Simple summary

Introduce to the community a simple GhoDirectMinter

facilitator smart contract to adopt if required for different initiatives appearing in this forum during the last months.

The codebase can be found (temporarily until gov approval) on:

https://github.com/bgd-labs/GhoDirectMinter

Motivation

Recently, various proposals/discussions have targeted GHO, including listings via minting (partially or total) like on v3 Lido HERE or minting + bridging via cross-chain GHO likeHERE.

After following those posts here on the forum, we perceived the need for some type of unification and simplification of the infrastructure required for the DAO to operate efficiently, more specifically on the minting + supply to Aave side of a GHO facilitator, and base architecture.

The motivation of this post is to propose a GHO mint + supply to Aave facilitator smart contract to move on operational requirements of the DAO as soon as possible, with simplicity as a design principle.

This post deviates from the GHO Facilitator Onboarding Process and Application, as this is just a proposal for the infrastructure of contracts to use, not the activation of a specific instance.

Specifications

While the aforementioned proposals differ in the suggested parameters, what they have in common is the goal of allowing the DAO to directly mint to the pool, while at the same time allowing other entities to supply GHO as well.

This approach is fundamentally different from the existing GHO integration on the Ethereum Main instance, given that in the original implementation, there is no AToken, no index accrual, and all fees go to the treasury.

To satisfy these different goals, we developed a new GHODirectMinter

facilitator smart contract.

[

GhoDirectMinter diagram

1920×1145 71.3 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aave/original/2X/6/6243520ba278a5550578612a0184b2c277956e9d.jpeg)

High level design of the GhoDirectMinter

The development of the GhoDirectMinter was done with these principles:

- Following the facilitator ethos

. The facilitators architecture proposed by Aave Labs on the GHO release had as its main objective modularity, for multiple use cases both on Ethereum or any other chain. Facilitators should not be monolithic, but compact and use-case specific, fast to move forward.

- Simplicity

. The goal of the development was to fulfill the current needs of the DAO while having a contract very simple to audit. No extra overhead to adapt to hypothetical use cases, that can be accommodated as they appear, via upgrade or deployment of new facilitators.

In terms of logic, and access control, the GHODirectMinter works the following way:

- Upgradeable contract by Aave governance.
- 3 access control roles, with an example configuration taking as reference minting GHO on v3 Lido:
- upgradeability admin

(proxy admin): Aave DAO, but via its ProxyAdmin contract.

- owner

: Aave DAO, via the Level 1 Executor.

- guardian

: GHO council, sometimes known as GHO steward in the forum.

- upgradeability admin

(proxy admin): Aave DAO, but via its ProxyAdmin contract.

- owner

: Aave DAO, via the Level 1 Executor.

- guardian

: GHO council, sometimes known as GHO steward in the forum.

- 3 external functions:
- mintAndSupply()

. * Mints GHO and supplies it into the target Aave Pool.

- To be used by the guardian to choose when liquidity gets minted and supplied, and how much.
- It ignores and supply cap on the Aave pool, this way giving “priority” to the DAO supplying liquidity versus others.
- Mints GHO and supplies it into the target Aave Pool.
- To be used by the guardian to choose when liquidity gets minted and supplied, and how much.
- It ignores and supply cap on the Aave pool, this way giving “priority” to the DAO supplying liquidity versus others.
- withdrawAndBurn()

. * Withdraws GHO from the target Aave Pool and burns it.

- To be used by the guardian to choose when liquidity gets withdrawn and burnt, and how much.
- Withdraws GHO from the target Aave Pool and burns it.
- To be used by the guardian to choose when liquidity gets withdrawn and burnt, and how much.
- transferExcessToTreasury()

. * Transfers to the configured Aave DAO treasury (e.g. Collector Ethereum) any excess of GHO aToken, majorly the yield accrued from supplying into the pool.

- Permissionless, oriented to be automated.
- Transfers to the configured Aave DAO treasury (e.g. Collector Ethereum) any excess of GHO aToken, majorly the yield accrued from supplying into the pool.
- Permissionless, oriented to be automated.
- mintAndSupply()

. * Mints GHO and supplies it into the target Aave Pool.

- To be used by the guardian to choose when liquidity gets minted and supplied, and how much.
- It ignores and supply cap on the Aave pool, this way giving “priority” to the DAO supplying liquidity versus others.
- Mints GHO and supplies it into the target Aave Pool.
- To be used by the guardian to choose when liquidity gets minted and supplied, and how much.
- It ignores and supply cap on the Aave pool, this way giving “priority” to the DAO supplying liquidity versus others.
- withdrawAndBurn()

. * Withdraws GHO from the target Aave Pool and burns it.

- To be used by the guardian to choose when liquidity gets withdrawn and burnt, and how much.
- Withdraws GHO from the target Aave Pool and burns it.
- To be used by the guardian to choose when liquidity gets withdrawn and burnt, and how much.
- transferExcessToTreasury()

. * Transfers to the configured Aave DAO treasury (e.g. Collector Ethereum) any excess of GHO aToken, majorly the yield accrued from supplying into the pool.

- Permissionless, oriented to be automated.
- Transfers to the configured Aave DAO treasury (e.g. Collector Ethereum) any excess of GHO aToken, majorly the yield accrued from supplying into the pool.
- Permissionless, oriented to be automated.

Regarding a potential flow of setup & operation for a hypothetical GhoDirectMinter facilitator for GHO on v3 Lido (via governance proposal):

1. Deploy proxy contract + implementation of the GhoDirectMinter.
2. Grant RISK_ADMIN role to the GhoDirectMinter proxy, in order to skip supply caps on mintAndSupply().
3. Call addFacilitator()

on GHO with the desired capacity for the facilitator.

1. Optionally, already do an initial minting in the target pool via governance proposal.
2. Once the system is active post-governance, the configured guardian entity can operate up to the capacity configured, by minting+supply and withdraw+burn.
3. Periodically, transferExcessToTreasury() can be triggered via an Aave Robot automation or Dolce Vita to send the accrued aGHO yield to the Collector. This is however not so critical, as the yield will simply keep accruing in the GhoDirectMinter, where is perfectly safe and optimal.

Props to the other contributors who helped us understand the potential use cases and DAO operational needs, especially @TokenLogic @ACI; and all other participants in the GHO discussion and contributions in this forum.

Next steps

- After 3 days of discussion in this forum, create an ARFC Snapshot for the community to pre-approve the facilitator contract.
- If/when the ARFC Snapshot passes, coordinate for review of the facilitator contract by Certora.
- In parallel with security procedures, sync with other service providers to create the proposal for the activation of 1 or n facilitators needed, using the GhoDirectMinter. Following the GHO Facilitator Onboarding Process and Application.
- Also in parallel with security procedures, we will coordinate with other contributors on the facilitator smart contracts (e.g. @AaveLabs) for the best location to place the code of the facilitator.