

Functionality

A user with a certain fungible-token balance can transfer all or parts of the balance to another address. Additionally, a user who owns a particular non-fungible token can create a transfer to assign ownership to another address.

Methods

Create

Create a new fungible or non-fungible token transfer.

Requirements: `Token balance/ownership`
Outputs: `Token transfer from sender to recipient`
Inputs:
`To` - To address for the transfer
`Token` - Token to be transferred
`Amount` - Transfer amount for FT, empty if NFT
`From` - From address in case of allowance
`Message` - Custom transfer message

Auction

Explanation

An auction is a program that enables NFT owners to create an auction for their NFT. Once the NFT auction is created, other users can bid on the NFT or buy it instantly at the limit price. An auction is considered successful if the target bid is reached at expiry. Successful auctions transfer the NFT to the highest bidder and the funds to the auction creator, and vice versa for unsuccessful auctions.

To create an auction, visit the Cubics network app and click on "connect wallet" in the top right corner. Click on your wallet and select "new pool" under wallet actions. You will be able to specify the program and choose the NFT that you want to sell. Furthermore, you can add Cubics target and limit values and set an expiration date.

Once an auction pool is created, the creator must deposit the associated NFT to it. Any other user can then bid on the NFT, as long as the bid is higher than the previous highest bid. To bid on an auction, visit the auction pool and click on "bid", and enter a CUBIC amount for your desired bid. You will receive the NFT from the auction pool after expiry if you remain the highest bidder or if your bid is the limit bid, or you will receive back your CUBIC if someone outbids you.

Intro

Auctions allow users to offer non-fungible tokens for sale. Participants can submit CUBIC-bids until the auction expires or the CUBIC limit is reached. Auction programs automatically handle the resolution by transferring the highest bid and NFT to their new or previous owners (depending on the auction status).

Use Cases:
Auctioning an NFT that a user owns in exchange for CUBIC

Functionality

The NFT owner creates an auction pool and sets a CUBIC target (minimum bid to consider the auction successful), a limit value (instant-buy bid), and a time of expiration. The NFT owner then deposits the NFT to the auction pool, and participants can place bids to the auction pool. When a participant places a bid higher than the previous bid, the pool automatically returns the earlier bid to the previous leader.

The auction outcome is determined when the auction expires or when the limit bid is reached before the expiration time. An auction is considered successful when the limit bid is reached before expiration, or when the target bid is reached at the time of expiration. Auction pools charge a commission of 2.5% for every concluded auction.

Upon success/failure of the auction, any Cubics user can call the resolve/cancel method of the auction pool, which then handles the asset transfers to the new or old owners. On auction failure (target bid not reached on expiry), the NFT is returned to the pool creator, and the highest bid is returned to the highest bidder. On auction success, the NFT is transferred to the highest bidder, and the highest bid is transferred to the auction-pool creator.

Methods

Transfer

Transfer a CUBIC bid to the auction pool.

Requirements: `Bid higher than previous bid`
Outputs: `Bid transfer from sender to pool`, `Return transfer from pool to previous highest bidder`
Inputs:
`Amount` - Bid amount

Resolve

Resolve a successfully concluded auction pool.

Requirements: `Pool closed or expired and bid >= baseTarget`, `Pool active and bid = baseLimit`
Outputs: `NFT transfer from pool to highest bidder`, `CUBIC transfer from pool to pool creator`
Inputs: `None`

Cancel

Cancel an unsuccessful auction pool.

Requirements: `Pool closed or expired and bid < baseTarget`
Outputs: `NFT transfer from pool to pool creator`, `CUBIC transfer from pool to highest bidder`
Inputs: `None`

Create

Create an auction pool for a particular NFT.

Requirements: `Pool creator`, `Pool token ownership`
Outputs: `Auction pool`
Inputs:
`Token` - NFT address
`CubicsTarget` - Minimum bid for auction to be considered successful upon expiry
`CubicsLimit` - Maximum bid, equivalent to instant-buy bid

`TransfersLimit` - Maximum limit of transfers
`Expires` - Date and time of expiry

Deposit

Deposit an NFT to the auction pool.

Requirements: `Pool creator`, `Pool token ownership`
Outputs: `NFT transfer from sender to pool`
Inputs:
`Unlocks` - Datetime when NFT claim can be unlocked and withdrawn

Close

Closes an auction before expiry. Allows anyone to call cancel or resolve methods immediately.

Requirements: `Pool creator`
Outputs: `None`
Inputs: `None`

Launch

Explanation

Launch programs allow creators to raise funds for a project by offering fungible tokens in exchange for CUBIC. After a launch has been created, users can participate in it until the launch completes and resolves. Afterward, users can claim the project tokens if the launch was successful, or their initial CUBIC contribution if the launch didn't reach the minimum fundraising target.

To participate in a launch, visit the launch pool on the Cubics network app and click on "participate". You will be able to specify a CUBIC amount that you would like to contribute, in exchange for tokens that the pool promotes. After the launch completes, you will be able to claim your tokens through the same pool page.

To create a launch pool, visit the Cubics network app, connect your wallet and click on "new pool" under wallet actions. You will be able to set your project token, the fundraising limit, target, as well as other configurations. Afterward, you can deposit tokens to the pool which will determine the launch rate. Keep in mind that half of the launch tokens are allocated for participants, while the other half is allocated for the swap pool liquidity of the associated token.

Intro

Launch pools allow creators to raise funds for a project by offering fungible tokens representing the project in exchange for CUBIC. Launch pools have a fixed rate of raising funds and a target (minimum) and limit (maximum) CUBIC amount. A successful launch pool (target amount reached before closure/expiry) automatically handles the liquidity deposit to the fungible token swap pool. Anyone who participated in the launch pool can then claim fungible tokens or recover their initial contribution if the launch didn't raise the target amount.

Use Case: Raise funds for a project and automatically distribute raised funds to liquidity and/or creators.

