5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} On this page

Create Pool

Context

Creating a pool on Uniswap v4 is permissionless and enables the trading of an asset. Uniswap v4 is a popular destination for creating markets due to its:

- Proven track record and battle-tested codebase
- Concentrated liquidity, unlocking capital efficiency
- Flexibile pool design through dynamic fees and hooks
- Gas-efficient architecture
- Integrations with alternative trading venues

For more information, developers should seeUniswap v4 Overview

The guide covers two approaches to creating a pool:

1. Create a pool only
2. Create a pool and add initial liquidity, with one transaction

Setup

Developing with Uniswap v4requiresfoundry

Install the dependencies:

forge install uniswap/v4-core forge install uniswap/v4-periphery Copy

Guide: Create a Pool Only

To initialize a Uniswap v4 Poolwithout initial liquidity , developers should callPoolManager.initialize()

Creating a pool without liquidity may be useful for "reserving" a pool for future use, when initial liquidity is not available, or when external market makers would provide the starting liquidity

1. Configure the Pool

import

{ PoolKey }

from

"v4-core/src/types/PoolKey.sol" ;

PoolKey memory pool =

PoolKey ({ currency0 : currency0 , currency1 : currency1 , fee : lpFee , tickSpacing : tickSpacing , hooks : hookContract }) ; Copy For native token pairs (Ether), useCurrencyLibrary.ADDRESS_ZERO ascurrency0 PoolKey uniquely identifies a pool

- Currencies
- should be sorted,uint160(currency0) < uint160(currency1)
- lpFee
- is the fee expressed in pips, i.e. 3000 = 0.30%
- tickSpacing
- is the granularity of the pool. Lower values are more precise but may be more expensive to trade on
- hookContract
- is the address of the hook contract

A note ontickSpacing :

Lower tick spacing provides improved price precision; however, smaller tick spaces will cause swaps to cross ticks more often, incurring higher gas costs

As a reference, Uniswap v3 pools are configured with:

Fee Fee Value Tick Spacing 0.01% 100 1 0.05% 500 10 0.30% 3000 60 1.00% 10_000 200

2. Callinitialize

Pools are initialized with a starting price

IPoolManager (manager) . initialize (pool , startingPrice) ; Copy * thestartingPrice * is expressed as sqrtPriceX96:floor(sqrt(token1 / token0) * 2^96) * * i.e.79228162514264337593543950336 * * is the starting price for a 1:1 pool

Guide: Create a Pool & Add Liquidity

Uniswap v4'sPositionManager supports atomic creation of a pool and initial liquidity usingmulticall . Developers can create a trading pool, with liquidity, in a single transaction:

1. Initialize the parameters provided tomulticall()

bytes [ ]

memory params =

new

bytes [ ] (2) ; Copy * The first call,params[0] * , will encodeinitializePool * parameters * The second call,params[1] * , will encode amint * operation formodifyLiquidities

2. Configure the pool

PoolKey memory pool =

PoolKey ({ currency0 : currency0 , currency1 : currency1 , fee : lpFee , tickSpacing : tickSpacing , hooks : hookContract }) ; Copy For native token pairs (Ether), useCurrencyLibrary.ADDRESS_ZERO ascurrency0 PoolKey uniquely identifies a pool

- Currencies
- should be sorted,uint160(currency0) < uint160(currency1)
- lpFee
- is the fee expressed in pips, i.e. 3000 = 0.30%
- tickSpacing
- is the granularity of the pool. Lower values are more precise but more expensive to trade
- hookContract
- is the address of the hook contract

3. Encode theinitializePool

parameters

Pools are initialized with a starting price

params [0]

= abi . encodeWithSelector (PositionManager . initializePool . selector , pool , startingPrice) ; Copy * thestartingPrice * is expressed assqrtPriceX96 * :floor(sqrt(token1 / token0) * 2^96) * * 79228162514264337593543950336 * * is the starting price for a 1:1 pool

4. Initialize themint-liquidity

parameters

PositionManager'smodifyLiquidities uses an encoded command system

bytes

memory actions = abi . encodePacked (uint8 (Actions . MINT_POSITION) ,

uint8 (Actions . SETTLE_PAIR)) ; Copy * The first commandMINT_POSITION * creates a new liquidity position * The second commandSETTLE_PAIR * indicates that tokens are to be paid by the caller, to create the position

5. Encode theMINT_POSITION

parameters

bytes [ ]

memory mintParams =

new

bytes [ ] (2) ; mintParams [0]

= abi . encode (pool , tickLower , tickUpper , liquidity , amount0Max , amount1Max , recipient , hookData) ; Copy * pool * the samePoolKey * defined above, in pool-creation * tickLower * andtickUpper * are the range of the position, must be a multiple ofpool.tickSpacing * liquidity * is the amount of liquidity units to add, seeLiquidityAmounts * for converting token amounts to liquidity units * amount0Max * andamount1Max * are the maximum amounts of token0 and token1 the caller is willing to transfer * recipient * is the address that will receive the liquidity position (ERC-721) * hookData * is the optional hook data

6. Encode theSETTLE_PAIR

parameters

Creating a position on a pool requires the caller to transfercurrency0 and currency1 tokens Copy mintParams [1]

= abi . encode (pool . currency0 , pool . currency1) ; Copy

7. Encode the[modifyLiquidites]

call

uint256 deadline = block . timestamp +

60 ; params [1]

= abi . encodeWithSelector (posm . modifyLiquidities . selector , abi . encode (actions , mintParams) , deadline) ; Copy

8. Approve the tokens

PositionManager usesPermit2 for token transfers

- Repeat for both tokens

// approve permit2 as a spender IERC20 (token) . approve (address (permit2) ,

type (uint256) . max) ;

// approve PositionManager as a spender IAllowanceTransfer (address (permit2)) . approve (token ,

address (positionManager) ,

type (uint160) . max ,

type (uint48) . max) ; Copy

9. Execute the multicall

Themulticall is used to execute multiple calls in a single transaction

PositionManager (posm) . multicall (params) ; Copy For pools paired with native tokens (Ether), providevalue in the contract call

PositionManager (posm) . multicall { value : ethToSend } (params) ; Copy Excess Ether isNOT refunded unless developers encodedSWEEP in theactions parameter For a full end-to-end script, developers should seev4-template's scripts Edit this page .css-1tclyyl{margin-top:1.5rem;} .css-1c3fvx8{display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-flex-direction:row;-ms-flex-direction:row;flex-direction:row;-webkit-align-items:center;-webkit-box-align:center;-ms-flex-align:center;align-items:center;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;} .css-1wsnqg4{font-size:1rem;padding-right:0.5rem;} Helpful? .css-y2jwfw{fill:transparent;opacity:0.5;}.css-y2jwfw:hover{fill:#5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} Previous Integrated Routing with UniswapX Next Setup