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

Position

To add or remove the tokens we call the functionmodifyPosition which is defined inPoolManager.sol file

struct

ModifyPositionParams

{ // the lower and upper tick of the position int24 tickLower ; int24 tickUpper ; // how to modify the liquidity int256 liquidityDelta ; }

/// @notice Modify the position for the given pool function

modifyPosition (PoolKey memory key , ModifyPositionParams memory params ,

bytes

calldata hookData) external returns

(BalanceDelta) ; Copy

Important Concepts

Some of the important concepts to understand when working with Uniswap V3 or V4 positions are:

1. Tick
2. Tick Spacing
3. SquareRoot Price X96
4. Liquidity Delta

Tick

tick is a measure used in this code to handle prices of two different assets (tokens) in a unique way. Atick represents a specific price ratio between these two assets, calculated using a mathematical formula.

There are minimum and maximumtick values defined within the code, ensuring that calculated prices are within a reasonable and acceptable range.

In Uniswap V3 (and V4), liquidity providers can provide liquidity at specific price ranges (ticks), allowing them to concentrate their liquidity and potentially earn more fees.

Each tick corresponds to a specific price, and not all prices are represented due to the discrete nature of the ticks.

Tick Spacing

The tick spacing is a parameter that determines the separation between these usable ticks, making only every Nth tick available for liquidity provision, whereN is the tick spacing. This is a form of quantization of the price levels that liquidity can be provided at.

SquareRoot Price X96

In Uniswap V3 (and V4), the square root price (sqrtPriceX96) is a key concept and a crucial part of the mathematical calculations. It's utilized for various calculations, including determining the amount of tokens that should be moved during a swap and the liquidity calculations within specific price ranges.

Here's a breakdown of whatsqrtPriceX96 represents:

1.Square Root Price:

The price is represented as the square root of the actual price ratio of the tokens. This representation simplifies the math, especially when it comes to calculating the amounts to be swapped, as well as the liquidity calculations within tick ranges.

2.X96:

The X96 suffix refers to the fixed-point format used. Uniswap V3 utilizes a 96-bit fixed-point number format. In this representation, there is a convention to maintain high precision in calculations. The fixed-point representation means that the actual floating-point number is multiplied by(2^{96})and stored as an integer. When reading the value, it has to be interpreted properly by dividing it by(2^{96})to get the actual value.

SqrtPriceX96 to Tick

Since both thetick andsqrtPriceX96 are representations of the price, they can be converted from one to the other.

The Uniswap V3/V4 core library provides two functions to convert between the two representations:

https://github.com/Uniswap/v4-core/blob/main/src/libraries/TickMath.sol

1. The functiongetSqrtRatioAtTick
2. takes atick
3. value as an input, and it computes the square root of the price
4. ratio of the two assets at that specifictick
5. . The result represents the price relationship between two
6. assets in a particular state or position.
7. ThegetTickAtSqrtRatio
8. function does the reverse—it takes a square root of a price ratio and calculates
9. the correspondingtick
10. . Thistick
11. value represents a position or state where the assets have the given
12. price relationship.

Liquidity Delta

TheliquidityDelta is the difference between the current liquidity and the desired liquidity in a position. It can be positive (when you're adding liquidity) or negative (when you're removing liquidity).

Here is the code from Uniswap V3 contracts which calculates the liquidity (or liquidityDelta):

https://github.com/Uniswap/v3-periphery/blob/main/contracts/base/LiquidityManagement.sol

(uint160 sqrtPriceX96 ,

,

,

,

,

,

)

= pool . slot0 ( ) ; uint160 sqrtRatioAX96 = TickMath . getSqrtRatioAtTick (params . tickLower) ; uint160 sqrtRatioBX96 = TickMath . getSqrtRatioAtTick (params . tickUpper) ;

liquidity

LiquidityAmounts . getLiquidityForAmounts (sqrtPriceX96 , sqrtRatioAX96 , sqrtRatioBX96 , params . amount0Desired , params . amount1Desired) ; Copy

Example - Add Liquidity

Here is the code that adds liquidity to a position int24 tickLower =

- 600 ; int24 tickUpper =

600 ; uint256 liquidity =

1e18 ;

PoolManager manager

new

PoolManager (500000) ; // Helpers for interacting with the pool PoolModifyPositionTest modifyPositionRouter =

new

PoolModifyPositionTest (IPoolManager (address (manager))) ;

modifyPositionRouter . modifyPosition (poolKey , IPoolManager . ModifyPositionParams ({ tickLower : tickLower , tickUpper : tickUpper , liquidityDelta :

int256 (liquidity) }) , ZERO_BYTES) ; Copy Note:PoolModifyPositionTest implements theILockCallback interface and adds thelockAcquired function, which in turn calls themanager.modifyPosition function.

Acquiring Lock

Full detail about the locking mechanism is explained in theLocking Mechanism section.

The contract that calls themodifyPosition must implement ILockCallback interface.

PoolModifyPositionTest.sol has some examples of how to acquire lock and some basic checks in place.https://github.com/Uniswap/v4-core/blob/main/src/test/PoolModifyPositionTest.sol

InPoolModifyPositionTest thelockAcquired function is executed when the lock is acquired, handling balance adjustments and interactions with external currencies and contracts. The function takes raw encoded data as input, which is then decoded into structured data, specificallyCallbackData . Essential validations and checks are performed, ensuring the caller of the function is the manager, and it processes the modifications such as settling amounts and making necessary transfers based on conditions like whether the amount being positive or negative. 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 Pool Manager and Pool Initialization Next Swap Tokens