Functionality

A fungible-token owner creates a launch pool for a fungible token and shares it with users who can participate by contributing CUBIC tokens. Pool creators can decide how many tokens they want to offer and how much they want to raise as a minimum and maximum amount. Once a launch expires or closes, participants can claim back either tokens (upon success) or their CUBIC contribution (upon failure). Since the pool automatically handles the liquidity deposit, successfully resolved launches are automatically tradeable at the token swap pool.

Methods

Transfer

Transfer CUBIC to the launch pool to participate.

Requirements: `Active pool`
Outputs: `CUBIC transfer from sender to pool`, `Transfer claim from pool to sender`
Inputs:
`Amount` - CUBIC contribution amount

Claim

Claim tokens upon success or CUBIC contribution upon failure.

Requirements: `Concluded pool`, `Transfer claim`
Outputs: `Token transfer from pool to sender on pool success`, `CUBIC transfer from pool to sender on pool failure`
Inputs:
`Claim` - Claim hash

Resolve

Resolve launch pool after expiry or if creator manually closed pool.

Requirements: `Pool CubicsBalance >= CubicsTarget`
Outputs: `Liquidity transfer from launch pool to swap pool`, `Funds transfer from launch pool to pool creator`
Inputs:
`Token` - Pool token

Create

Create a launch pool for a fungible token.

Requirements: `Token balance`
Outputs: `Launch pool`
Inputs:
`Token` - Token to launch
`CubicsTarget` - Minimum CUBIC target
`CubicsLimit` - Maximum CUBIC limit
`Percentage` - Percentage for liquidity vs. creator
`TransfersLimit` - Maximum number of participants
`Expires` - Datetime of expiry

Deposit

Deposit the fungible tokens and automatically determine the swap rate, based on 50% of the deposited amount allocated for swaps, 50% allocated for liquidity.

Requirements: `Pool creator`
Outputs: `Token transfer from sender to pool`, `Deposit claim from pool to sender`
Inputs:
`Amount` - Amount to be deposited
`Unlocks` - Unlocks datetime for claim

Withdraw

Withdraw previously deposited tokens, or remainder for a closed launch.

Requirements: `Pool creator`, `Token balance in the pool`
Outputs: `Token transfer from pool to sender`
Inputs:
`Claim` - Claim hash

Close

Creators can close the launch pool manually before expiry.

Requirements: `Pool creator`
Outputs: `None`
Inputs: `None`

Lending

Explanation

Lending enables users to borrow fungible tokens against CUBIC collateral. Lending promotes efficient markets as users can borrow tokens to speculate on the decline of a token price. Other use cases for lending might include paying or resolving a claim without owning the token. Lenders can earn attractive interest rates for tokens that they make available for borrowing.

To borrow tokens, visit the lending pool on the Cubics network app, connect your wallet and click on borrow. You will be prompted to specify a collateralization value, depending on the minimum collateralization required by the pool. This value determines the amount of tokens that you will receive in return for the CUBIC amount that you specify.

You can keep borrowed tokens indefinitely as long as your collateralization remains above 75% of the minimum. You can return your tokens using the same lending pool that you've borrowed from while being charged an interest rate based on the borrowed duration. In case your collateralization drops below 75% of the pools' minimum, other users can liquidate your lending claim, and your CUBIC collateral will be used to cover open interest, penalty fees, and the borrowed token amount. The liquidated fees will be used to reimburse lenders.

To lend tokens to others, visit the lending pool and select deposit from the dropdown. You will be able to deposit the associated pool tokens, and you will earn interest from all collected CUBIC rewards proportionally that are collected while your tokens remain deposited.

Intro

Lending pools allow users to borrow tokens against CUBIC collateral. Lending pools allow "short selling" a token or resolving claims without owning the token. At the same time, it enables lenders to earn interest by making their tokens available for borrowers.

Use Cases:
Short-selling a borrowed token based on the assumption that tokens can be purchased back at a lower price (contributing to more efficient markets)
Earning interest by lending out fungible tokens to borrowers

Functionality

A lending pool is created with a certain collateralization and interest rate. Lenders deposit tokens into the lending pool. Borrowers can then use it to borrow tokens while specifying the desired collateral. As long as the collateralization is kept above the minimum, users can keep tokens for as long as they wish and pay the respective interest rate upon settlement. If, however, the collateralization

rate falls below the minimum, any user can liquidate the borrower's claim, which applies penalties equivalent to a one-year interest rate to the borrower's collateral.

All collected funds, via interest rate paid by the borrower, including liquidations, are collected and proportionally divided between lenders based on lending duration and lending amount. Lending pools make it possible for lenders to earn a steady and predictable interest rate.

Methods

Transfer

Lend tokens from the lending pool at certain collateralization.

Requirements: `Collateralization >= Min. collateralization`
Outputs: `CUBIC transfer from sender to pool`, `Token transfer from pool to sender`
Inputs:
`Token` - Pool token (token to borrow)
`Collateralization` - Desired collateralization rate
`Amount` - CUBIC amount to be transferred as collateral

Settle

Return previously lent tokens while paying an interest rate based on the rate specified by the pool and the lending duration.

Requirements: `None`
Outputs: `Token transfer from sender to pool`, `CUBIC transfer from pool to sender`
Inputs:
`Claim` - Claim hash

Liquidate

Liquidate someone's claim that has dropped below 75% of the minimum collateralization required by the pool.

Requirements: `Claim collateralization < 0.75 * min. collateralization`
Outputs: `CUBIC transfer from pool to borrower for remaining collateral`, `CUBIC transfer from pool to liquidator for finders reward`
Inputs:
`Claim` - Claim hash `Token` - Pool token

Deposit

Deposit tokens as specified by the pool to earn a specific interest rate.

Requirements: `None`
Outputs: `Token transfer from sender to pool`, `Deposit claim from pool to sender`
Inputs:
`Amount` - The deposited amount
`Unlocks` - A datetime when the deposit claim can be redeemed

Withdraw

Withdraw tokens that have been deposited previously to the lending pool.

Requirements: `Active deposit claim`
Outputs: `Token transfer from pool to sender`, `CUBIC transfer from pool to sender for interest earned`
Inputs:
`Claim` - Deposit claim hash
`Token` - Pool token
`Percentage` - Percentage to withdraw

Create

Create a lending pool for a fungible token.

