Typescript Examples for Integration

```

Copy import{ solidityPack,parseEther }from"ethers/lib/utils"; import{ BigNumber }from"ethers";

exportdefaultasyncfunction(hre) { constgetContractAt=hre.ethers.getContractAt;
constvault=awaitgetContractAt("IVault","0x5BA1C526b79bA08668fa23733E762a2579058697");
constfactory=awaitgetContractAt("ConstantProductPoolFactory","0xb4e79Fc4F09f8126c1a5a862DA536daBEf455b6f");

consttoToken=(spec:string,id,addr:string)=> solidityPack( ["uint8","uint88","address"],
[["erc20","erc721","erc1155"].indexOf(spec),id,addr] ) constpoolId=(i,poolAddress:string)=> solidityPack(
["bytes1","uint88","address"], [i,0,poolAddress] )
constINT128_MAX=ethers.BigNumber.from("170141183460469231731687303715884105727"); consttokenInformation=
(index:number,amountType:string,amount)=> solidityPack(["uint8","uint8","uint112","int128"],[ index, ["exactly","at
most","all","flashloan"].indexOf(amountType), 0, amount ] )

constcompileAndExecute=(value,ops)=>{ consttokenRef=[...newSet(ops.flatMap(x=>x[1].map(i=>i[0])))].sort();
returnvault.execute([ tokenRef, (newArray(tokenRef.length)).fill(0), ops.map(op=>({ poolId:op[0],
tokenInformations:op[1].map(i=>tokenInformation(tokenRef.indexOf(i[0]),i[1],i[2])).sort(), data:[] })) ,{ value } ]) }

constvevc_address="0x1b187693Aa0D260896a804aEe1816eC4ACE56257";

constvc=toToken("erc20",0,"0xB0B88DF91c5f1E48854225a076AA39f0933Ea2F5");
constvevc=toToken("erc20",0,"0x1b187693Aa0D260896a804aEe1816eC4ACE56257");
constusdc=toToken("erc20",0,"0x699a9c41da7721cb53eb99de3fba2a5a6165f711");
constdai=toToken("erc20",0,"0x342712712a2e16b31761291bf1931f5eba2d928d");
consteth="0xeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee";
constINT128_MAX=BigNumber.from("170141183460469231731687303715884105727");

constusdc_eth_pool=awaitfactory.pools(eth,usdc); constvc_eth_pool=awaitfactory.pools(eth,vc);
constdai_eth_pool=awaitfactory.pools(eth,dai); constusdc_eth_lp=toToken('ERC20',0,usdc_eth_pool);

// swap usdc -> eth, exact in awaitcompileAndExecute(0,[ [poolId(0,usdc_eth_pool),[ [usdc,"exactly",parseEther("100")],
[eth,"at most",parseEther("-0.001")], ]] ])

// swap usdc -> eth, exact out awaitcompileAndExecute(0,[ [poolId(0,usdc_eth_pool),[ [usdc,"at most",parseEther("100")],
[eth,"exactly",parseEther("-0.001")], ]] ])

// swap usdc -> eth -> dai, exact in awaitcompileAndExecute(0,[ [poolId(0,usdc_eth_pool),[
[usdc,"exactly",parseEther("100")], [eth,"at most",INT128_MAX] ]], [poolId(0,dai_eth_pool),[ [eth,"all",INT128_MAX], [dai,"at
most",parseEther("-99")], ]] ])

// swap usdc -> eth -> dai, exact out awaitcompileAndExecute(0,[ [poolId(0,dai_eth_pool),[ [eth,"at most",INT128_MAX],
[dai,"exactly",parseEther("-99")], ]], [poolId(0,usdc_eth_pool),[ [usdc,"at most",parseEther("100")], [eth,"all",INT128_MAX] ]]
])

// swap eth -> dai, exact in awaitcompileAndExecute(parseEther("0.001"),[ [poolId(0,dai_eth_pool),[ [eth,"all",INT128_MAX],
[dai,"at most",parseEther("-99")], ]] ])

// swap eth -> dai, exact out, remaining eth will be refunded awaitcompileAndExecute(parseEther("0.001"),[
[poolId(0,dai_eth_pool),[ [eth,"at most",INT128_MAX], [dai,"exactly",parseEther("-99")], ]] ])

// add liquidity and stake usdc-eth == "buying lp token and paying usdc and eth"
awaitcompileAndExecute(parseEther("0.001"),[ [poolId(0,usdc_eth_pool),[// deposit lp [eth,"all",INT128_MAX],
[usdc,"exactly",parseEther("100")], [usdc_eth_lp,"at most",0] ]], [poolId(1,usdc_eth_pool),[// stake
[usdc_eth_lp,"all",INT128_MAX] ]] ])

// add liquidity and stake usdc-eth, using single token awaitcompileAndExecute(0,[ [poolId(0,usdc_eth_pool),[// deposit lp
[usdc,"exactly",parseEther("100")], [usdc_eth_lp,"at most",0] ]], [poolId(1,usdc_eth_pool),[// stake
[usdc_eth_lp,"all",INT128_MAX], [vc,"at most",0]// vc must be included to receive emissions. it will revert otherwise ]] ])

// harvest vc awaitcompileAndExecute(0,[ [poolId(1,usdc_eth_pool),[ [vc,"at most",0]// vc must be included to receive
emissions. it will revert otherwise ]], ])

// harvest -> swap -> compound awaitcompileAndExecute(0,[ [poolId(1,usdc_eth_pool),[//harvest [vc,"at most",0]// vc must
be included to receive emissions. it will revert otherwise ]], [poolId(0,vc_eth_pool),[// swap [vc,"all",INT128_MAX], [eth,"at
most",0], ]], [poolId(0,usdc_eth_pool),[// deposit lp [eth,"all",INT128_MAX], [usdc_eth_lp,"at most",0] ]],
[poolId(1,usdc_eth_pool),[// stake [usdc_eth_lp,"all",INT128_MAX], [vc,"at most",0]// vc must be included to receive
emissions. it will revert otherwise ]] ])
```

// unstake and remove liquidity to usdc letstakedAmount= (await(awaitgetContractAt("IGauge",usdc_eth_pool)).stakedTokens("0xmy_address"))[0] awaitcompileAndExecute(0,[[poolId(1,usdc_eth_pool),[// unstake [usdc_eth_lp,"exactly",stakedAmount.neg()], [vc,"at most",0]// vc must be included to receive emissions. it will revert otherwise]], [poolId(0,usdc_eth_pool),[// remove lp [usdc,"at most",0], //[eth, "at most", 0], // uncomment to receive both eth and usdc in proportion [usdc_eth_lp,"all",INT128_MAX]]],])

// lock vc awaitcompileAndExecute(0,[[poolId(0,vevc_address),[[vc,"exactly",parseEther("123")], [vevc,"at most",0],]],])

// vote awaitcompileAndExecute(0,[[poolId(4,usdc_eth_pool),[[vevc,"exactly",parseEther("123")],]],])

// unvote awaitcompileAndExecute(0,[[poolId(4,usdc_eth_pool),[[vevc,"exactly",parseEther("-123")],]],])

}

```

Github repo :https://github.com/velocore/velocore-contracts

Last updated5 months ago On this page
```