vMOONEY Vesting

Author: pmoncada
Date: Created May 30th, 2023

Abstract

For Q1 vesting and for all MOONEY vesting going forward in exchange for work done for the DAO, the MOONEY will be locked in the form of vMOONEY before being given to contributors.

vMOONEY will be locked for 4 years and contributors will immediately be able to vote with the vMOONEY.

Problem Statement

A recurring point of contention between the broader MoonDAO community v.s. Contributors is that Contributors sell the MOONEY they received from our Retroactive Rewards and this creates a downhill spiral of the community feeling like the contributors are "enemies of the DAO."

Solution

We implement a solution to distribute vMOONEY directly to contributors, that unlocks after 4 years into MOONEY.

For Q1 and Q2 we will create these transactions manually and send them in the form of a Gnosis. In the future we will automate and make this process simpler.

Reading:
https://blog.thirdweb.com/erc-6551-token-bound-accounts/
https://blog.thirdweb.com/account-abstraction-erc4337/
https://eips.ethereum.org/EIPS/eip-4337

ERC-4337 allows for the creation of a smart account. This means that transactions can be made on a user's account without them needing to initiate it. These transactions can be programmed, therefore we could program in a transaction that once MOONEY is received in an account the MOONEY is locked and delegated to a certain wallet owned by the contributor.

ERC-6551 allows us to create multiple wallets that are bound to an ERC-721 NFT. We could have one ERC-721 representing each contributor, and that ERC-721 is owned by our multisig and we create a new wallet each time the contributor receives vested payment, and then the wallet locks the MOONEY into vMOONEY and delegates the voting power to the user's Externally owned wallet.

We then allow for the user with their Externally owned wallet to be able to claim the MOONEY once the vesting period ends. Optionally, we have a "rug" switch if the member leaves MoonDAO before 6 months to implement a cliff. Unsure if this is needed.

For ease of understanding:
ERC-721 = NFT
ERC-4337 = Smart account
ERC-6551 = Token bound accounts

Therefore the flow looks like:

1. MoonDAO creates an NFT for each Contributor.
2. Every quarter we generate a new token bound account wallet within the NFT that represents that Contributor's quarterly earnings. We simply lump-sum the MOONEY into the new wallet.
3. Using smart accounts we automatically lock all that MOONEY for 4 years, and then delegate the votes to the contributor.
4. At the end of the locking period, the MOONEY can be claimed by the user and sent to their regular wallet to do whatever they'd like with the MOONEY.

As an MVP to issue the Q1 and Q2 vesting payments, until the full implementation, we can use a flow that would look like:
1. MoonDAO creates multi-sigs with each contributor
2. MoonDAO Delegate the vMOONEY voting power
3. MoonDAO leave the contributor as the only signer for that wallet.

Specification:
1. ***Wallets:***
    a. *Incentives Operator Wallet(IOW)*
        i. *A new wallet for creating safes, creating transactions, signing transactions and executing transactions using the cli. This wallet is also a delegate of the treasury (can only create transactions)*
    b. *Incentives Operations Signer(IOS)*
        i. *A new wallet that is added as an owner of the newly created safes. This wallet is used only for signing and executing transactions of contributor safes.*
2. ***Safe-Cli Scripts***
    a. *Init Vesting (IOW)*
        i. *Loop through a CSV containing contributor addresses and mooney values, create a new safe for each address and add the contributor's*

address as an owner. Append the CSV adding the safe address for each contributor.

Safe:

- *Owners: IOW, IOS, contributor*
- *Threshold: 2/3*

ii. *Loop through the appended CSV and create a batch transaction for the Treasury that sends the mooney values to the safe addresses.*

b. *Approve, Lock, Delegate (IOW)*
   i. *Loop through the CSV, create 3 transactions for each contributor safe address:*
      1. *Approve the amount mooney received from the Treasury for the vMooney contract*
      2. *Lock Mooney for 4 years*
      3. *Delegate vMooney from the contributor's safe to the contributor's "owner" address*

   ii. *Append the CSV with the new transactions*

c. *Signing (IOW & IOS)*
   i. *Loop through the CSV, sign each transaction and execute if possible*

d. *Unlink (IOW)*
   i. *Loop through the CSV, set the threshold of each contributor safe to 1 and remove the IOW and IOS as owners*

Benefits

- While we're building, MoonDAO Contributors can't sell their stake.
- We can pay members in larger sums of MOONEY to protect our treasury without worrying about price impact.
- Avoid negative situations pitting the Contributors against the broader Community.
- Members don't need to spend gas locking their MOONEY or wait for it to vest over the period of a year.
- Members can vote immediately with the MOONEY they receive.