Requirements: `None`
Outputs: `Lending pool`
Inputs:
`Token` - Token to lend/borrow
`Percentage` - Collateralization rate
`Rate` - Annual interest rate

Listing

Lock

Explanation

Lock programs enable anyone to lock tokens for a specified period of time. Locks can also be used to lock tokens for others, for example, to distribute locked tokens to team members or investors. Once a lock is created, it can be unlocked only once the specified unlock period expires. Afterward, you or the beneficiary of the lock claim will be able to unlock their tokens.

To participate in an existing lock pool, visit the pool page and connect your wallet. You will be able to lock the tokens associated with the pool for your specified amount of time. Once the lock period expires, you or the beneficiary can unlock these tokens on the same page.

To create a lock pool for your token, visit the Cubics network app, connect your wallet and click on "new pool" under wallet actions. You will be able to specify the associated token and parameters such as minimum or maximum lock time and amount.

Intro

Lock pools make it possible for anyone with a fungible-token balance to lock tokens for a specific time. Locks can be set to unlock after a particular date or be valid only before a date through an expiration mechanism. Furthermore, lock pools allow the transfer of lock claim ownership to other addresses.

Use Cases:
Locking fungible-tokens for someone (e.g. allocated for another entity as a form of vesting)
Locking fungible-tokens until a specific datetime
Locking fungible-tokens claimable after a particular date, and before an expiry date

Functionality

A lock pool is created for a particular token. Users can then use this pool to lock their tokens in exchange for a lock claim. This claim can then be redeemed after the locking period expires. Lock claims can be assigned to another address, allowing project creators to distribute tokens to others in a locked state.

Methods

Transfer

Create a new lock-claim for the token as specified by the pool.

Requirements: `Token balance`
Outputs: `Token transfer from sender to pool`, `Lock claim from pool to sender`
Inputs:
`Amount` - Fungible-token amount to be locked
`Address` - Address for which to lock the tokens
`Unlocks` - Datetime from which the claim will be claimable
`Expires` - Datetime at which the claim will expire and become non-claimable

Claim

Allows participants with a valid lock-claim to unlock tokens from the lock-pool.

Requirements: `Active lock claim`
Outputs: `Token transfer from pool to sender`
Inputs:
`Claim` - Lock claim hash

Loot

Explanation

Loot programs enable token creators to distribute their NFTs in a randomized way. A creator can set up a loot pool and deposit NFTs to it. Other users can then participate for the minimum required contribution and receive a random NFT based on the probability that the creator specified upon creation.

To participate in a loot pool, navigate to the loot pool page, connect your wallet, and click on "participate". You will be prompted to send the minimum required amount of participation tokens to the pool, after which the program will determine whether you win an NFT. If you win, the NFT will be instantly transferred to your wallet.

To create a loot pool and distribute NFTs, visit the Cubics network app, connect your wallet and click on "new pool" under wallet actions. You will be able to specify a participation token, the participation amount, a win probability, and other details. Once the pool has been created, you can deposit NFTs to the pool, which will then be distributed to participants randomly once they enter. Once the pool has collected tokens from participants, you will be able to withdraw these tokens from the pool as earnings.

Intro

Loot pools enable token-creators to promote their NFTs in a randomized fashion. Loot pools are equivalent to NFT drops. Users can participate by paying a certain fungible-token amount and receiving a random NFT with a certain probability.

Use Case: Distributing or launching an NFT collection through randomized token distribution process

Functionality

The creator or owner of NFTs creates a loot pool and sets parameters such as the participation amount and win-probability. The creator deposits NFTs to the pool to be distributed. Participants can then enter by transferring the entry amount to receive a random NFT with a specific probability from the pool.

Methods

Transfer

Participate in the loot pool by transferring the required participation amount.

Requirements: `Positive NFT balance by pool`
Outputs: `NFT transfer to sender based on win-probability`
Inputs: `None`

Create

Create a new loot pool.

Requirements: `None`
Outputs: `Loot pool`
Inputs:
`Token` - A fungible token to be used as participation token
`Percentage` - Probability to win, between 0 and 1
`MinAmount` - Participation amount
`Expires` - Datetime when the loot pool expires

Deposit

Deposit an NFT as loot to the loot pool.

Requirements: `Pool creator`, `NFT ownership`
Outputs: `NFT transfer from sender to pool`, `Deposit claim from pool to sender`
Inputs:
`Token` - NFT token to deposit to the loot pool
`Unlocks` - Time until which the token will remain locked in the pool

Withdraw

Withdraw a previously deposited NFT, which has not been distributed yet.

Requirements: `Pool creator`, `Active deposit claim`
Outputs: `NFT transfer from pool to sender`
Inputs:
`Claim` - Deposit claim hash

Clear

Clear all fungible-token proceeds held by the pool collected as entry amounts by participants by transferring them to the creator.

Requirements: `Pool creator`, `Token balance by pool`
Outputs: `Token transfer from pool to pool creator`
Inputs: `None`

Lottery

Explanation

Lotteries enable the distribution of fungible and non-fungible tokens in a randomized way. Users can participate by sending the minimum required CUBIC amount to the lottery pool. After the lottery expires, winners can claim from the prize pool. In the case of an NFT lottery, the lottery automatically distributes its prize to one random winner.

To participate in a lottery pool, open the lottery pool page, connect your wallet and click on participate. You will be requested to transfer the minimum required CUBIC amount to enter, in exchange for a participation claim. After the lottery pool expires or reaches its maximum number of participants, you will be able to claim your ticket and you will be awarded from the prize pool with the probability as determined by the pool upon resolution.

To create a lottery, visit the Cubics network app, connect your wallet and select "new pool" under wallet actions. You will be able to specify the associated prize token and values such as minimum participation amount, number of winners, or required CUBIC balance of eligible users. Once the pool has been created, you need to deposit the prize tokens to the pool, after which users can participate until the pool expires and resolves.

