SDK

The Connext SDK allows developers to interact with the Connext protocol in standard Node.js or web environments. Seehere for a reference of all SDK methods.

Cross-Chain Transfer

This example demonstrates how to execute anxcall to transfer funds from a wallet on the source domain to the same address on the destination domain.

1. Setup

InstallNode.js and useNode.js v18 . Follow the instructions to installnvm , a node version manager, which will make switching versions easier.

Create a project folder and initialize the package. Fill out the project information as you please.

```

Copy mkdirconnext-sdk-example&&cdconnext-sdk-example npminit

```

We'll be using TypeScript so install the following and generate thetsconfig.json file.

```

Copy npminstall--save-dev@types/node@types/chai@types/mochatypescript npxtsc--init# oryarn tsc --init

```

We want to use top-level await so we'll set the compiler options accordingly intsconfig.json :

```

Copy { "compilerOptions":{ "outDir":"./dist", "target":"es2017", "module":"esnext", "moduleResolution":"node", "allowSyntheticDefaultImports":true, "skipLibCheck":true }, "exclude":["node_modules"] }

```

Addtype andscripts as root-level entries topackage.json - they may already exist, so just replace them with the following.

```

Copy { ... "type":"module", "scripts":{ "xtransfer":"tsc && node dist/xtransfer.js" } ... }

```

1. Install dependencies

Install the latest beta version of Connext SDK and ethers.

```

Copy npminstall@connext/sdk npminstallethers@^5

```

1. The code

First, we'll configure the SDK. Create aconfig.ts file with the following contents.

```

Copy import{ SdkConfig }from"@connext/sdk"; import{ ethers }from"ethers";

// Create a Signer and connect it to a Provider on the sending chain constprivateKey="";

letsigner=newethers.Wallet(privateKey);

// Use the RPC url for the origin chain
constprovider=newethers.providers.JsonRpcProvider("https://public.stackup.sh/api/v1/node/ethereum-sepolia");
```

signer=signer.connect(provider); constsignerAddress=awaitsigner.getAddress();

constsdkConfig:SdkConfig={ signerAddress:signerAddress, // Use mainnet when you're ready... network:"testnet", environment:"production",

// Add more chains here! Use mainnet domains if network: mainnet. // This information can be found at https://docs.connext.network/resources/supported-chains chains:{ 1936027759:{ providers: ["https://public.stackup.sh/api/v1/node/ethereum-sepolia"] }, 1869640549:{ providers:['https://sepolia.optimism.io'] }, }, };

export{ signer,sdkConfig };

```

Replace with your own private key on line 5.

Notice that the config supports Goerli and Optimism-Goerli. We've also hard-coded the origin chain provider on line 10.

Now create axtransfer.ts file with the following:

```

Copy import{ create }from"@connext/sdk"; import{ BigNumber }from"ethers"; import{ signer,sdkConfig }from"./config.js";

const{sdkBase}=awaitcreate(sdkConfig);

constsignerAddress=awaitsigner.getAddress();

// xcall parameters constoriginDomain="1936027759"; constdestinationDomain="1869640549"; constoriginAsset="0xd26e3540A0A368845B234736A0700E0a5A821bBA"; constamount="100000000000000"; constslippage="10000";

// Estimate the relayer fee constrelayerFee=(awaitsdkBase.estimateRelayerFee({ originDomain, destinationDomain })).toString();

// Prepare the xcall params constxcallParams={ origin:originDomain,// send from Sepolia destination:destinationDomain,// to Op-Sepolia to:signerAddress,// the address that should receive the funds on destination asset:originAsset,// address of the token contract delegate:signerAddress,// address allowed to execute transaction on destination side in addition to relayers 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:"0x",// empty calldata for a simple transfer (byte-encoded) relayerFee:relayerFee,// fee paid to relayers };

// Approve the asset transfer if the current allowance is lower than the amount. // Necessary because funds will first be sent to the Connext contract in xcall. constapproveTxReq=awaitsdkBase.approveIfNeeded(originDomain, originAsset, amount)

if(approveTxReq) { constapproveTxReceipt=awaitsigner.sendTransaction(approveTxReq); awaitapproveTxReceipt.wait(); }

// Send the xcall constxcallTxReq=awaitsdkBase.xcall(xcallParams); xcallTxReq.gasLimit=BigNumber.from("20000000"); constxcallTxReceipt=awaitsigner.sendTransaction(xcallTxReq); console.log(xcallTxReceipt); awaitxcallTxReceipt.wait();

```

Most of the parameters are hardcoded in this example. For a detailed description of each parameter, see the SDK reference forxcall .

Information like asset addresses be found in the Deployments page.

1. Run it

Fire off the cross-chain transfer!

```

Copy npmrunxtransfer

```

1. Track thexcall

We can now use the transactionhash from the logged transaction receipt to track the status of thisxcall .

After the transfer isstatus: Executed on the destination side, the transferred tokens should show up in the recipient wallet.

Previous Frontend Next Estimating Fees Last updated8 days ago On this page * Cross-Chain Transfer * 1. Setup * 2. Install
```

Edit on GitHub