Summary

Based on community feedback, we plan to update the StrategicAssetManager by:

- Adding vlAURA support based on community temp check
- Assigning the guardian role to the GHO Liquidity Committee multisig based on community temp check

Note that updating the StrategicAssetManager only gives the Aave community the option to support vlAURA; the final decision is up to the community.

Context

We introduced the StrategicAssetsManager contract in July 2023, with the first version supporting only veBAL. From Llama's side, the code supporting veBAL position is already implemented via the StrategicAssetManager contract and is ready to be deployed.

However, as the TEMP CHECK to update the Balancer ecosystem holdings was approved, and considering that the Aave DAO already acquired AURA tokens OTC from Olympus DAO, this proposal aims to add vlAURA support in addition to the existing veBAL support.

Motivation

Following the recent impact of AIP-42 in the Aura governance about the emission power, there are considerations to switch holdings from B-80BAL-20WETH to AURA locked for vlAURA, enabling to retain the maximum emission power.

The goal of this contract remains the same, to create a practical means to manage the DAO's voting power, which can either be managed by AIPs or by an elected committee (AssetManager).

Unlike veBAL, the vlAURA holdings don't need to be frequently relocked to retain the maximum voting power as there is no decay. Gauge votes are created every round (bi-weekly votes) and are published off chain on this snapshot space.

The Aave DAO also has an opportunity to participate in both Aura & Balancer governance decisions as well on this snapshot space. Aave's Snapshot will be used for updates on the voting strategy.

The contract is upgradeable and controlled by Aave's Short Executor role via governance. Each function on the contract can be implemented via the ShortExecutor whilst also having the ability to assign/remove an Guardian role to a community elected address.

We propose to enable the GHO Liquidity Committee to be assigned as the Guardian role to interact with the SAM contract.

Functions added in the SAM v1 to support veBAL:

- Lock/relock B-80BAL-WETH
- Participate in Balancer gauge votes
- Delegate Balancer governance voting rights
- Allocate veBAL boost
- Delegate veBAL boost
- Allow & Sell veBAL boost on Warden
- Claim veBAL protocol fees & boost rewards
- Transfer Assets

Extra functions to be added in the SAM v2 to add vlAURA support:

- Allow & lock AURA
- Process Expired Locks (Relock or Withdraw Expired)
- Claim vlAURA protocol fees (GetReward)
- Delegate vlAURA voting/gauge power

- Emergency Withdraw

Specifications

The below details the functions on the contract with a brief description on how each works:

veBAL functions:

function withdrawERC20(address token, address to, uint256 amount) external onlyOwner

Sends ERC20 tokens to an address. Withdrawal mechanism.

function updateGuardian(address _manager) external onlyOwner

Updates guardian role, which in this contract functions as a strategic asset manager. Inherited from OwnableWithGuardian.

function transferOwnership(address _owner) external onlyOwner

Updates the owner of the contract. Inherited from Ownable.

function buyBoost(

address underlying,

address delegator,

address receiver,

uint256 amount,

uint256 duration

) external onlyOwnerOrManager

Purchase boost to incentivize rewards earned by locking (up to 2.5x of earnings).

For more info see: Boost Market - Docs

The idea is to increase the yield in the provided liquidity. For example, pay 10 BAL to boost rewards in a veBAL pool up to 2.5x times, to earn more BAL in return.

function sellBoost(

address underlying,

uint256 pricePerVote,

uint64 maxDuration,

uint64 expiryTime,

uint16 minPerc,

uint16 maxPerc,

bool useAdvicePrice

) external onlyOwnerOrManager

Owner of veToken allows others to incentivize their liquidity pools by selling boost. The price can be chosen by the user, or by setting useAdvicePrice, let Warden determine the price.

The seller of boost receives the native token.

function updateBoostOffer(

address underlying,

uint256 pricePerVote,

uint64 maxDuration,

uint64 expiryTime,

uint16 minPerc,

uint16 maxPerc,

bool useAdvicePrice

) external onlyOwnerOrManager

Allows the user to update an existing offer to sell boost.

function removeBoostOffer(address underlying) external onlyOwnerOrManager

Removes a boost offer.

function claim(address underlying) external onlyOwnerOrManager

Claim rewards earned by selling boost.

function setSpaceId(address underlying, bytes32 _spaceId) external onlyOwnerOrManager

Sets the spaceID that's used by protocol on Snapshot for voting. For example, "balancer.eth" is Balancer's spaceId on Snapshot.

function setDelegate(

address underlying,

address newDelegate

) external onlyOwnerOrManager

Delegate tokens so they can vote on Snapshot.

function clearDelegate(address underlying) external onlyOwnerOrManager

Remove the active delegate.

function setLockDuration(

address underlying,

uint256 newLockDuration

) external onlyOwnerOrManager

Set the lock duration to a specific time. For example, max lock for veBAL is 1 year, so set to 1 year (or less).

function lock(address underlying) external onlyOwnerOrManager

The main function for veBAL. Initially, it locks the B-80BAL-20WETH token to receive veBAL. (This contract needs to be allow-listed by Balancer prior to calling or it will fail).

On subsequent calls (for example, weekly) it extends the lock duration once again. The voting % available per token is dependent on the locking duration.

If the locking duration is 6 months and the maximum duration is 1 year, then the voting weight is only half. This function also locks more of the native token held by StrategicAssetsManager available on the contract.

function unlock(address underlying) external onlyOwnerOrManager

Unlocks the veToken in order to receive the underlying once again. Lock duration needs to have passed or the transaction will revert.

function setGaugeController(address token, address gaugeController) public onlyOwnerOrManager

Sets the address that handles gauges for veTokens.

Curve docs on liquidity gauges: The Curve DAO: Liquidity Gauges and Minting CRV — Curve 1.0.0 documentation

The main concept here is that the ecosystem rewards liquidity providers by rewarding them with token emissions. These tokens are distributed according to which gauges receive the

most votes.

function voteForGaugeWeight(

address token,

address gauge,

uint256 weight

) external onlyOwnerOrManager

Utilizing the veToken holdings, the DAO can vote to redirect emissions to the DAO's own gauge. Here, by voting for the DAO's gauge, and also purchasing boost, the DAO can expect to earn a lot more BAL rewards over time than just by holding a veToken for example. ti

vlAURA functions

function lockVLAURA(uint256 amount) external onlyOwnerOrGuardian

Locks AURA into vlAURA (if not locked before).

function claimVLAURARewards() external onlyOwnerOrGuardian

Claims rewards accrued by locking vlAURA.

function delegateVLAURA(address delegatee) external onlyOwnerOrGuardian

Delegates vlAURA for voting purposes.

function relockVLAURA() external onlyOwnerOrGuardian

Relocks vlAURA that has been previously locked.

function unlockVLAURA() external onlyOwnerOrGuardian

Unlock vlAURA position into AURA. Lock period needs to have passed or it will revert.

function emergencyWithdrawVLAURA() external onlyOwnerOrGuardian

Emergency function to exit a position if the AURA system is shut down.

Assign the guardian role to the following address

GHO Liquidity Committee Multisig - eth: 0x205e795336610f5131Be52F09218AF19f0f3eC60

Disclaimer

Llama is not compensated by anyone outside of Aave to submit this proposal.

Copyright