# Quickstart

Quickstart (Greeter)

This quickstart will teach you how to usexcall , the cross-chain communication primitive, to send funds and data across chains.

Introduction

In this guide, we will build a cross-chain Greeter. TheDestinationGreeter contract on the destination chain has anupdateGreeting function that changes a storedgreeting variable. TheSourceGreeter contract on the origin chain usesxcall to send encoded calldata forupdateGreeting .

To demonstrate a combination of an asset transfer and an arbitrary call in a singlexcall , theupdateGreeting function will require a payment to update the greeting. For this example, the contract will be okay with any amount greater than 0.

updateGreeting is implemented as an unauthenticated call (there are no checks to determinewho is calling the function). Therefore, this type ofxcall will be go through the "Fast Path".

If you prefer to fork a repo instead of following this step-by-step guide, ourxapp-starter kit contains a full example of this quickstart (plus more) and is compatible with both Hardhat and Foundry.

Prerequisites

- Node v18 installed
- Follow the instructions to installNode.js
- and useNode.js v18
- . We also recommend installingnvm
- , a node version manager, which will make switching versions easier.
- An Ethereum development environment like Foundry, Hardhat, Truffle, etc.
- This guide will be using Hardhat. Follow the instructions to installHardhat
- .
- If you don't already have gas funds on Goerli, try these faucets to get some:
-
    - https://goerli-faucet.mudit.blog/ (Requires Twitter account)
-
    - https://goerlifaucet.com/ (Requires signing up with Alchemy)
- *
-

Create a new project

Create a new project by running the following command:

```

Copy npxhardhat 888888888888888 888888888888888 888888888888888
88888888888888b.888d888.d8888888888b.8888b.888888 888888"88b 888P"d88" 888 888 "88b"88b 888 888 888
.d888888 888 888 888 888 888 .d888888 888 888 888 888 888 888 Y88b 888 888 888 888 888 Y88b. 888 888
"Y888888888"Y88888 888 888 "Y888888"Y888

                    Welcome to Hardhat v2.12.1

? What do you want to do? … ❯ Create a JavaScript project Create a TypeScript project Create an empty hardhat.config.js Quit

```

Choose a Javascript project. Choosey on all of the prompts.

Install the latest version of Connext contracts package in your project:

```

Copy npminstall@connext/interfaces

```

Next, install the OpenZeppelin contract package:

```

Copy npminstall@openzeppelin/contracts

```

You'll need to manually install the library@openzeppelin/contracts-upgradeable

```

Copy npminstall@openzeppelin/contracts-upgradeable

```

Installdotenv to protect your private key needed to deploy your contract:

```

Copy npminstalldotenv

```

In the root of your project, create a new.env file. Here you will store your private key used to deploy your contract.

Update.env with the following line:

```

Copy PRIVATE_KEY = YOUR-PRIVATE-KEY-HERE

```

Source Contract

The source contract initiates the cross-chain operation withxcall and passes the encoded greeting into the call. Allxcall params are detailed here.

In the/contracts directory, create a new contract calledSourceGreeter.sol :

```

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";

/ *@titleSourceGreeter* @noticeExample source contract that updates a greeting on DestinationGreeter. */
contractSourceGreeter{ // The Connext contract on this domain IConnextpublicimmutableconnext;

// The token to be paid on this domain IERC20publicimmutabletoken;

// Slippage (in BPS) for the transfer set to 100% for this example uint256publicimmutableslippage=10000;

constructor(address_connext,address_token) { connext=IConnext(_connext); token=IERC20(_token); }

/*@noticeUpdates a greeting variable on the DestinationGreeter contract.@paramtarget Address of the DestinationGreeter contract.* @paramdestinationDomain The destination domain ID.*@paramnewGreeting New greeting to update to.* @paramrelayerFee The fee offered to relayers. */ functionxUpdateGreeting( addresstarget, uint32destinationDomain, stringmemorynewGreeting, uint256amount, uint256relayerFee )externalpayable{ 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);

// Encode calldata for the target contract call bytesmemorycallData=abi.encode(newGreeting);

connext.xcall{value:relayerFee}( destinationDomain,// _destination: Domain ID of the destination chain target,// _to: address of the target contract address(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: max slippage the user will accept in BPS (e.g. 300 = 3%) callData// _callData: the encoded calldata to send ); } }

```

xUpdateGreeting is what the user will call on origin to initiate thexcall .

One important detail to note is thatxUpdateGreeting is apayable method. This is necessary because arelayerFee in native gas is passed into thexcall . More on how this fee is determined later.

Compile Contract

Make sure the solidity compiler version in yourhardhat.config.js is at least0.8.17 .

```

Copy module.exports={ solidity:"0.8.17", };

```

Compile the contract with the following command:

```

Copy npxhardhatcompile

```

Note: Hardhat may require you to manually install dependencies for @nomicfoundation/hardhat-toolbox. If you get an error about missing dependencies for that plugin, run the following command:

```

Copy npm install --save-dev "@nomicfoundation/hardhat-network-helpers@^1.0.0" "@nomicfoundation/hardhat-chai-matchers@^1.0.0" "@nomiclabs/hardhat-ethers@^2.0.0" "@nomiclabs/hardhat-etherscan@^3.0.0" "@types/chai@^4.2.0" "@types/mocha@^9.1.0" "@typechain/ethers-v5@^10.1.0" "@typechain/hardhat@^6.1.2" "solidity-coverage@^0.8.1" "ts-node@>=8.0.0" "typescript@>=4.5.0"