Intro

Lottery pools allow the distribution of fungible or non-fungible tokens in a randomized way. Lotteries can be based on specific criteria, such as a minimum token amount that someone must hold to participate or require a minimum amount to enter. After a lottery expires, it can be resolved to automatically distribute the NFT or make the fungible tokens claimable for winning participants.

Use Cases:
Distributing an NFT through a randomized mechanism and a single winner
Distributing fungible-tokens through a randomized mechanism

Functionality

A lottery pool can promote a fungible or single non-fungible token to participants, who can enter to win with a probability dependent on the number of entries. After creating the lottery pool, the creator must deposit the fungible tokens or a single NFT. Once the deposit is complete, any user can participate, either for free or the minimum amount required by the pool. Furthermore, pools can set a minimum amount of CUBIC that a user needs to hold to be eligible to participate. Requiring a minimum amount of CUBIC to participate improves incentives and prevents dishonest entries (especially if the lottery pool is sponsored).

After the lottery pool expires, the resolution call automatically transfers the

NFT to a single winner chosen from all participants. If the lottery is promoting a fungible token, winners (based on the pools' probability) can claim their fungible-token rewards using their claimable ticket.

Methods

Transfer

Transfer the entry amount to participate in the lottery.

Requirements: `Token balance`, `CUBIC balance`
Outputs: `CUBIC transfer from sender to pool`, `Transfer claim from pool to sender`
Inputs: `None`

Claim

Claim a lottery ticket, and receive the pool token with a certain probability.

Requirements: `Active transfer claim`
Outputs: `NFT or token with a probability as specified by the pool`
Inputs:
`Claim` - Claim hash

Resolve

Resolve a lottery pool and determine the winner of an NFT lottery.

Requirements: `NFT lottery`
Outputs: `NFT transfer from pool to random participant`
Inputs: `None`

Create

Create a lottery pool for fungible tokens or a single non-fungible token.

Requirements: `None`
Outputs: `Lottery pool`
Inputs:
`Token` - The token which is to be promoted
`MinAmount` - Participation amount
`TokenTarget` - Amount of pool tokens that someone must hold to participate
`CubicsTarget` - Amount of CUBIC, that someone must hold to participate
`TransfersLimit` - Maximum amount of participants
`ClaimsLimit` - Maximum number of winners
`Expires` - Datetime when lottery naturally expires

Deposit

Allows the creator to deposit the pool token.

Requirements: `Pool creator`
Outputs: `Token transfer from sender to pool`, `Deposit claim from pool to sender`
Inputs:
`Amount` - Amount in case of a fungible-token lottery
`Unlocks` - Datetime when the claim unlocks

Withdraw

Allows the creator to withdraw a previously deposited pool token.

Requirements: `Pool creator`, `Active deposit claim`
Outputs: `Token transfer from pool to sender`
Inputs:
`Claim` - Deposit claim hash

Close

The creator can close the lottery before expiration. If there are participants, the creator can only withdraw the unclaimable share.

Requirements: `Pool creator`
Outputs: `None`
Inputs: `None`

Clear

Allows the creator to clear CUBIC earnings collected by the lottery pool if the participation amount is positive.

Requirements: `Pool creator`, `CUBIC balance by pool`
Outputs: `CUBIC transfer from pool to pool creator`
Inputs: `None`

Cubics Program Overview

Cubics focuses on gaming and metaverse applications that commonly require features such as auctions, voting, swapping. With Cubics, we provide 10 in-built programs, which can be specified as program when creating a new pool:

Auction

Auctions allow anyone to offer fungible or non-fungible tokens for sale. Bids are handled by the program, and auctions are considered successful when the auction pool has no target, the target is met, or the limit is met (equivalent to an instant purchase at maximum price).

Launch

Fundraising pools allow creators to raise CUBIC until a specific target or limit is met. Successful funds distribute portions of the raised amount to the creator and the remainder to the assets' swap pool as liquidity. If a pool rate is set, the fund automatically swaps CUBIC to asset tokens at that specified rate directly, without the condition of having to meet a fundraising goal.

Lending

Lending pools allow users to borrow tokens against CUBIC collateral. Lending pools allow "short selling" a token or resolving claims without owning the token. At the same time, it enables lenders to earn interest by making their tokens available for borrowers.

Lock

Locking pools allow asset owners to lock funds for various use-cases, such as delayed payouts and vesting.

Loot

Loot pools allow NFT creators and games to distribute (drop) new NFTs, items, skins from a collection in a randomized and engaging way. Users can enter for free or for a fee as specified by the pool creator. Entrants receive tickets that have a certain probability of winning a random item from the loot pool.

Lottery

Promo pools provide an airdrop/lottery mechanism, which can engage users or distribute tokens in a simple and fair way. Users can participate for free or for a fee (specified upon pool creation) and receive tickets with a certain probability to win from the prize pool.

Royalty

Royalty pools are set up by an asset creator who wishes to reward all or parts of his assets based on a specified daily rate. Currently, this finds application in providing NFTs with yield for holding (royalties).

Staking

Staking pools allow token creators to reward holders of fungible tokens, depending on lock-up length and amount. For example, a pool creator might specify a certain APY and bonus rate for his pool, deposit reward tokens, and allow users to lock up tokens into the pool who will receive rewards once their lock-up period expires.

Swap

Swapping pools are pools holding CUBIC and the pool asset, which can be exchanged one for another. Swap pools are the same functionality that makes popular DEXs like Uniswap or PancakeSwap possible.

Vote

Voting pools allow users to submit votes given a list of assets/addresses or numbers (prediction markets). Voting can be used for governance (voting for new proposals), for predictions with payouts to the winning voters, or to engage users (a mechanism to vote on any tokenized objects and get paid if the vote was correct after pool expiry).

Programs

Cubics focuses on Gaming and Metaverse applications that frequently require features such as auctions, voting, swapping, and similar. Cubics provides ten in-built programs available as the underlying mechanism when creating a new asset pool. Asset pools are smart contracts that send and receive transactions while executing a set of pre-written instructions tied to an underlying asset.

