kETH Properties

We will focus on specifying high level key properties. Most revert conditions will for example not be in scope for this document.

Properties of $KETHVault$

Invariants

- $totalSupply() \leq maxSupply$

Function properties of $deposit$

In pre state, set $A = totalAssets()$ and $V = assetValue(a, x)$. Suppose $y = deposit(a, x, r, b)$ is applied to valid input, then

- $totalAssets()$ will increase by V modulo slippage
- In case $a \neq dETH$ and $b = true$ and $kETHStrategy.autoSellFor() \neq 0$, then $kETHStrategy.assetValue(kETHStrategy.reserves(kETHStrategy.autoSellFor()))$ will increase by at least $V - 0.011\ ether$ modulo slippage
- In case $a = dETH$, $kETHStrategy.assetValue(kETHStrategy.reserves(savETH))$ will increase by at least V modulo slippage
- In case $a \neq dETH$ and $b = false$ and $kETHStrategy.autoSellFor() \neq 0$, then $kETHStrategy.assetValue(kETHStrategy.reserves(a))$ will increase by at least V modulo slippage
- $y = amountToShare(V)$, modulo slippage
- $totalSupply()$ will increase by y
- $balanceOf(r)$ will increase by y

Function properties of $withdraw$

In pre state, let $A = totalAssets()$ and $T = totalSupply()$. When calling $withdraw(x, r)$ on valid input,

- $totalAssets()$ will decrease by $shareToAmount(x)$ modulo rounding errors
- $strategy.dETH().balanceOf(r) + r.balance + strategy.giantLP().balanceOf(r)$ will increase by $shareToAmount(x)$ modulo slippage and bounded asset loss
- $totalSupply()$ will decrease by x
- $balanceOf(r)$ will decrease by x

Function properties of $shareToAmount$ and $amoutToShare$

The same as those for $DETHVault$. See the Properties of $DETHVault$ section.

Function properties of $setStrategy$

In pre state let $R_a = strategy.reserves(a)$ and $T = strategy.totalAssets()$ and set $s = strategy$. Assuming that $s \neq s'$, after successful invocation of $setStrategy(s')$, for each a in $holdingAssets()$,

- $s.reserves(a) = s.totalAssets() = 0$
- $strategy = s'$
- $s'.balance(a)$ has increased by R_a
- $s'.reserves(a) = s'.balance(a)$
- $s'.totalAssets()$ has increased by T

Properties of $SavETHManagerHandler$

The properties of $SavETHManagerHandler$ are properties of both $KETHStrategy$ and $DETHVault$.

Invariants

- $isolatedKeys[i] \neq 0$ if and only if $i < numOfIsolatedKeys$
- For each $i < numOfIsolatedKeys$, with $s = isolatedKeys[i]$, $savETHManager.associatedIndexIdForKnot(s.blsPublicKey) > 0$
- The BLS public keys $isolatedKeys[i].blsPublicKey$ for $i < numOfIsolatedKeys$ are pairwise distinct

Function properties of $isolateKnotFromOpenIndex$

Suppose non-reverting call $isolateKnotFromOpenIndex(h, k)$,

- $numOfIsolatedKeys$ is increased by one
- $\_reserves[savETH]$ is decreased by at least $savETHManager.dETHToSavETH(savETHManager.KNOT\_BATCH\_AMOUNT)$
- $isolatedKeys[numOfIsolatedKeys - 1] = (h, k)$

Function properties of $addKnotToOpenIndex$

In pre state set $s = isolatedKeys[i]$ and $t = isolatedKeys[numOfIsolatedKeys - 1]$. After successful call $addKnotToOpenIndex(i)$,

- $numOfIsolatedKeys$ is decreased by one
- $\_reserves[savETH]$ is increased by at least $savETHManager.dETHToSavETH(savETHManager.KNOT\_BATCH\_AMOUNT)$
- $isolatedKeys[i] = t$
- $savETHManager.associatedIndexIdForKnot(s.blsPublicKey) = 0$

Function properties of $rotateSavETH$

In pre state set $s = isolatedKeys[i]$. Assume successful call $rotateSavETH(i,\ h,\ k)$. Except for possible rounding errors, that is having the same effect on $reserves(savETH)$ as the sequence of calls $addKnotToOpenIndex(i)$; $isolateKnotFromOpenIndex(h,\ k)$. Moreover the following properties are satisfied in the post state:

- $isolatedKeys[i] = (h,\ k)$
- $savETHManager.associatedIndexIdForKnot(s.blsPublicKey) = 0$.

