Bookkeeper registry

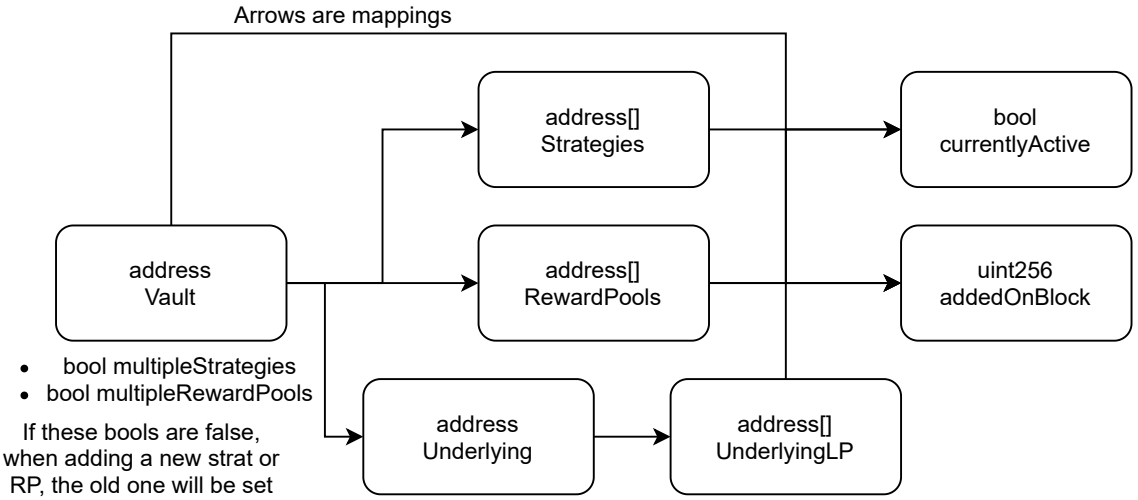
Every contract is part of a set containing a vault, (multiple) strategies, (multiple) reward pools, an underlying asset and a LP for the underlying asset. In the registry these contracts and the links will be stored. The diagram below shows the mappings between addresses and other parameters. Any address that is added will get an added on block stamp. This allows to store historical strategies/reward pools/underlying LPs. It is assumed that a vault will only ever have 1 underlying token. This can be changed if this seems useful.

As there might be future possibilities for pools to have multiple strategies and/or reward pools, this is incorporated in the design. To this extend a bool "currentlyActive" is stored. This allows multiple strategies to be active for a single vault.

Every address mapped to "type" which returns string: vault, strategy, rewardPool, underlying or underlyingLP. Not completely sure if strings work well to return. Could use index numbers. Adding new types would be possible, and it might also be possible to allow new added types to also be mapped to the vaults. I have to think of the technical application of this.

I was thinking to also have the possibility to add other contracts of the Harvest ecosystem. These are not directly mapped to vaults or strategies, but can be stored to keep a complete registry of all contracts and allow people to verify whether certain contracts belong to our system.

I think generally it would be nice to have the possibility to pull arrays of addresses per type, so vaults/strategies/etc. Could be possible to also categorize in other ways. Group vaults into stable/uniswap/etc.



- bool multipleStrategies
- bool multipleRewardPools

If these bools are false, when adding a new strat or RP, the old one will be set to inactive. When they are true adding new strat or RP will not set old one to inactive. These will need to be set to inactive manually when they are stopped to be used.

Functions

- **addVault(**vault, rewardpool, underlyingLP, multipleStrats, multipleRPs)
- **addStrategy(**strategy)
- **addRewardPool(**rewardPool)
- **changeUnderlyingLP(**underlying, underlyingLP)
- **getType**(address) -> returns address type (vault/strat/RP/etc)
- **getHistoricalSet**(vault) -> returns vault, all strategies, all RPs, Underlying and all Underlying LP. Could be made to take any contract in the set as input. (Should also return the addedOnBlock values for all addresses.)
- **getActiveSet**(vault) -> returns vault, active strategies, active RPs, Underlying and Underlying LP. Could be made to take any contract in the set as input. (I think addedOnBlock values not necessary here.)
- **deactivate**(address) -> set currentlyActive to false
- **activate**(address) -> set currentlyActive to true
- **remove**(address) -> remove address from the registry. Removing vault will remove the whole set. Underlying cannot be removed. A vault should always have at least one strategy/RP/underlyingLP.