- Auction
  - Auctions allow anyone to offer fungible or non-fungible tokens for sale. Bids are handled by the program, and auctions are considered successful when the auction pool has no target, the target is met, or the limit is met (equivalent to an instant purchase at maximum price).
- Launch
  - Fundraising pools allow creators to raise CUBIC until a specific target or limit is met. Successful funds distribute portions of the raised amount to the creator and the remainder to the assets' swap pool as liquidity. If a pool rate is set, the fund automatically swaps CUBIC

to asset tokens at that specified rate directly, without the condition of having to meet a fundraising goal.

- Lending
  - Lending pools allow users to borrow tokens against CUBIC collateral. Lending pools allow "short selling" a token or resolving claims without owning the token. At the same time, it enables lenders to earn interest by making their tokens available for borrowers.
- Lock
  - Locking pools allow asset owners to lock funds for various use-cases, such as delayed payouts and vesting.
- Loot
  - Loot pools allow NFT creators and games to distribute (drop) new NFTs, items, skins from a collection in a randomized and engaging way. Users can enter for free or for a fee as specified by the pool creator. Entrants receive tickets that have a certain probability of winning a random item from the loot pool.
- Lottery
  - Promo pools provide an airdrop/lottery mechanism, which can engage users or distribute tokens in a simple and fair way. Users can participate for free or for a fee (specified upon pool creation) and receive tickets with a certain probability to win from the prize pool.
- Royalty
  - Royalty pools are set up by an asset creator who wishes to reward all or parts of his assets based on a specified daily rate. Currently, this finds application in providing NFTs with yield for holding (royalties).
- Staking
  - Staking pools allow token creators to reward holders of fungible tokens, depending on lock-up length and amount. For example, a pool creator might specify a certain APY and bonus rate for his pool, deposit reward tokens, and allow users to lock up tokens into the pool who will receive rewards once their lock-up period expires.
- Swap
  - Swapping pools are pools holding CUBIC and the pool asset, which can be exchanged one for another. Swap pools are the same functionality that makes popular DEXs like Uniswap or PancakeSwap possible.
- Vote
  - Voting pools allow users to submit votes given a list of assets/addresses or numbers (prediction markets). Voting can be used for governance (voting for new proposals), for predictions with payouts to the winning voters, or to engage users (a mechanism to vote on any tokenized objects and get paid if the vote was correct after pool expiry).

Cubics Program Overview

Explanation

Let me walk you through all official pool programs on Cubics. To start, let's see why and when you might want to use a pool. Pools enable trustless execution of code through prewritten programs. When you deal with another party, trust plays an essential part in the transaction. With pools, you can simply trust that the program will execute the code instead of relying on human integrity.

There are ten prewritten programs on Cubics, which have been extensively tested and verified. Let's look at each pool program and explain what they are used for.

Auction programs enable creators to sell their NFT. Users submit bids to the auction pool, and upon completion, the pool automatically transfers the assets to the individual parties depending on the success of the auction.

Launch programs enable projects to raise funds by distributing tokens to participants who wish to contribute CUBIC tokens. The CUBIC contribution is added to the liquidity of the launched token and can be traded afterward through its swap pool. Participants in the launch can claim their share of tokens after the launch completes.

Lending programs enable the lending and borrowing of fungible tokens. Lenders can deposit tokens to earn interest, and borrowers can borrow tokens in exchange for CUBIC collateral.

Lock programs enable users to lock tokens for themselves, or for someone else. This might come in handy when a project wants to distribute tokens without making them available instantly.

Loot programs are dropzones, where NFT creators can launch and distribute their NFTs in a randomized way. Creators specify a win probability, and users can transfer the required participation amount to the loot pool. Every transfer has a chance to win a random NFT, dependent on the loot pool probability.

Lottery programs enable creators to run promotions for a prize, which can be a fungible token or an NFT. Users enter a lottery for free or for the minimum required contribution amount and receive a participation ticket otherwise known as a claim. Once the lottery completes, users can redeem their claim to see if they hold a winning ticket and receive their part of the prize.

Royalty pools allow users to earn passive income by holding NFTs. Creators of NFTs can set up a royalty pool and deposit reward tokens to it. Royalties accumulate day by day and can be claimed by holders of eligible NFTs, which are specified by the creator.

Staking programs enable token creators to offer rewards for locking tokens for a certain amount of time. Users can stake tokens, and receive interest as well as bonus tokens in return once their staking period expires.

Swap programs are simply decentralized exchanges operating through a mathematical swap function. After liquidity is added to a swap pool, users can swap CUBIC into tokens or vice versa in a fully trustless and open way.

Vote programs enable voting on numeric targets like budgets, prediction markets, or a predefined set of candidates. Furthermore, vote programs can create betting games that reward voters with the correct answer.

To create a pool, visit the Cubics network app and connect your wallet. Click on your wallet, and select "new pool" under your wallet actions. Specify the associated pool token as well as the program. Check the Cubics documentation for your selected program or the program explanation video to find out more about the program and any additional fields that you can use to customize your pool.

Royalty

Explanation

Royalty programs enable NFT holders to earn passive income by claiming royalties from it. Creators set up a pool with token rewards for all or a subset of NFTs, and any user holding an eligible NFT can claim royalties simply for holding the NFT.

To claim royalties, visit the royalty pool page, connect your wallet and click on "claim". You will automatically receive a payout from the pool, as long as you hold an NFT that is eligible. Rewards accumulate day by day, and you receive daily rewards that have accumulated since your last claim date.

To create a royalty pool and reward NFT holders with passive income, visit the Cubics network app, connect your wallet and click on "new pool" under wallet actions. You will be able to specify the associated reward token as well as a daily reward rate. Once the pool has been created, you can deposit reward tokens to it, so that other users can claim these tokens as royalties.

Intro

Royalty pools allow NFT creators to allocate a particular token for royalty payouts of NFT holders. Anyone who owns an NFT specified by a royalty pool can claim royalties based on the holding duration.

