Grant Title:

Milkman 2.0

Author:

@charlesndalton

About You:

Former Yearn dev who wrote Milkman

Grant Category:

Developer tools (SDK)

Grant Description:

The core Milkman smart contract was completed in December 2022. Since then, due in large part to the business development efforts of the CoW core contributors, things have gone well. In February 2023, ENS used Milkman to swap 10,000 ETH (~$16.5M at the time) into USDC to cover their payroll for the next 2 years. In July 2023, Aave used Milkman to swap ~$2M of WETH and BAL into the WETH/BAL Balancer pool token (BPT). Fermin Carranza from Llama (which helps manage Aave's treasury) has stated that he was pleased with the recent swap and is working on a tool that will allow Aave to use Milkman to swap assets within Aave's treasury, which will translate into more order flow for CoW protocol.

There were a few take-aways from these experiences:

- Requiring a user to specify a price checker is cumbersome. Requiring a user to manually encode bytes to be passed to the price checker is even more cumbersome.
- Brownie, the original framework used to write Milkman's tests, makes it hard for developers to write their own price checkers. I picked Brownie over Foundry because at the time there was no Solidity HTTP client. But now, there is.
- Sometimes, as was the case for Aave, users of Milkman will prefer a dedicated support engineer. This is especially true when the user requires a custom price checker.

The goal of this grant proposal is threefold: (1) to fund solutions to these former two problems, (2) to fund further improvements to Milkman, including more documentation and price checkers, and (3) to come to an agreement on compensation for my continuing support of those who are integrating CoW protocol via Milkman. I'm going to divide the proposal into these categories.

Grant Goals and Impact:

1: Fund solutions for DevEx woes

1.1: Migrate testing from Brownie to Foundry

Brownie was originally chosen over Foundry because it was impossible to make HTTP

requests from Foundry tests at the time. Now, withsurl,

this is no longer the case. Migrating to Foundry would provide the following benefits:

- Make it easier for other developers to write their own price checkers, since far

more developers are comfortable with Foundry than Brownie

- Escape Brownie, which is beneficial because Brownie maintenance has slowed down

and Brownie has become less dependable. For example, tests will pass on my machine

but won't pass on @fleupolds

'S machine and won't even complete in the CI. For context, the last

commit to Brownie master was on Jun 12th, and it had 6/15 failing checks.

- Speed up the test suite by a factor of ~10x (my personal experience with migrating

from Brownie to Foundry, YMMV)

Felix has expressed that he would like Milkman to migrate to Foundry, so one of the

tasks of this grant would be doing this migration. I estimate that this will take

me 1 development day, or 8 hours.

1.2: A wrapper contract

Milkman has the following interface:

function requestSwapExactTokensForTokens(uint256 amountIn, IERC20 fromToken, IERC20 toToken, address to, address priceChecker, bytes calldata priceCheckerData) external;

It would make for a much

better DevEx if we could have the interface look like

the following:

function requestSwapExactTokensForTokens(uint256 amountIn, IERC20 fromToken, IERC20 toToken, address to) external;

This is important given that a developer is more likely to use Milkman, and thus

to route their order flow through CoW protocol, if it's easier to use.

The idea here would be to create an opinionated wrapper contract that decides

which price checkers to use on the fly. If interested, technical details can

be found here. I estimate that this

wrapper contract will take me 4 development days, or 32 hours.

2: Further improvements to Milkman

2.1: More documentation

While I'm happy to help support Milkman's users, ideally they would be able to

figure out everything themselves. It's been made apparent to me that the current

docs are insufficient for this. This deliverable would be more comprehensive

documentation that allows developers to easily jump into using Milkman

or developing their own price checker. I was thinking a GitBook, but open
to ideas if there's a better way. I estimate that this will take me 3
development days, or 24 hours.

2.2: Limit orders

Since Milkman went live, CoW protocol has introduced limit orders. These can be
used to solve a griefing vector,
that could end up causing the user to lose ~0.5% - ~1% of their trade value.
We were aware of this problem going into production and to our knowledge noone has
exploited it, but doing so would be fairly trivial and it would probably be best
to close off this gap.
To solve this problem, I can write a new set of price checkers that enforce 0 fees
and modify the milkman bot accordingly. Between building and testing, I estimate that this
will take me 1.5 development days, or 12 hours.

2.3: Multi-way price checkers

It may be useful to have price checkers that themselves check other price
checkers. This would allow a user to specify something like “ensure that
the amount out is at least 99% of what I would get from Uniswap AND at least
99% of what Chainlink would provide.” The benefit here is that even if one
of the two has problems (e.g., Chainlink oracle nodes gone rogue), the user
still gets assurances that their order will be executed gracefully.
I think the writing of the code might not take much time; what will take more
time is figuring out how these should work architecturally to preserve modularity.
I estimate that this will take me 1.5 development days, or 12 hours.

3: Supporting Milkman's users

More documentation and migrating the tests from Brownie to Foundry will make it
much easier for developers to get onboarded without any help from the CoW side.
Still, while Milkman is still in its youth, projects are likely to want some
help to confirm that they are doing the right things. Some projects may also
want help developing their own price checker, as Aave did. This leg of the grant
would fund a support engineer, me :).
So far, I estimate that I've put 10 hours into supporting users.

Milestones:

For the purposes of the timeline, I define t
as the time that the grant is
accepted.

Milestone 1: Tests migrated from Brownie to Foundry

Due date: t + 1 week

Milestone 2: Limit orders

Due date: t + 2 weeks

Milestone 3: More documentation

Due date: t + 4 weeks

Milestone 5: Multi-way price checkers

Due date: t + 5 weeks

Milestone 4: Wrapper contract

Due date: t + 8 weeks

The wrapper contract would need to be reviewed by someone on the CoW team, likely

Nick, to be considered completed.

Funding Request:

These are the time estimations for each task:

- Test migration: 8 hours
- Limit orders: 12 hours
- More documentation: 24 hours
- Wrapper contract: 32 hours
- Multi-way price checker: 12 hours

I propose that the grant pay at $250 per estimated hour. This number was derived

from:

- Googling 'what is a typical software engineer consulting rate per hour' and

looking at the top three responses:Uptech, Cleveroad, and FullStack Labs.

- Looking at comparable funded CoW grants
- Converting my past salaries into an hourly figure

The estimations of how much time tasks will take are based on my knowledge and my experience. If someone from the COW team thinks that they can accomplish a task in less time, I am happy to have them take that task and we can remove it from the grant.

If I was the one assigned to complete all of them, this would be 88 development hours or $22,000. As far as the split between xDAI and COW, I'm boolish on COW and would be happy with up to 50% of the grant being paid in vCOW.

As for the support, what I suggest is that I am paid at the normal rate, but with the amount of spend capped at 8 hours a month

. On some months, I may end up spending more than 8 hours on supporting users, and in these months the 8 hours * hourly rate would be like my salary for that month. This structure ensures two things:

- When no support is needed, no spending is required
- CoW can predict its spending, since no month will consume more than 8 hours *

hourly rate ($2000).

If possible, it would also be appreciated if we could do retroactive compensation for the 10 hours thus far.

Budget Breakdown:

All development

Gnosis Chain Address (to receive the grant):

0x8b4eA96254229f4E93538319010A17A7A5fBf864 (charlesndalton.eth)

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.