StargateComposer.sol

Wrapper Contract that wraps IStargateRouter to add additional functionality to Stargate Composed calls.

swap()

This code snippet shows how theStargateComposer.sol uses theIStargateRouter to swap tokens using Stargate to another chain.

```

Copy / *@param_dstChainId - destination chain identifier*@param_srcPoolId - source pool identifier*@param_dstPoolId - destination pool identifier* @param_refundAddress - refund address*@param_amountLD - amount (local decimals) to swap on source* @param_minAmountLD - min amount (local decimals) to receive on destination*@param_lzTxParams - struct: dstGasForCall,dstNativeAmount,dstNativeAddr* @param_to - destination address (the sgReceive() implementer) *@param_payload - bytes payload*/ functionswap( uint16_dstChainId, uint256_srcPoolId, uint256_dstPoolId, addresspayable_refundAddress, uint256_amountLD, uint256_minAmountLD, IStargateRouter.lzTxObjmemory_lzTxParams, bytescalldata_to, bytescalldata_payload )externaloverridepayablenonSwapReentrant{ bytesmemorynewPayload; bytesmemorypeer; if(_payload.length>0) { newPayload=_buildPayload(_to,_payload); peer=_getPeer(_dstChainId);

// overhead for calling composer's sgReceive() _lzTxParams.dstGasForCall+=dstGasReserve+transferOverhead; }else{ newPayload=""; peer=_to; }

if(isEthPool(_srcPoolId)) { require(msg.value>_amountLD,"Stargate: msg.value must be > _swapAmount.amountLD"); IStargateEthVault(stargateEthVaults[_srcPoolId]).deposit{value:_amountLD}(); IStargateEthVault(stargateEthVaults[_srcPoolId]).approve(address(stargateRouter),_amountLD); }else{ PoolInfomemorypoolInfo=_getPoolInfo(_srcPoolId); // remove dust if(poolInfo.convertRate>1) _amountLD=_amountLD.div(poolInfo.convertRate).mul(poolInfo.convertRate); // transfer token to this contract IERC20(poolInfo.token).safeTransferFrom(msg.sender,address(this),_amountLD); }

stargateRouter.swap{value:isEthPool(_srcPoolId)?msg.value-_amountLD:msg.value}( _dstChainId, _srcPoolId, _dstPoolId, _refundAddress, _amountLD, _minAmountLD, _lzTxParams, peer,// swap the to address with the peer address newPayload ); }

```

buildPayload()

In the swap function it calls buildPayload to include the msg.sender in the payload.

```

Copy function_buildPayload( bytescalldata_to, bytescalldata_payload )internalviewreturns(bytesmemory) { require(_to.length==20,"Stargate: invalid to address");

// new payload = to(20) + sender(20) + payload // encoding the sender allows the receiver to know who called the Stargate returnabi.encodePacked(_to,msg.sender,_payload); }

```

sgReceive()

StargateComposer.sol implementsIStargateReceiver so it can implement thesgReceive function and receive the tokens and payload from theIStargateRouter on the destination chain. It then forwards thesgReceive call to the intended receiver with the original msg.sender who initialed the swap on source.

```

Copy / *@param_srcChainId - source chain identifier*@param_srcAddress - source address identifier*@param_nonce - message ordering nonce* @param_token - token contract *@param_amountLD - amount (local decimals) to recieve* @param_payload - bytes containing the toAddress */ functionsgReceive( uint16_srcChainId, bytesmemory_srcAddress, uint256_nonce, address_token, uint256_amountLD, bytesmemory_payload )externaloverride{ require(msg.sender==address(stargateRouter),"Stargate: only router"); // will just ignore the payload in some invalid configuration if(_payload.length<=40)return;// 20 + 20 + payload

addressintendedReceiver=_payload.toAddress(0);

(boolsuccess,bytesmemorydata)=_token.call(abi.encodeWithSelector(SELECTOR,intendedReceiver,_amountLD)); if(success&&(data.length==0||abi.decode(data,(bool)))) { if(!intendedReceiver.isContract())return;// ignore
```

bytesmemorycallData=abi.encodeWithSelector(IStargateReceiver.sgReceive.selector, _srcChainId, abi.encodePacked(_payload.toAddress(20)),// use the caller as the srcAddress (the msg.sender caller the StargateComposer at the source) _nonce, _token, _amountLD, _payload.slice(40,_payload.length-40));

// no point in requires, because it will revert regardless uint256externalGas=gasleft()-dstGasReserve;

(boolsafeCallSuccess,bytesmemoryreason)=intendedReceiver.safeCall(externalGas,0,150,callData);// only return 150 bytes of data

if(!safeCallSuccess) { payloadHashes[_srcChainId][_srcAddress] [_nonce]=keccak256(abi.encodePacked(intendedReceiver,callData)); emitCachedSwapSaved(_srcChainId,_srcAddress,_nonce,reason); }

}else{ // do nothing, token swap failed and can't be delivered, tokens are held inside this contract emitComposedTokenTransferFailed(_token,intendedReceiver,_amountLD); } }

``` Previous Stargate Composability Next StargateComposed.sol Last updated6 months ago On this page 
```