Router

This documentation provides an overview of the IRouter.sol. This contract defines various functions and structs used for interacting with the Maverick AMM.

Contents

Contract Details

- Name :
- IRouter
- Solidity Version :
- ^0.8.0
- SPDX License-Identifier : GPL-2.0-or-later
- Router
- is Deployed to
-
    - Ethereum:0xbBF1EE38152E9D8e3470Dc47947eAa65DcA94913
-
    - ZKSync Era:0x39E098A153Ad69834a9Dac32f0FCa92066aD03f4
- *
- Code:Github
-

Structs

PoolParams

```

Copy structPoolParams{ uint256fee; uint256tickSpacing; int256lookback; int32activeTick; IERC20 tokenA; IERC20 tokenB; }

```

- fee
- : The fee valueuint256
- associated with the pool.
- tickSpacing
- : The tick spacing valueuint256
- for the pool.

- lookback
- : The lookback valueint256
- for the pool.
- activeTick
- : The active tick valueint32
- for the pool.
- tokenA
- : The ERC20 token Aaddress
- associated with the pool.
- tokenB
- : The ERC20 token Baddress
- associated with the pool.
-

ExactInputParams

```

Copy structExactInputParams{ bytespath; addressrecipient; uint256deadline; uint256amountIn; uint256amountOutMinimum; }

```

- path
- : Abytes
- string representing the path of tokens to swap.
- recipient
- : Theaddress
- where the swapped tokens will be sent.
- deadline
- : The deadline timestamp (in seconds)uint256
- until which the swap can be executed.
- amountIn
- : The amount of the input tokenuint256
- to be swapped.
- amountOutMinimum
- : The minimum acceptable amount of the output tokenuint256
- specified to prevent infinite slippage.
-

ExactOutputParams

```

Copy structExactOutputParams{ bytespath; addressrecipient; uint256deadline; uint256amountOut; uint256amountInMaximum; }

```

- path
- : Abytes
- string representing the path of tokens to swap(reversed)
- .
- recipient
- : Theaddress
- where the swapped tokens will be sent.
- deadline
- : The deadline timestamp (in seconds)uint256
- until which the swap can be executed.
- amountOut
- : The amount of the output tokenuint256
- desired.
- amountInMaximum
- : The maximum acceptable amount of the input tokenuint256
- required.
-

Functions

factory()

Returns the address of the factory.

```
```

Copy functionfactory()externalviewreturns(IFactory);

```

- Returns :
-
    - IFactory : Theaddress
-
    - of the Factory
- *
-

position()

Returns the address of the Position NFT.

```

Copy functionposition()externalviewreturns(IPosition);

```

- Returns
-
    - IPosition : The position NFTaddress
- *
-

exactInput()

SwapsamountIn of one token for as much as possible of another along the specified path.

```

Copy functionexactInput(ExactInputParamscalldataparams)externalpayablereturns(uint256amountOut);

```

- Parameters :
-
    - params
-
    - : The parameters necessary for the multi-hop swap, encoded asExactInputParams
-
    - in calldata
- *
- Returns :
-
    - amountOut
-
    - : The amount of the received token inuint256
- *
-

exactOutput()

Swaps as little as possible of one token foramountOut of another along the specified path(reversed) .

```

Copy functionexactOutput(ExactOutputParamscalldataparams)externalpayablereturns(uint256amountIn);

```

- Parameters :
-
    - params
-
    - : The parameters necessary for the multi-hop swap, encoded asExactOutputParams
-
    - in calldata
- *
- Returns :
-
```

- amountIn
- : The amount of the input token inuint256
- *
-

getOrCreatePoolAndAddLiquidity()

Either gets an existing pool or creates a pool if it does not exist and adds liquidity to it.

```

Copy functiongetOrCreatePoolAndAddLiquidity( PoolParamscalldatapoolParams, uint256tokenId, IPool.AddLiquidityParams[]calldataaddParams, uint256minTokenAAmount, uint256minTokenBAmount, uint256deadline )externalpayablereturns(uint256receivingTokenId,uint256tokenAAmount,uint256tokenBAmount,IPool.BinDelta[]memorybinDeltas);

```

- Parameters :
- poolParams
- : Parameters of a pool withpoolParams
- tokenId
- : NFT ID of the tokenuint256
- that will hold LP balance. Use0
- to mint a new token
- addParams
- : Parameters of liquidity addition withaddParams
- minTokenAAmount
- : Minimum amount of token Auint256
- to add. Reverts if not met
- minTokenBAmount
- : Minimum amount of token Buint256
- to add. Reverts if not met
- deadline
- : Epoch timestamp (in seconds)uint256
- *
- Returns :
- receivingTokenId
- : The IDuint256
- of the receiving token
- tokenAAmount
- : The amount of token Auint256
- tokenBAmount
- : The amount of token Buint256
- binDeltas
- : An array ofBinDelta
-

    - structures
- *
-

addLiquidityToPool()

