Stake

Staking in volatile and stable pools is slightly different, so we'll show you how to do it in turn.

Harvesting (without additional staking) can be done by interacting with the gauge with the combination [VC,AT_MOST,0] without specifying the lp token, which we'll also show as an example below.

1) Staking USDC-ETH LP.

- The only difference between Wombat Pool and Volatile Pool is that Wombat Pool has a separate Gauge contract for each token.
-

```

Copy // Stake involves 2 tokens. LP token and VC because it also trigger harvest of VC token. Token[]memorytokens=newToken; tokens[0]=toToken(IERC20(usdc_eth_lp)); tokens[1]=toToken(IERC20(vc));

VelocoreOperation[]memoryops=newVelocoreOperation; // address usdc_eth_pool = vault.getPair(usdc, eth); // OP type STAKE(GAUGE). For volatile pools, LP contract itself is a gauge contract. so use pool address for relevant pool address. // This is different for Wombat Pool, and I will return to that later. ops[0].poolId=toPoolId(GAUGE,usdc_eth_pool); ops[0].tokenInformations=newbytes32; int128stakeAmount=int128(int256(IERC20(usdc_eth_lp).balanceOf(address(this)))); ops[0].tokenInformations[0]=toTokenInfo(0x00,EXACTLY,stakeAmount); //we don't know how much VC we'd harvest in this action. so just use AT_MOST 0 . ops[0].tokenInformations[1]=toTokenInfo(0x01,AT_MOST,0); ops[0].data=""; returnexecute(tokens,newint128,ops);

```

2) Harvesting

Quite similar to Staking. Just use only VC in the token info.

```

Copy //Harvest Token[]memorytokens=newToken; tokens[0]=toToken(IERC20(vc));

VelocoreOperation[]memoryops=newVelocoreOperation; // Similar with Stake, we use op type STAKE with the pool but giving only VC as token info. // OP type STAKE(GAUGE). For volatile pools, LP contract itself is a gauge contract. so use pool address for relevant pool address. // This is different for Wombat Pool, and I will return to that later. ops[0].poolId=poolId(GAUGE,usdc_eth_pool); ops[0].tokenInformations=newbytes32; //we don't know how much VC we'd harvest in this action. so just use AT_MOST 0 . ops[0].tokenInformations[0]=toTokenInfo(0x00,AT_MOST,0);

ops[0].data=""; returnexecute(tokens,newint128,ops);

```

3) Staking stable LP (Wombat pool)

Let's say you are staking USDC_LP from USDC/USDT/DAI-3pool.

2 differences with volatile pool.

- Use gauge address on poolId instead of lp address.
- LP token is ERC-1155 , not ERC-20. Should wrap into 'Token' accordingly.
-

Step1. Getting gauge address

WombatPool.lpTokens() => bytes32 type for each tokens in the pool lpTokens() function on the LP contract returns all bytes32 type Token data in order of ERC1155's id. Let's say it's USDC-LP.

WombatPool.underlyingTokens(Token) => bytes32 for underlying assets You could use this USDC-LP(bytes32) to get bytes32 representation of the underlying assets like USDC.

Or instead, you could always use toToken function on 'src/lib/Token.sol' directly to wrap token into bytes32 type.

WombatPool.gauges(Token) Finally, you could query gauge address with this underlying token's bytes32.

STEP2. same job as before

feeling a bit complicated for stableswap? Actually, if you already got the gauge address, you could just skip STEP1 and

execute right away just like before.

```

Copy // Major difference between volatile pool is that Wombat pool has a seperate gauge contract for each tokens. // You should interact with gauge, not pool to stake. Token[]memorytokens=newToken; // Instead of wrapping yourself, you could just query bytes32 from pool's lpTokens() function. usdc_id_at_lp=0;// let's assume USDC's id at 3poolLP(ERC1155) is 0. tokens[0]=toToken(ERC1155,usdc_id_at_lp,IERC1155(usdc_lp)); tokens[1]=toToken(IERC20(vc));

VelocoreOperation[]memoryops=newVelocoreOperation; ops[0].poolId=toPoolId(GAUGE,usdc_gauge);
ops[0].tokenInformations=newbytes32;
int128stakeAmount=int128(int256(IERC1155(usdc_lp).balanceOf(address(this),usdc_id_at_lp)));
ops[0].tokenInformations[0]=toTokenInfo(0x00,EXACTLY,stakeAmount);
ops[0].tokenInformations[1]=toTokenInfo(0x01,AT_MOST,0);

ops[0].data=""; returnexecute(tokens,newint128,ops);

```

Last updated4 months ago On this page