Use Cases:
Reward NFT holders based on the length of holding a certain NFT Pay a passive income/royalties for anything in the physical or virtual reality, represented by an NFT

Functionality

A creator creates a royalty pool and specifies candidates and a linear pay-out rate of rewards. After depositing rewards, any owner of NFTs created by the creator (and as determined by the royalty pool) can then periodically claim royalties from the pool. Royalties accumulate every day until a claim is made (claims reset the daily counter).

Methods

Claim

Claim royalties for an NFT created by the royalty-pool creator, depending on the length of holding the NFT.

Requirements: `Claim answer in pool candidates, or no candidates`
Outputs: `Royalty transfer from pool to sender`
Inputs:
`Token` - Token address

Create

Create a royalty pool while specifying the eligible NFT candidates and a daily royalty rate.

Requirements: `None`
Outputs: `Royalty pool`
Inputs:
`Token` - A fungible token that is used as reward payout
`Candidates` - A list of NFT candidates, or empty if all NFTs by the creator should be eligible
`Rate` - Amount of pool-tokens that is paid for every day of holding the NFT
`Expires` - Datetime when the royalty pool is no longer valid

Deposit

Deposit a certain amount of pool-tokens available as reward payouts for NFT holders.

Requirements: `Pool creator`
Outputs: `Reward token transfer from sender to pool`, `Deposit claim from pool to sender`
Inputs:
`Amount` - Reward amount to be deposited
`Unlocks` - Datetime when the deposit claim can be withdrawn

Withdraw

Withdraw previously deposited reward tokens (partially if rewards have been claimed already).

Requirements: `Pool creator`, `Active deposit claim`, `Token balance by pool`
Outputs: `Token transfer from pool to sender`
Inputs:
`Claim` - Deposit claim hash

Close

The creator can close a royalty pool at any time, making it impossible to claim additional royalties.

Requirements: `Pool creator`
Outputs: `None`
Inputs: `None`

Staking

Explanation

Staking enables users to commit tokens for a specified duration of time, and earn rewards based on the staking interest and bonus rate.

To participate in a staking pool, visit the pool page, connect your wallet and click on "stake". You will be required to enter your desired amount as well as the lock duration. The pool issues a claim to you, which you can redeem once your lock period expires. You will receive your initial contribution, including interest and bonuses accumulated depending on your selected staking duration.

To create a staking pool for a token, connect your wallet and click on "new pool" under wallet actions. You can specify the token, the minimum and maximum staking time and amount, as well as the yearly interest and bonus rate. After you create the pool, deposit reward tokens to it so that other users can lock tokens and receive rewards once their staking claim unlocks.

Intro

Staking pools allow creators to reward holders for locking (staking) their tokens for a definite period. After the period passes, the participant can unlock their staked tokens and receive an additional reward from the deposited rewards, based on the specified yearly interest rate and yearly interest bonus.

Use Case: Rewarding holders of a fungible token with a certain APY and bonus rate for committing their holding by locking tokens

Functionality

A creator creates a staking pool with a specific yearly APY rate and an optional bonus percentage. Optionally, a minimum/maximum amount and locking time may be set. After the pool is created, the creator deposits tokens as reward payouts. Any holder of the pool token can lock and commit their tokens for a reward payout based on staking time and APY, and bonus rate. Once the staking period passes, the participant can unlock their tokens and receive their initial stake, including the interest rate as defined by the pool.

Methods

Transfer

Stake tokens by locking them for a specific time.

Requirements: `Min/max amount valid`, `Min/max time valid`
Outputs: `Token transfer from sender to pool`, `Staking claim from pool to sender`
Inputs:

`Amount` - Staking amount
`Unlocks` - Datetime until which the amount will be locked

Claim

Claim back the initial stake, including rewards accumulated during the staking period.

Requirements: `Active staking claim`
Outputs: `Token transfer from pool to sender (with rewards)`
Inputs:
`Claim` - Claim hash

Withdraw

Withdraws reward tokens as creator or deposited tokens as a participant (but without rewards).

Requirements: `Active staking claim`
Outputs: `Token transfer from pool to sender (without rewards)`
Inputs:
`Claim` - Deposit claim hash
`Percentage` - Percentage to withdraw (defaults to 1 equivalent to 100%)

Create

Create a staking pool with a specific APY rate and bonus percentage.

Requirements: `None`
Outputs: `Staking pool`
Inputs:
`Token` - Pool token to be staked and paid as rewards
`Rate` - The yearly APY rate
`Percentage` - A bonus percentage, paid additional to the annual APY rate for one year of holding
`MinAmount` - Minimum staking amount
`MaxAmount` - Maximum staking amount
`MinTime` - Minimum staking time in seconds
`MaxTime` - Maximum staking time in seconds
`TransfersLimit` - Maximum number of transfers to the staking pool
`Expires` - Datetime when the staking pool expires and is not accepting new participants

Deposit

Deposit reward tokens to the staking pool.

Requirements: `Pool creator`
Outputs: `Token transfer from sender to pool`

Inputs:
`Amount` - Amount to be deposited as rewards
`Unlocks` - Lock time of the reward token claim

Swap

Explanation

Swap pools are automatically created for every fungible token that is minted on Cubics. Swaps enable anyone to exchange CUBIC for tokens and vice versa. This exchange process is fully decentralized, and prices are automatically determined based on the available CUBIC and token liquidity.

To trade a token, visit the swap pool page of a token and connect your wallet. You will be able to select the amount you would like to swap, as well as the input token. Once you confirm your transaction, you will receive the output token within a short period of time.

Intro

Swap pools are automatically created for every fungible token that is minted. These allow participants to swap tokens against CUBIC and vice versa in a decentralized way. The price is based on the liquidity that token creators and participants can contribute, and the trades that affect the liquidity.

Use Case: Decentralized exchange mechanism for fungible tokens

Functionality

A swap pool is automatically created with the creation of every fungible token. Creators can add liquidity via a launch pool, or directly by depositing the pool tokens and CUBIC. The amount of deposited liquidity determines the price, which is determined by dividing the number of tokens available as liquidity by the amount of CUBIC available as liquidity. Participants can swap CUBIC for tokens or tokens for CUBIC. Transfers in and out are entirely handled by the program in an automated way, charging 0.25% commission on every trade and automatically adding 1% of the trade value to the pools' liquidity.

