Executing Private Swap on L1

To execute the swaps on L1, go back to theUniswapPortal.sol we[created earlier] inl1-contracts .

solidity_uniswap_swap_private /* **@notice Exit with funds from L2, perform swap on L1 and deposit output asset to L2 again privately * @dev* msg.value *indicates fee to submit message to inbox. Currently, anyone can call this method on your behalf. * They could call it with 0 fee causing the sequencer to never include in the rollup. * In this case, you will have to cancel the message and then make the deposit later * @param _inputTokenPortal - The ethereum address of the input token portal * @param _inAmount - The amount of assets to swap (same amount as withdrawn from L2) * @param _uniswapFeeTier - The fee tier for the swap on UniswapV3 * @param _outputTokenPortal - The ethereum address of the output token portal * @param _amountOutMinimum - The minimum amount of output assets to receive from the swap (slippage protection) * @param _secretHashForRedeemingMintedNotes - The hash of the secret to redeem minted notes privately on Aztec. The hash should be 254 bits (so it can fit in a Field element) * @param _secretHashForL1ToL2Message - The hash of the secret consumable message. The hash should be 254 bits (so it can fit in a Field element) * @param _deadlineForL1ToL2Message - deadline for when the L1 to L2 message (to mint output assets in L2) must be consumed by * @param _canceller - The ethereum address that can cancel the deposit * @param _withCaller - When true, using* msg.sender *as the caller, otherwise address(0) * @return The entryKey of the deposit transaction in the Inbox*/ function

swapPrivate (address _inputTokenPortal , uint256 _inAmount , uint24 _uniswapFeeTier , address _outputTokenPortal , uint256 _amountOutMinimum , bytes32 _secretHashForRedeemingMintedNotes , bytes32 _secretHashForL1ToL2Message , uint32 _deadlineForL1ToL2Message , address _canceller , bool _withCaller)

public

payable

returns

(bytes32)

{ LocalSwapVars memory vars ;

vars . inputAsset =

TokenPortal (_inputTokenPortal) . underlying ( ) ; vars . outputAsset =

TokenPortal (_outputTokenPortal) . underlying ( ) ;

// Withdraw the input asset from the portal TokenPortal (_inputTokenPortal) . withdraw (address (this) , _inAmount ,

true) ; { // prevent stack too deep errors vars . contentHash = Hash . sha256ToField (abi . encodeWithSignature ("swap_private(address,uint256,uint24,address,uint256,bytes32,bytes32,uint32,address,address)" , _inputTokenPortal , _inAmount , _uniswapFeeTier , _outputTokenPortal , _amountOutMinimum , _secretHashForRedeemingMintedNotes , _secretHashForL1ToL2Message , _deadlineForL1ToL2Message , _canceller , _withCaller ? msg . sender :

address (0))) ; }

// Consume the message from the outbox registry . getOutbox ( ) . consume (DataStructures . L2ToL1Msg ({ sender : DataStructures . L2Actor (l2UniswapAddress ,

1) , recipient : DataStructures . L1Actor (address (this) , block . chainid) , content : vars . contentHash })) ;

// Perform the swap ISwapRouter . ExactInputSingleParams memory swapParams ; { swapParams = ISwapRouter . ExactInputSingleParams ({ tokenIn :

address (vars . inputAsset) , tokenOut :

address (vars . outputAsset) , fee : _uniswapFeeTier , recipient :

address (this) , deadline : block . timestamp , amountIn : _inAmount , amountOutMinimum : _amountOutMinimum , sqrtPriceLimitX96 :

0 }) ; } // Note, safeApprove was deprecated from Oz vars . inputAsset . approve (address (ROUTER) , _inAmount) ; uint256 amountOut = ROUTER . exactInputSingle (swapParams) ;

// approve the output token portal to take funds from this contract // Note, safeApprove was deprecated from Oz vars . outputAsset . approve (address (_outputTokenPortal) , amountOut) ;

// Deposit the output asset to the L2 via its portal return

TokenPortal (_outputTokenPortal) . depositToAztecPrivate { value : msg . value } (_secretHashForRedeemingMintedNotes , amountOut , _canceller , _deadlineForL1ToL2Message , _secretHashForL1ToL2Message) ; } } Source code: l1-contracts/test/portals/UniswapPortal.sol#L134-L231 This works very similarly to the public flow. Edit this page

Previous Swapping Privately Next Redeeming Swapped Assets on L2