Uniswap Portal on L1

In this step we will set up our Solidity portal contract.

InI1-tokens create a new file calledUniswapPortal.sol

cd I1-contracts/contracts && touch UniswapPortal.sol and paste this inside:

pragma

solidity

^ 0.8.20 ;

import

{ IERC20 }

from

"@openzeppelin/contracts/token/ERC20/IERC20.sol" ;

import

{ IRegistry }

from

"@aztec/I1-contracts/src/core/interfaces/messagebridge/IRegistry.sol" ; import

{ DataStructures }

from

"@aztec/I1-contracts/src/core/libraries/DataStructures.sol" ; import

{ Hash }

from

"@aztec/I1-contracts/src/core/libraries/Hash.sol" ;

import

{ TokenPortal }

from

"./TokenPortal.sol" ; import

{ ISwapRouter }

from

"../external/ISwapRouter.sol" ;

/* * @title UniswapPortal * @author Aztec Labs * @notice A minimal portal that allow an user inside L2, to withdraw asset A from the Rollup * swap asset A to asset B, and deposit asset B into the rollup again. * Relies on Uniswap for doing the swap, TokenPortals for A and B to get and send tokens * and the message boxes (inbox & outbox). / contract

UniswapPortal

{ ISwapRouter public

constant ROUTER =

ISwapRouter (0xE592427A0AEce92De3Edee1F18E0157C05861564) ;

IRegistry public registry ; bytes32

public I2UniswapAddress ;

function

initialize (address _registry ,

bytes32 _l2UniswapAddress)

external

{ registry =

IRegistry (_registry) ; l2UniswapAddress = _l2UniswapAddress ; }

// Using a struct here to avoid stack too deep errors struct

LocalSwapVars

{ IERC20 inputAsset ; IERC20 outputAsset ; bytes32 contentHash ; } In this set up we defined theinitialize() function and a struct (LocalSwapVars) to manage assets being swapped.

Like we saw in theTokenPortal , we initialize this portal with the registry contract address (to fetch the appropriate inbox and outbox) and the portal's sister contract address on L2.

In the next step we will set up the appropriate L2 Uniswap contractEdit this page

Previous Setup & Installation Next L2 Contract Setup