Simple Bridge

TheSimpleBridge just transfers tokens from a user to another wallet (could be themselves) on a different chain. Since no calldata is involved, no target contract is needed.

In this example,SimpleBridge has two functions:

- xTransfer
- bridges any ERC20 token
-
  - The user must first approve a spending allowance of the token to theSimpleBridge
-
  - contract.
-
  - relayerFee
-
  - is paid in native ETH so whenxTransfer
-
  - is called,msg.value
-
  - MUST be passed in equal to the specifiedrelayerFee
-
  - . Informaation for calculating relayer fees can be found on Estimating Fees page.
- *
- xTransferEth
- bridges ETH (for origin/destination chains whose native asset is ETH)
-
  - To send and receive native ETH, the flow is a bit different. Since Connext doesn't accept native ETH as the bridged asset, ETH should be first wrapped into WETH on the origin domain and then the delivered WETH on destination should be unwrapped back to ETH.
-
  - An Unwrapper contract that implementsIXReceive
-
  - already exists on all supported networks to be used as the_to
-
  - target inxcall
-
  - . The final recipient on destination should be encoded into thecallData
-
  - param for the Unwrapper to send ETH to (demonstrated on line 92 below).
-
  - When sending ETH,msg.value
-
  - =relayerFee
-
  - +amount
-
  - . See example below (note: in Etherscan, the payable field is inether
-
  - while the other fields are specified inwei
-
  -).
- *
-

```

Copy // SPDX-License-Identifier: UNLICENSED pragmasolidity^0.8.15;

import{IConnext}from"@connext/interfaces/core/IConnext.sol";
import{IERC20}from"@openzeppelin/contracts/token/ERC20/IERC20.sol";

interfaceIWETH{ functiondeposit()externalpayable; functionapprove(addressguy,uintwad)externalreturns(bool); }

/ *@titleSimpleBridge* @noticeExample of a cross-domain token transfer. */ contractSimpleBridge{ // The connext contract on the origin domain IConnextpublicimmutableconnext;

constructor(address_connext) { connext=IConnext(_connext); }
```

/ *@noticeTransfers non-native assets from one chain to another.*@devUser should approve a spending allowance before calling this. *@paramtoken Address of the token on this domain.*@paramamount The amount to transfer. *@paramrecipient The destination address (e.g. a wallet).* @paramdestinationDomain The destination domain ID.*@paramslippage The maximum amount of slippage the user will accept in BPS.* @paramrelayerFee The fee offered to relayers. */ functionxTransfer(addresstoken, uint256amount, addressrecipient, uint32destinationDomain, uint256slippage, uint256relayerFee)externalpayable{ IERC20 _token=IERC20(token);

require(_token.allowance(msg.sender,address(this))>=amount, "User must approve amount");

// User sends funds to this contract _token.transferFrom(msg.sender,address(this),amount);

// This contract approves transfer to Connext _token.approve(address(connext),amount);

connext.xcall{value:relayerFee}(destinationDomain,// _destination: Domain ID of the destination chain recipient,// _to: address receiving the funds on the destination token,// _asset: address of the token contract msg.sender,// _delegate: address that can revert or forceLocal on destination amount,// _amount: amount of tokens to transfer slippage,// _slippage: the maximum amount of slippage the user will accept in BPS (e.g. 30 = 0.3%) bytes("")// _callData: empty bytes because we're only sending funds); }

/ *@noticeTransfers native assets from one chain to another.*@paramdestinationUnwrapper Address of the Unwrapper contract on destination. *@paramweth Address of the WETH contract on this domain.*@paramamount The amount to transfer. *@paramrecipient The destination address (e.g. a wallet).*@paramdestinationDomain The destination domain ID. *@paramslippage The maximum amount of slippage the user will accept in BPS*@paramrelayerFee The fee offered to relayers. */ functionxTransferEth(addressdestinationUnwrapper, addressweth, uint256amount, addressrecipient, uint32destinationDomain, uint256slippage, uint256relayerFee)externalpayable{ // Wrap ETH into WETH to send with the xcall IWETH(weth).deposit{value:amount}();

// This contract approves transfer to Connext IWETH(weth).approve(address(connext),amount);

// Encode the recipient address for calldata bytesmemorycallData=abi.encode(recipient);

// xcall the Unwrapper contract to unwrap WETH into ETH on destination connext.xcall{value:relayerFee}(destinationDomain,// _destination: Domain ID of the destination chain destinationUnwrapper,// _to: Unwrapper contract weth,// _asset: address of the WETH contract msg.sender,// _delegate: address that can revert or forceLocal on destination amount,// _amount: amount of tokens to transfer slippage,// _slippage: the maximum amount of slippage the user will accept in BPS (e.g. 30 = 0.3%) callData// _callData: calldata with encoded recipient address); } }

```

Information like asset addresses be found in the Deployments page.
```