Grant Title:

CoW Hooks dApps

Author:

@bleu @yvesfracari @ribeirojose @mendesfabio

About You:

[bleu

](https://github.com/bleu-studio) collaborates with companies and DAOs as a web3 technology and user experience partner. We're passionate about bridging the experience gap we see in blockchain and web3.

Additional Links:

Our work for CoW so far:

- [CoW] AMM Deployer:

a Safe app to deploy new CoW AMM pools from a Safe Wallet.

- [CoW] MilkmApp:

a Safe App designed for creating and managing Milkman orders within the CoW ecosystem. It empowers DAOs to sell tokens while deferring the price determination to execution time, leveraging price checkers for optimal pricing strategies.

- [CoW] Python SDK (ongoing):

we're helping CoW put together a Python SDK to provide for developers for query on-chain data, managing orders, and integrating with the CoW Protocol's smart contracts.

- [CoW] Stop-Loss (closing):

a Safe app UI to allow for multisig wallets to fully manage stop-loss orders.

- [CoW] Have I been MEV'd:

a suite of bots and a web application integrated with the ZeroMEV API to inform the Web3 community about potential MEV losses.

Grant Category:

User interface and user experience (UI/UX)

Grant Description:

The purpose of this grant is to develop a set of hooks dApps for theCoW Swap Hook Store. The Hook Store is a feature within CoW Swap that allows users to add pre and post hooks to their swaps. Currently, the Hook Store offers a limited number of hook dApps, including a transaction builder and a Gnosis rewards claim function. Our goal is to expand this selection and facilitate a wider range of DeFi workflows.

To achieve this, we plan to integrate the recently developed cow-shed

and weiroll

tools, enabling the creation of more complex hooks that can handle permissions.

For each hook, we aim to deliver a complete integration into CoW Swap, with the final code merged into the codebase. We have already initiated collaboration with the CoW Swap frontend team to ensure smooth implementation.

In preparation for this project, we have:

1. Drafted mockups and transaction flows for the proposed hooks (availablehere).
2. Developed a proof-of-concept (PoC) for an exit and swap operation (demo availablehere)

Our objective is to significantly enhance the functionality of the CoW Swap Hook Store, providing users with a more versatile and powerful set of tools for managing their DeFi operations. Additionally, we aim to provide valuable feedback to the CoW team based on our development experience, contributing to the ongoing improvement of the CoW Hooks developer experience and expanding its future possibilities.

Grant Goals and Impact:

This project aims to enhance the CoW Swap Hook Store by developing hooks for DeFi operations. This will significantly improve CoW Swap's versatility and enhance user experience. Ultimately, these improvements will position the CoW Swap Hook Store as a more complete programmatic order tool, sharing the potential and versatility of the CoW with more devs.

Milestones:

Milestone	Due Date	Payment (32k xDAI + 32k COW vested)
UI/UX Validation	1 week	2k xDAI + 2k COW vested
Claim Airdrop	2 week	4k xDAI + 4k COW vested
Hooks Setup	3 weeks	6k xDAI + 6k COW vested
Liquidity pool hooks	3 weeks	6k xDAI + 6k COW vested
Lending hooks	5 weeks	10k xDAI + 10k COW vested
Vesting hooks	2 weeks	4k xDAI + 4k COW vested

UI/UX validation (1 week)

Since the end goal of this project is to achieve a production-ready feature on the CoW Swap, the agreement on the UI/UX is essential. For that, the bleu designer will create the designs and get validation from the CoW Swap team.

Claim Airdrop (2 weeks)

As a suggestion of Anxo, from the CoW Swap FE team, this pre-hook can be used for claiming regular airdrops or encouraging the usage of CoW in a new network. We know that each airdrop can have a different way of claiming and we want to support the same used for COW in the past. If a new airdrop is launched in the future but using another interface, we're open to integrating it, but it's outside the scope of this grant.

Hooks Setup (3 weeks)

While the current state of the hook store interface and UX is good, a few adjustments are still needed to cover the use cases presented in this proposal. We believe that working on this setup first will enhance the UI's usability.

- Publish cow-shed

and weiroll

as packages to facilitate their usage on the front end. In addition to current features, these packages will include functions for signing. * This package will include all the ts features of this repo.

- All code for this will be integrated on the same repo and publish on npm

.

- We will add a useCowShed

hook that will receive the calls

to be encoded in the cow-shed

transaction. To do that, a signature

will be needed, to do that, this hook should receive the signer interface.

- This package will include all the ts features of this[repo](repo).

- All code for this will be integrated on the same repo and publish on npm

.

- We will add a useCowShed

hook that will receive the calls

to be encoded in the cow-shed

transaction. To do that, a signature

will be needed, to do that, this hook should receive the signer interface.

- Add an interface to hooks for setting up transactions to be executed before the swap, as some hooks require extra transactions before execution.
- Include a setupTransactions

attribute into the CowHookCreation

interface.

- Include this setup transactions on the order signing flow by integrating it on the SwapFlow

or adding a new flow to handle orders with hooks.

- Include a setupTransactions

attribute into the CowHookCreation

interface.

- Include this setup transactions on the order signing flow by integrating it on the SwapFlow

or adding a new flow to handle orders with hooks.

