Stableswap

This section contains a full API reference of all public functions & events related to Connext's stableswap contracts.

Events

TokenSwap

```
Copy eventTokenSwap(bytes32key,addressbuyer,uint256tokensSold,uint256tokensBought,uint128soldId,uint128boughtId)
```

AddLiquidity

```
Copy event AddLiquidity(bytes32 key, address provider, uint256[] tokenAmounts, uint256[] fees, uint256 invariant, uint256 lpTokenSupply)
```

RemoveLiquidity

```
Copy eventRemoveLiquidity(bytes32key,addressprovider,uint256[] tokenAmounts,uint256lpTokenSupply)
```

RemoveLiquidityOne

```
Copy event RemoveLiquidityOne(bytes32 key, address provider, uint256 lpTokenAmount, uint256 lpTokenSupply, uint256 boughtId, uint256 tokensBought)
```

Getters

getSwapStorage

```
Copy functiongetSwapStorage(bytes32key)externalviewreturns(structSwapUtils.Swap)
```

Return Stable swap storage

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain

Return Values

Name Type Description [0] struct SwapUtils.Swap SwapUtils.Swap

getSwapLPToken

```
Copy functiongetSwapLPToken(bytes32key)externalviewreturns(address)
```

Return LP token for canonical Id

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain

Return Values

Name Type Description [0] address LPToken

getSwapA

```
Copy functiongetSwapA(bytes32key)externalviewreturns(uint256)
```

Return A, the amplification coefficient _ n _ (n - 1)

See the StableSwap paper for details

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain

Return Values

Name Type Description [0] uint256 A parameter

getSwapAPrecise

```
Copy functiongetSwapAPrecise(bytes32key)externalviewreturns(uint256)
```

Return A in its raw precision form

See the StableSwap paper for details

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain

Return Values

Name Type Description [0] uint256 A parameter in its raw precision form

getSwapToken

```
Copy functiongetSwapToken(bytes32key,uint8index)publicviewreturns(contractIERC20)
```

Return address of the pooled token at given index. Reverts if tokenIndex is out of range.

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain index uint8 the index of the token

Return Values

Name Type Description [0] contract IERC20 address of the token at given index

getSwapTokenIndex

```
Copy functiongetSwapTokenIndex(bytes32key,addresstokenAddress)publicviewreturns(uint8)
```

Return the index of the given token address. Reverts if no matching token is found.

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain tokenAddress address address of the token

Return Values

Name Type Description [0] uint8 the index of the given token address

getSwapTokenBalance

```

Copy functiongetSwapTokenBalance(bytes32key,uint8index)externalviewreturns(uint256)

```

Return current balance of the pooled token at given index

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain index uint8 the index of the token

Return Values

Name Type Description [0] uint256 current balance of the pooled token at given index with token's native precision

getSwapVirtualPrice

```

Copy functiongetSwapVirtualPrice(bytes32key)externalviewreturns(uint256)

```

Get the virtual price, to help calculate profit

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain

Return Values

Name Type Description [0] uint256 the virtual price, scaled to the POOL_PRECISION_DECIMALS

calculateSwap

```

Copy function calculateSwap(bytes32 key, uint8 tokenIndexFrom, uint8 tokenIndexTo, uint256 dx) external view returns (uint256)

```

Calculate amount of tokens you receive on swap

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain tokenIndexFrom uint8 the token the user wants to sell tokenIndexTo uint8 the token the user wants to buy dx uint256 the amount of tokens the user wants to sell. If the token charges a fee on transfers, use the amount that gets transferred after the fee.

Return Values

Name Type Description [0] uint256 amount of tokens the user will receive

calculateSwapTokenAmount

```
```

Copy functioncalculateSwapTokenAmount(bytes32key,uint256[] amounts,booldeposit)externalviewreturns(uint256)

```

A simple method to calculate prices from deposits or withdrawals, excluding fees but including slippage. This is helpful as an input into the various "min" parameters on calls to fight front-running

This shouldn't be used outside frontends for user estimates.

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain amounts uint256[] an array of token amounts to deposit or withdrawal, corresponding to pooledTokens. The amount should be in each pooled token's native precision. If a token charges a fee on transfers, use the amount that gets transferred after the fee. deposit bool whether this is a deposit or a withdrawal

Return Values

Name Type Description [0] uint256 token amount the user will receive

calculateRemoveSwapLiquidity

```

Copy functioncalculateRemoveSwapLiquidity(bytes32key,uint256amount)externalviewreturns(uint256[])

```

A simple method to calculate amount of each underlying tokens that is returned upon burning given amount of LP tokens

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain amount uint256 the amount of LP tokens that would be burned on withdrawal

Return Values

Name Type Description [0] uint256[] array of token balances that the user will receive

calculateRemoveSwapLiquidityOneToken