Adds liquidity to a pool.

```

Copy functionaddLiquidityToPool( IPoolpool, uint256tokenId, IPool.AddLiquidityParams[]calldataparams,
uint256minTokenAAmount, uint256minTokenBAmount, uint256deadline
)externalpayablereturns(uint256receivingTokenId,uint256tokenAAmount,uint256tokenBAmount,IPool.BinDelta[]memorybinDeltas);

```

- Parameters :
-
    - pool
-
    - : Pool to add liquidity toIPool
-
    - tokenId
-
    - : NFT ID of the tokenuint256
-
    - that will hold LP balance. Use0
-
    - to mint a new token
-
    - params
-
    - : Parameters of liquidity addition withparams
-
    - minTokenAAmount
-
    - : Minimum amount of token Auint256
-
    - to add. Reverts if not met
-
    - minTokenBAmount
-
    - : Minimum amount of token Buint256
-
    - to add. Reverts if not met
-
    - deadline
-
    - : Epoch timestamp (in seconds)uint256
- *
- Returns :
-
    - receivingTokenId
-
    - : The ID of the receiving tokenuint256
-
    - tokenAAmount
-
    - : The amount of token Auint256
-
    - tokenBAmount
-
    - : The amount of token Buint256
-
    - binDeltas
-
    - : An array ofBinDelta
-
    - structures
- *
-

addLiquidityWTickLimits()

Adds liquidity to a pool with active tick limits.

```

Copy functionaddLiquidityWTickLimits( IPoolpool, uint256tokenId, IPool.AddLiquidityParams[]calldataparams, uint256minTokenAAmount, uint256minTokenBAmount, int32minActiveTick, int32maxActiveTick, uint256deadline )externalpayablereturns(uint256receivingTokenId,uint256tokenAAmount,uint256tokenBAmount,IPool.BinDelta[]memorybinDeltas);

```

- Parameters :
-
  - pool
-
  - : Pool to add liquidity toIPool
-
  - tokenId
-
  - : NFT ID of the tokenuint256
-
  - that will hold LP balance. Use0
-
  - to mint a new token
-
  - params
-
  - : Parameters of liquidity addition withparams
-
  - minTokenAAmount
-
  - : Minimum amount of token Auint256
-
  - to add. Reverts if not met
-
  - minTokenBAmount
-
  - : Minimum amount of token Buint256
-
  - to add. Reverts if not met
-
  - minActiveTick
-
  - : Lowest activeTickint32
-
  - (inclusive)
-
  - of the pool that will permit the transaction to pass
-
  - maxActiveTick
-
  - : Highest activeTickint32
-
  - (inclusive)
-
  - of the pool that will permit the transaction to pass
-
  - deadline
-
  - : Epoch timestamp(in seconds)
- *
- Returns :
-
  - receivingTokenId
-
  - : The ID of the receiving tokenuint256
-
  - tokenAAmount
-
  - : The amount of token Auint256
-

- tokenBAmount
- : The amount of token Buint256
- binDeltas
- : An array ofBinDelta
- structures
- *
-

migrateBinsUpStack()

Moves the head of input merged bins to the active bin.

```

Copy functionmigrateBinsUpStack(IPoolpool,uint128[]calldatabinIds,uint32maxRecursion,uint256deadline)external;

```

- Parameters :
- pool
- : Pool to remove from withIPool
- binIds
- : Array ofbinIds
- to migrate
- maxRecursion
- : Maximum recursion depthuint32
- before returning;0
- means no maximum
- deadline
- : Epoch timestamp(in seconds)
- uint256
- *
-

removeLiquidity()

Removes liquidity from a pool and receives WETH if one of the tokens is WETH.

```

Copy functionremoveLiquidity( IPoolpool, addressrecipient, uint256tokenId, IPool.RemoveLiquidityParams[]calldataparams, uint256minTokenAAmount, uint256minTokenBAmount, uint256deadline
)externalreturns(uint256tokenAAmount,uint256tokenBAmount,IPool.BinDelta[]memorybinDeltas);

```

Router must be approved for the withdrawing tokenId:Position.approve(router, tokenId)

- Parameters :
- pool
- : Pool to remove from withIPool
- recipient
- :address

-
  - where the proceeds are sent. Usezero
-
  - or router address to leave tokens in the router
-
  - tokenId
-
  - : IDuint256
-
  - of the position NFT that holds liquidity
-
  - params
-
  - : Parameters of liquidity removal withparams
-
  - minTokenAAmount
-
  - : Minimum amount of token Auint256
-
  - to receive. Reverts if not met
-
  - minTokenBAmount
-
  - : Minimum amount of token Buint256
-
  - to receive. Reverts if not met
-
  - deadline
-
  - : Epoch timestamp(in seconds)
-
  - uint256
- *
- Returns :
-
  - tokenAAmount
-
  - : The amount of token Auint256
-
  - received
-
  - tokenBAmount
-
  - : The amount of token Buint256
-
  - received
-
  - binDeltas
-
  - : An array ofBinDelta
-
  - structure
- *
-

Previous Development FAQ Next Pool Last updated9 months ago On this page *