Properties of $KETHStrategy$

Invariants

- If $defaultSwapper[i][j] = s \neq 0$ then $swapper[i][j][s] = true$
- If $swapper[i][j][s] = true$ then $s.inputToken() = i$ and $s.outputToken() = j$
- For all $a \neq savETH$, $balance(a) \geq reserves(a)$
- $balance(savETH) + .totalIsolatedSavETH() \geq reserves(a)$
- $reserves(a) > 0$ implies $isHoldingAsset(a)$
- $isUnderlyingAsset(a)$ implies $isHoldingAsset(a)$
- Required holding assets, which swappers may assume
    - $isHoldingAsset(giantLP)$
    - $isHoldingAsset(dETH)$
    - $isHoldingAsset(savETH)$
    - $isHoldingAsset(ETH)$
- For all i satisfying $isHoldingAsset(i)$ and $i \notin \{ETH,\ dETH,\ giantLP,\ savETH\}$, $defaultSwapper[i][ETH] \neq 0$

Comments:

- $isUnderlyingAsset(stETH)$ is unused

Function properties of $assetValue$

For any asset a such that $isHoldingAsset(a)$, the following properties are required

- $assetValue(a,\ 0) = 0$
- $assetValue(a,\ 1\ ether) > 0$
- $assetValue(a,\ x + y) = assetValue(x) + assetValue(y)$, modulo rounding errors
- $assetValue(a,\ x + 1) \geq assetValue(a,\ x)$

Function properties of $invokeSwap$

After successful invocation of $invokeSwap(s,\ a,\ x,\ b,\ M,\ D)$, then

- $assetValue(a,\ reserves(a))$ will decrease by $assetValue(a,\ x)$ plus/minus $0.011\ ether$, modulo slippage
- $assetValue(b,\ reserves(b))$ will increase by at least $assetValue(a,\ x) - 0.011\ ether$ modulo slippage

- $reserves(b)$ will increase by at least M
- $totalAssets()$ is constant, modulo slippage

Properties of $DETHVault$

Invariants

- $dETH.balanceOf(address(this)) \geq reserves(dETH)$
- $savETH.balanceOf(address(this)) + totalIsolatedSavETH() \geq reserves(savETH)$
- $address(this).balance \geq reserves(ETH)$

Function properties of $deposit$

After successful invocation of $deposit(x, r)$:

- $balanceOf(r)$ will increase by $amountToShare(x)$
- $dETH.balanceOf(r)$ will decrease by x
- $reserves(savETH)$ will increase by $savETHManager.dETHToSavETH(x)$

Function properties of $withdrawToETH$

After successful invocation of $withdrawToETH(x, r)$:

- $balanceOf(r)$ will decrease by x
- $r.balance$ will increase by $shareToAmount(x)$
- $reserves(ETH)$ will decrease by $shareToAmount(x)$

Function properties of $withdrawToDETH$

After successful invocation of $withdrawToDETH(x, r)$:

- $balanceOf(r)$ will decrease by x
- $dETH.balanceOf(r)$ will increase by $savETHManager.savETHToDETH(savETHManager.dETHToSavETH(x))$
- $reserves(savETH)$ will decrease by $savETHManager.dETHTOSavETH(x)$

Function properties of $swapETHToDETH$

After successful invocation of $swapETHToDETH(r)$:

- $reserves(ETH)$ will increase by $msg.value$
- $reserves(savETH)$ will decrease by approximately $savETHManager.dETHToSavETH(msg.value)$
- $dETH.balanceOf(msg.sender)$ will increase by approximately $msg.value$.

Function properties of $shareToAmount$ and $amoutToShare$

- $shareToAmount(amountToShare(x))$ is x, modulo rounding errors

- $amountToShare(shareToAmount(x))$ is x, modulo rounding errors
- $shareToAmount(0) = amountToShare(0) = 0$
- $shareToAmount(x + 1) \geq shareToAmount(x)$
- $amountToShare(x + 1) \geq amountToShare(x)$
- $shareToAmount(totalSupply()) = totalAssets()$
- $amountToShare(totalAssets()) = totalSupply()$
- $shareToAmount(x + y) = shareToAmount(x) + shareToAmount(y)$, modulo rounding errors
- $amountToShare(x + y) = amountToShare(x) + amountToShare(y)$, module rounding errors

Properties of $ISwapper$

Let $KETHStrategy\ strategy$ be given. Any $ISwapper$ must satisfy the following. After non-reverting call $swap(a,\ x,\ b,\ M,\ D)$, then

- Sender's balance of a token will decrease in asset value corresponding to $strategy.assetValue(a,\ x) - 0.011\ ether$, modulo a constant bounded slippage
- Sender's balance of b token will increase in asset value corresponding to $strategy.assetValue(a,\ x) - 0.011\ ether$, modulo a constant bounded slippage
- Sender's balance of b token will increase by at least M