```

Copy function calculateRemoveSwapLiquidityOneToken(bytes32 key, uint256 tokenAmount, uint8 tokenIndex) external view returns (uint256 availableTokenAmount)

```

Calculate the amount of underlying token available to withdraw when withdrawing via only single token

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain tokenAmount uint256 the amount of LP token to burn tokenIndex uint8 index of which token will be withdrawn

Return Values

Name Type Description availableTokenAmount uint256 calculated amount of underlying token available to withdraw

Functions

swap

```

Copy function swap(bytes32 key, uint8 tokenIndexFrom, uint8 tokenIndexTo, uint256 dx, uint256 minDy, uint256 deadline) external returns (uint256)

```

Swap two tokens using this pool
```

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain tokenIndexFrom uint8 the token the user wants to swap from tokenIndexTo uint8 the token the user wants to swap to dx uint256 the amount of tokens the user wants to swap from minDy uint256 the min amount the user would like to receive, or revert. deadline uint256 latest timestamp to accept this transaction

swapExact

```

Copy function swapExact(bytes32 key, uint256 amountIn, address assetIn, address assetOut, uint256 minAmountOut, uint256 deadline) external returns (uint256)

```

Swap two tokens using this pool

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain amountIn uint256 the amount of tokens the user wants to swap from assetIn address the token the user wants to swap from assetOut address the token the user wants to swap to minAmountOut uint256 deadline uint256

swapExactOut

```

Copy function swapExactOut(bytes32 key, uint256 amountOut, address assetIn, address assetOut, uint256 maxAmountIn, uint256 deadline) external returns (uint256)

```

Swap two tokens using this pool

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain amountOut uint256 the amount of tokens the user wants to swap to assetIn address the token the user wants to swap from assetOut address the token the user wants to swap to maxAmountIn uint256 deadline uint256

addSwapLiquidity

```

Copy function addSwapLiquidity(bytes32 key, uint256[] amounts, uint256 minToMint, uint256 deadline) external returns (uint256)

```

Add liquidity to the pool with the given amounts of tokens

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain amounts uint256[] the amounts of each token to add, in their native precision minToMint uint256 the minimum LP tokens adding this amount of liquidity should mint, otherwise revert. Handy for front-running mitigation deadline uint256 latest timestamp to accept this transaction

Return Values

Name Type Description [0] uint256 amount of LP token user minted and received

removeSwapLiquidity

```

Copy function removeSwapLiquidity(bytes32 key, uint256 amount, uint256[] minAmounts, uint256 deadline) external returns (uint256[])

```

Burn LP tokens to remove liquidity from the pool. Withdraw fee that decays linearly over period of 4 weeks since last deposit will apply.

Liquidity can always be removed, even when the pool is paused.

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain amount uint256 the amount of LP tokens to burn minAmounts uint256[] the minimum amounts of each token in the pool acceptable for this burn. Useful as a front-running mitigation deadline uint256 latest timestamp to accept this transaction

Return Values

Name Type Description [0] uint256[] amounts of tokens user received

removeSwapLiquidityOneToken

```

Copy function removeSwapLiquidityOneToken(bytes32 key, uint256 tokenAmount, uint8 tokenIndex, uint256 minAmount, uint256 deadline) external returns (uint256)

```

Remove liquidity from the pool all in one token. Withdraw fee that decays linearly over period of 4 weeks since last deposit will apply.

Parameters

Name Type Description key bytes32 Hash of the canonical id + domain tokenAmount uint256 the amount of the token you want to receive tokenIndex uint8 the index of the token you want to receive minAmount uint256 the minimum amount to withdraw, otherwise revert deadline uint256 latest timestamp to accept this transaction

Return Values

Name Type Description [0] uint256 amount of chosen token user received

Previous Routers Next Subgraphs Last updated9 months ago On this page *Events * TokenSwap * AddLiquidity * RemoveLiquidity * RemoveLiquidityOne * Getters * getSwapStorage * getSwapLPToken * getSwapA * getSwapAPrecise * getSwapToken * getSwapTokenIndex * getSwapTokenBalance * getSwapVirtualPrice * calculateSwap * calculateSwapTokenAmount * calculateRemoveSwapLiquidity * calculateRemoveSwapLiquidityOneToken * Functions * swap * swapExact * swapExactOut * addSwapLiquidity * removeSwapLiquidity * removeSwapLiquidityOneToken

Edit on GitHub