Methods

Transfer

Swap tokens into CUBIC or CUBIC into tokens at the current exchange rate.

Requirements: `Pool liquidity`
Outputs: `Transfer from sender to pool`, `Transfer from pool to sender`
Inputs:
`Token` - CUBIC or the pool-token, to use as input token
`Amount` - Number of tokens to swap

Deposit

Deposit CUBIC and tokens to liquidity at a specific rate.

Requirements: `CUBIC balance`, `Token balance`
Outputs: `CUBIC transfer from sender to pool`, `Token transfer from sender to pool`, `Liquidity deposit claim from pool to sender`
Inputs:
`TokenAmount` - Amount of tokens to deposit
`CubicsAmount` - Amount of CUBIC to deposit
`Unlocks` - Datetime when the deposit claim shall unlock

Withdraw

Allows withdrawing previously deposited liquidity tokens adjusted for the new liquidity rate.

Requirements: `Active liquidity deposit claim`
Outputs: `CUBIC transfer from pool to sender`, `Token transfer from pool to sender`
Inputs:
`Claim` - Deposit claim hash
`Percentage` - Percentage between 0-1 to withdraw

Vote

Explanation

Voting pools enable users to vote on a numeric outcome such as budgets and prediction markets, or on candidates, based on a predefined list of choices. Answer-based vote pools can resolve in multiple ways and reward the creator, the top candidate, the voters who voted correctly, or the voters of an answer that was set through an oracle. Rewarding voters of the most voted answer or the answer as set by an oracle, allows voting pools to be used as betting games with reward functionality.

To submit a vote, visit the vote pool page, connect your wallet and click on vote. You will be able to specify a number or answer that you would like to vote on, as well as an amount to add weight to your vote. If the voting mechanism is set to reward voters, voters can claim rewards after the voting pool resolves.

To create a vote, connect your wallet and click on "new pool" under wallet actions. You will be able to specify a list of candidates to create a choice-based vote, as well as the mechanism of resolution after the vote expires.

Intro

The voting pool allows participants to vote on a numeric outcome (such as bets, estimations, budgets, prediction markets) or on an outcome based on provided answers. Voting pools can require a contribution amount, which serves as a weight for the vote. Vote pools can be resolved using various mechanisms: transferring all collected tokens to the creator, the highest candidate, or the voters (by the highest voted answer or through oracle result).

Use Cases:
Voting on a numeric outcome (bets, estimations, budgets, prediction markets)
Voting on an answer outcome (guesses, candidates, tokens, proposals)
Voting by collecting funds and rewarding the vote pool creator
Voting by collecting funds and rewarding the highest voted candidate
Voting by collecting funds and rewarding voters of the highest voted answer
Voting by collecting funds and rewarding voters of the correct answer, as determined by an oracle

Functionality

A creator creates a voting pool and specifies the participation token, resolution mechanism, and optional candidates. Optionally, a minimum/maximum participation amount can be specified. Voters can then vote by providing a numerical answer (if the voting pool is a numerical vote) or by providing one of the specified answer candidates. Every vote can be weighted by the CUBIC participation amount. After the vote expires, it can be resolved based on the specified resolution mechanism. The mechanism rewards the creator, candidates

or voters of the highest voted answer or the correct answer as determined by an oracle. If the mechanism rewards voters of the highest or correct answer, they can claim their rewards proportional to their weighted vote.

Methods

Transfer

Participate in the voting pool by providing a numerical answer or a candidate from the available choices.

Requirements: `None`
Outputs: `Transfer claim from pool to sender`
Inputs:
`Amount` - CUBIC amount to add weight to the vote
`Answer` - Choice from available candidates
`Number` - Number if the vote is numerical

Claim

Allows claiming rewards, proportional to the weighted vote divided by the total weight of the winning vote.

Requirements: `Claim with valid answer`
Outputs: `Reward transfer from pool to sender`
Inputs:
`Claim` - Claim hash

Resolve

Resolves vote using the specified vote-pool mechanism. Pays collected CUBIC weight amounts to either creator, top candidate or enables voters of the highest answer to claim their rewards.

Requirements: `Vote mechanism != oracle`
Outputs: `Rewards transfer from pool to creator/candidate`
Inputs: `None`

Create

Creates a voting pool, with a numerical or candidate type and a specific resolution mechanism.

Requirements: `None`
Outputs: `Vote pool`
Inputs:
`Mechanism` - Resolution mechanism, creator, candidate, voters, oracle
`Candidates`- List of candidates to vote on, numerical vote if empty
`MinAmount` - Minimum amount to participate in vote

`MaxAmount` - Maximum amount to participate in vote
`TransfersLimit` - Maximum number of votes
`ClaimsLimit` - Maximum number of winning votes for top candidate
`CubicsLimit` - Maximum collected CUBIC as vote-weights
`Expires` - Datetime when the vote expires

Oracle

The creator can set the correct answer so voters who claimed that answer can claim their rewards.

Requirements: `Pool creator`, `Candidate pool`, `Pool mechanism = oracle`
Outputs: `None`
Inputs:
`Answer` - Correct answer

Bridge

Explanation

The Cubics Bridge enables users to swap Ether, BNB, or Cubics from Binance Smart Chain into native Cubics tokens and vice versa.

To create a bridge request, visit the Cubics network app and connect your wallets for the chains you want to exchange between. You will be able to specify your desired input amount, and you will see the approximate output amount. After you send the input transaction, it should take less than 5 minutes for the swap request to be completed, with a fee of around 2%.

Intro

The Cubics bridge swaps between ETH/BNB/BSC CUBIC and native CUBIC tokens, with slippage of around 1.5% (plus regular liquidity slippage for amounts that impact liquidity). This process also works vice versa. The swap process takes about 5 minutes and is completely automated in both directions.