Risks

- Implementation risks. This proposal uses new features of Ethereum (account abstraction and token bound wallets). There may not be an "off-the-shelf" contract we can use for this.
- It takes a longer time than expected to get this to run effectively.

Objectives and Key Results:

Objective #1: MoonDAO Members receive vMOONEY directly for their Contributions, rather than MOONEY.
Key Results for Objective #1:
- Successfully send Q1 and Q2 payments sent out in vMOONEY manually.
- Payments going forward are sent out in vMOONEY with a new implementation.

Member(s) responsible for OKR and their role:
- Name.get: Implementation going forward, manual signatures for Q1 and Q2
- LucasArt: Implementation going forward, manual signature for Q1 and Q2
- Pmoncada: Implementation going forward, manual signatures for Q1 and Q2

Team Table (Table A)

*There can be a maximum of two team Rocketeers. Project teams may not need an initial team. It can just be an individual submitting a proposal. As a general rule of thumb, try to keep teams small and focused in the beginning, **with clear deliverables and OKRs for each member**.*

Team Rocketeer(s)	*pmoncada*
Initial Team	Name.get, LucasArt
Delegate	*mitchie*
Multi Language Representative (optional)	*N/A*
Multisig signers	*N/A*
Revenue Split Agreement Address	*N/A*

Team Member:
Bio:

Estimated Project Timeline (Table B)

Date	**Description**
12 July	Senate Vote

18-19 July	Manual distributions
1 August- 22 September	Implementation going forward

Deadline for the project: End of Q3.

Budget Justifications (Table C)

These are fixed costs to make your project happen. This might also include bounties that you'll make inside the DAO (it's recommended to have some amount allocated for bounties or competitions), or specific work that must be contracted out to complete the project. Please provide links to quotes where possible.

Description	**Amount**	**Justification**
Gas fees	From 10 MATIC up to 0.3 ETH	We are considering doing this process on an L2, but if it's not possible due to incompatibility problems, this total amount would be spent in gas fees to run it on Ethereum Mainnet.
Total	**0.3 ETH**	

Transactions to be Executed (Table D)

Transaction Type	Amount	Token Type	Receiving Address
Send	0.3	ETH	TBD

Revenue Distribution Agreement

- ☑ ~~N/A~~
- ☐ Standard MoonDAO Agreement.
    - ☐ Terms here:
- ☐ Different Agreement
    - ☐ Link to on-chain agreement:
    - ☐ Link to off-chain agreement:

In-Depth Approval Senate Signature

This is only required if your project requires up-front funding from MoonDAO or it is going to crowd-raise with support from MoonDAO.

Senate Member Signature #1 ____________Luffy54________
Senate Member Signature #2 __________Mitchie__________
Senate Member Signature #3 ____________manu________

Appendix

1. Create Incentives Operator Wallet (one time) (IOW)
2. IOW becomes delegate to the MoonDAO Treasury
3. IOW runs script taking in CSV of receivers and MOONEY amount
4. Script uses Safe CLI to generate N addresses with:
    a. Two MoonDAO operators
    b. The receiver
    c. 2/3 threshold

5. Script reads CSV and creates custom_send for each generated address with correct amount and creates delegate transaction from MoonDAO Treasury with this amount
6. Wait for Treasury signers to complete the transaction
7. IOW runs a second script to loop through each safe with the following transactions:
    a. use custom_send to queue approve locking
    b. use custom_send to queue lock tokens for 4 years
    c. use custom_send to delegate voting power to receiver
    d. remove two owners of the multisig
8. Operators sign 5*N transactions for each wallet