- Enhance Tenderly simulation for complex hook orders

- Develop a complete settlement transaction simulation on Tenderly (mocking the swap using the limit price) or/and:

- Enhance individual hook simulations by considering extra transactions and modifications to previous hook states.

- To achieve this, we'll utilize the simulation-bundle

feature of the Tenderly API.

- Develop a complete settlement transaction simulation on Tenderly (mocking the swap using the limit price) or/and:

- Enhance individual hook simulations by considering extra transactions and modifications to previous hook states.

- To achieve this, we'll utilize the simulation-bundle

feature of the Tenderly API.

- Implement a feature to modify swap/previous hooks recipient to make the settlement transaction more efficient.
- Example: In the swap and join pool flow, cow-shed

needs to have access to the pool token to perform the join action for the user, so is more efficient to set the swap recipient as the cow-shed

proxy.

- Explanation to the user what parameters are being changed and why via tooltips;
- Capability of changing the swap/previous hook parameters;
- Show each address token difference after the entire execution.
- Use the swap

recipient as the last hook recipient to make sure it receives the tokens on the end.

- Example: In the swap and join pool flow, cow-shed

needs to have access to the pool token to perform the join action for the user, so is more efficient to set the swap recipient as the cow-shed

proxy.

- Explanation to the user what parameters are being changed and why via tooltips;
- Capability of changing the swap/previous hook parameters;
- Show each address token difference after the entire execution.
- Use the swap

recipient as the last hook recipient to make sure it receives the tokens on the end.

- Add security validation to verify if the cow-shed

proxy is safe. * For some hooks, it will be necessary to pre-approve/interact with the proxy address, which might not be created yet. What should raise warnings on the wallets. We're thinking about strategies for how to work around this issue.

- For some hooks, it will be necessary to pre-approve/interact with the proxy address, which might not be created yet. What should raise warnings on the wallets. We're thinking about strategies for how to work around this issue.
- Enable each hook dApp to access the updated state of the blockchain considering all the previous hooks and modifications due to setup transactions. The updated state must consider:
- Token balances on each address;
- Lending balances/health factor;
- To do that, we aim to improve the CowHookDetails

interface with extra data.

- Token balances on each address;
- Lending balances/health factor;
- To do that, we aim to improve the CowHookDetails

interface with extra data.

Liquidity Pool Hooks (3 weeks)

The main use cases for this hook are single-side add or remove liquidity without impacting the pool price, and migrating liquidity from one pool to another. To achieve this, add (pre) and remove (post) hooks will be created. This quote considers the integration of two protocols and we suggest Uniswap V2 and the recently launched CoW AMM built on Balancer.

[

image

1406×430 77.8 KB

](https://europe1.discourse-cdn.com/flex013/uploads/cow/original/2X/9/987ff9723efdaaa5472e1f79c7fd1323606bfd28.png)

Lending Hook (5 weeks)

Similar to the Liquidity Provider Hooks, the Lending hooks will facilitate the management of positions on Lending protocols. We will integrate hooks to execute withdrawals, supplies, borrows, and repayments. All hooks will be added as pre and post-actions and can be combined. This deadline covers the integration of one protocol and we suggest Aave. For all hooks, the health index will be displayed considering all previous hooks, and a warning will be shown if a hook causes the health index to fall below 1.

[

image

1408×436 58.6 KB

](https://europe1.discourse-cdn.com/flex013/uploads/cow/original/2X/6/6e211019a2ae0b69384cb9f9d91e47591c88e10d.jpeg)

Vesting Hooks (2 weeks)

Two hooks will be created: one to claim an existing vesting contract using a pre-hook, and another to use the output of the swap to create a new vesting contract. The claim hook is simpler and doesn’t require cow-shed

for execution. We suggest the integration with Llama Pay.

[

image

1412×786 56.5 KB

](https://europe1.discourse-cdn.com/flex013/uploads/cow/original/2X/1/17ab356b1177f11c5c284ce04e2669e3b89382aa.jpeg)

Funding Request:

We propose that milestone payments be released upon each milestone’s approval.

Budget Breakdown:

The budget includes the hourly rates of a developer during the execution and a project manager on a need basis. The xDAI part of the budget shall be paid after each milestone’s completion and the COW shall be vested over 1 year to cover diluted maintenance and related costs for the same period.

Gnosis Chain Address (to receive the grant):

0x554866e3654E8485928334e7F91B5AfC37D18e04

Other Information:

- All the code will be open-source from day 0. We’re open to feedback during PRs as well;
- We’re happy to answer any questions and are open to feedback about this proposal;
- weiroll contract isn’t audited yet, so if an audit is needed it is outside the scope of this grant;
- Since post-hooks aren’t enforced there is a possibility to some tokens stays on the cow-shed

proxy. Only the user can interact with its own proxy. A UI might be useful to handle this case, however we didn’t quote it yet because its specifications are still to be defined.

- Since this project will be developed on the cowswap

codebase, we anticipate a need for closer collaboration between bleu and the CoW Swap team. We’ve already begun this process by validating and discussing each point of this proposal with them. All work developed by bleu will be reviewed by the CoW Swap team. To ensure this process runs as smoothly as possible, we recognize that communication is key. We’ll maintain close contact, validating technical decisions and seeking guidance during the early stages of the project.

Terms and Conditions:

By submitting this grant application, I acknowledge and agree to be bound by the CoW DAO Participation Agreement and the CoW Grant Terms and Conditions.