Use Cases:
Swapping supported assets from Ethereum (ETH) or Binance Smart Chain (BNB, CUBIC) to Cubics native tokens, Swapping native Cubics tokens into assets on Ethereum (ETH) or Binance Smart Chain (BNB, CUBIC)

Functionality

The user makes an input transfer to Cubics's bridge wallet address on the chosen input chain. Cubics listens to all incoming transactions with a specifically formatted schema (the data/message field of transactions) and awaits a sufficient number of confirmations. After the transaction is confirmed, Cubics calculates the exchange rate based on swap size and its effect on liquidity to account for possible slippage. Slippage fees added on top of the base fee of 1.25%. Cubics then sends the output transfer on the chosen target chain to the wallet address connected by the user.

Methods

Create Bridge Request

Create a new bridge request.

Requirements: `ETH/BSC wallet`, `CUBIC wallet`
Outputs: `Transfer on target chain`
Inputs:
`ETH/BSC public address`
`CUBIC public address`
`Swap amount`

```
Source-chain
Target-chain
```

Faucet

Explanation

The Cubics Faucet enables anyone with a Gmail account to receive 10 Cubics tokens on Mainnet and 1000 Cubics tokens on Testnet.

To receive your tokens, just visit the Cubics network app and select the free token option on the homepage. You will be prompted to create or connect a Cubics wallet and enter your email address. After confirming your email, you will receive the tokens to your connected wallet. Go ahead and test how fast transfers are on Cubics, or start minting your first tokens!

Intro

Cubics's Faucet allows users to request a certain number of free CUBIC tokens on the Mainnet and Testnet. It allows users and creators to try out Cubics without purchasing or swapping tokens.

Use Cases:
Create a few simple transactions on Mainnet
Create, test, and launch any desired functionalities on Testnet

Functionality

Users who own a Gmail address can request free CUBIC tokens from the faucet feature. The user needs to confirm their email, after which the Faucet automatically makes two transfers to the users' connected wallet, one on Mainnet and one on Testnet.

Methods

Create Faucet Request

Create a faucet request and receive tokens on Mainnet and Testnet.

Requirements: `Valid Gmail address`, `Cubics wallet`
Outputs: `CUBIC transfer from faucet to wallet on Mainnet`, `CUBIC transfer to from faucet wallet on Testnet`
Inputs: `None`

Sponsored

Explanation

Cubics enables users to sponsor an address and delegate network fees. Sponsored addresses automatically reimburse other users who send transfers to it. This works exceptionally well with pools that would like to enable free transactions. Think about a swap pool without fees or a voting pool where anyone can vote without dealing with Cubics tokens. Similarly, you can sponsor your address and allow others to send you transfers without a fee.

To sponsor an address, visit the Cubics network app and connect your wallet by clicking "connect wallet" on the top right. Under wallet actions, select "new transfer" and enter the address you would like to sponsor in the to-field. Under token, select "Cubics Sponsored," which will be debited as Cubics and credited as sponsored Cubics to the recipients' balance. Just keep in mind that sponsored Cubics balances cannot be transferred, even if you sponsor your own address.

Intro

Cubics enables users to transfer a sponsored balance to addresses or pools, so other users interacting with these resources can do so without paying network fees.

Use Cases:
Recipient sponsoring incoming transfers to their address
Pool sponsoring incoming transfers and enabling participation without owning CUBIC
Project creators who would like to encourage participation in particular pools

Functionality

A user transfers sponsored CUBIC tokens to an address or pool. The transferred token is specified as sponsored CUBIC and is automatically swapped from the users' CUBIC balance into sponsored CUBIC and credited to the resource. Other users who interact with the sponsored resource won't be charged any fees until the sponsored balance of the resource is depleted.

Methods

Create

Swaps the users' owned CUBIC into sponsored CUBIC. Sponsored CUBIC cannot be withdrawn.

Requirements: `CUBIC balance`
Outputs: `Sponsored CUBIC transfer from sender to recipient`
Inputs:

`To` - Recipient resource to be sponsored
`Token` - Must be specified as 111111111111111111111111sponsored
`Amount` - Amount to sponsor
`From` - From address in case of allowance
`Message` - Custom transfer message

Wallet

Explanation

The Cubics Wallet enables users to create a managed or self-managed wallet on Cubics. Wallets can be used to send and receive transfers, mint and hold tokens, or participate in pools. Wallets on Cubics are available in two options.

The self-managed wallet generates a private and public key pair that you have to store securely. The 3rd party managed wallet option handles private keys for you and allows you to access your wallet with a username and password.

To create a new Cubics wallet, visit the Cubics network app and click on "connect wallet" in the top right corner. You will be prompted to choose between the managed or self-managed option to create a new wallet. To connect an existing wallet, choose your desired method and enter your private key if you have a self-managed wallet or your account username and password if you choose the managed wallet option.

Once your wallet is connected, you can click on it to perform various actions, such as viewing your address page, backing up your keys, or creating transactions.

Intro

Cubics's wallet allows users to create and manage their CUBIC blockchain wallets, either by managing keys yourself or letting a 3rd party provider (currently Cubics foundation) manage keys for you.

Use Cases:
Create a new self-managed or managed Cubics wallet
Connect an existing self-managed or managed Cubics wallet

Functionality

Users need a wallet to interact with the Cubics blockchain. Users can create a key pair (public key/private key) or let a provider manage this keypair for them (using a more memorable account and secret). Once a wallet is created through the self-managed or managed option, it can be connected to Cubics's apps, as long as the owner has access to the private key or account and secret (in the case of managed keys).

Methods

Create Self-Managed

Create a new CUBIC blockchain key pair.

Requirements: `None`
Outputs: `Public key`, `Private key`
Inputs: `None`

Create Managed

Create a managed wallet.

Requirements: `None`
Outputs: `Public key`
Inputs:
`Account` - Memorable account name, for example, email, phone number, or username
`Secret` - Memorable password

Connect Self-Managed

Connect a self-managed wallet.

Requirements: `Private key`
Outputs: `None`
Inputs: `None`

Connect Managed

Connect a managed wallet.

Requirements: `Account name`, `Account secret`
Outputs: `None`
Inputs: `None`