```

Deploy Contract

Update thehardhat.config.js file:

```

Copy require("@nomicfoundation/hardhat-toolbox"); require('@openzeppelin/hardhat-upgrades'); require('dotenv').config();

module.exports={ solidity:"0.8.17", networks:{ goerli:{ url:"https://rpc.ankr.com/eth_goerli", accounts: [0x{process.env.PRIVATE_KEY}] } } };

```

Create a/scripts/deploySource.js file with the following:

```

Copy constmain=async()=>{ constsourceGreeterContract=awaithre.ethers.deployContract( "SourceGreeter", [ "0xFCa08024A6D4bCc87275b1E4A1E22B71fAD7f649",// Connext on Goerli "0x7ea6eA49B0b0Ae9c5db7907d139D9Cd3439862a1"// TEST on Goerli ] ); awaitsourceGreeterContract.waitForDeployment(); console.log("Contract deployed to:",awaitsourceGreeterContract.getAddress()); };

construnMain=async()=>{ try{ awaitmain(); process.exit(0); }catch(error) { console.log(error); process.exit(1); } };

runMain();

```

The addresses for Connext and supported tokens in different domains can be referencedhere . We'll be using Goerli as our origin domain and the TEST token for this contract.

Now run the deploy script:

```

Copy npxhardhatrunscripts/deploySource.js--networkgoerli
```

```

Output:Contract deployed to: 0x9Af84578B89FcA019580af02326388987A074ca1

Verify Contract

Add anetherScan section tohardhat.config.js with yourgoerli api key (note: Etherscan API keys for the mainnet explorers will work for testnets):

```

Copy module.exports={ solidity:"0.8.17", networks:{ goerli:{ url:"https://rpc.ankr.com/eth_goerli", // PRIVATE_KEY loaded from .env file accounts:[0x{process.env.PRIVATE_KEY}] } }, //highlight-start etherscan:{ apiKey:{ goerli:"YOUR-API-KEY-HERE", } } //highlight-end };

```

Using the contract address you just deployed, run thehardhat verify command, including the contract address and its constructor arguments:

```

Copy npx hardhat verify --network goerli 0x9Af84578B89FcA019580af02326388987A074ca1 0xFCa08024A6D4bCc87275b1E4A1E22B71fAD7f649 0x7ea6eA49B0b0Ae9c5db7907d139D9Cd3439862a1

```

If you run into any errors likeProviderError: Too Many Requests , then replace the public RPC url inhardhat.config.js with another one from https://chainlist.org/ or use your own private RPC from a provider like Infura or Alchemy.

Target Contract

In the/contracts directory, create another contract calledDestinationGreeter.sol :

All target contracts must implement Connext'sIXReceiver interface. This interface ensures that Connext can call the contract and pass necessary data.

```

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

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

/ *@titleDestinationGreeter* @noticeExample destination contract that stores a greeting. */
contractDestinationGreeterisIXReceiver{ stringpublicgreeting;

// The token to be paid on this domain IERC20publicimmutabletoken;

constructor(address_token) { token=IERC20(_token); }

/*@noticeThe receiver function as required by the IXReceiver interface.@devThe Connext bridge contract will call this function.* / functionxReceive( bytes32_transferId, uint256_amount, address_asset, address_originSender, uint32_origin, bytesmemory_callData )externalreturns(bytesmemory) { // Check for the right token require( _asset==address(token), "Wrong asset received" ); // Enforce a cost to update the greeting require( _amount>0, "Must pay at least 1 wei" );

// Unpack the _callData stringmemorynewGreeting=abi.decode(_callData,(string));

_updateGreeting(newGreeting); }

/*@noticeInternal function to update the greeting.@paramnewGreeting The new greeting.* /
function_updateGreeting(stringmemorynewGreeting)internal{ greeting=newGreeting; } }

```

Compile Contract

Compile:

```

Copy npxhardhatcompile

```

Deploy Contract

Add another entry tohardhat.config.js , this time for Optimism-Goerli.

```

Copy module.exports={ solidity:"0.8.17", networks:{ "goerli":{ url:"https://rpc.ankr.com/eth_goerli", accounts: [0x{process.env.PRIVATE_KEY}] }, //highlight-start "optimism-goerli":{ url:"https://goerli.optimism.io", accounts: [0x{process.env.PRIVATE_KEY}], // gasPrice: 800000 // you may need to set this manually if you get "transaction underpriced" } //highlight-end } };

```

Create ascripts/deployTarget.js file with the following:

```

Copy constmain=async()=>{ constdestinationGreeterContract=awaithre.ethers.deployContract( "DestinationGreeter", [ "0x68Db1c8d85C09d546097C65ec7DCBFF4D6497CbF"// TEST on Optimism-Goerli ] ); awaitdestinationGreeterContract.waitForDeployment(); console.log("Contract deployed to:",awaitdestinationGreeterContract.getAddress()); };

construnMain=async()=>{ try{ awaitmain(); process.exit(0); }catch(error) { console.log(error); process.exit(1); } };

runMain();

```

Then run the deploy script:

```

Copy npxhardhatrunscripts/deployTarget.js--networkoptimism-goerli

```

Output:Contract deployed to: 0xC4e508cEe84499958a84C3562e92bD9e71d7D38a

Verify Contract

Add anapiKey tohardhat.config.js foroptimism-goerli :

```

Copy module.exports={ solidity:"0.8.17", networks:{ goerli:{ url:"https://rpc.ankr.com/eth_goerli", // PRIVATE_KEY loaded from .env file accounts:[0x{process.env.PRIVATE_KEY}] } }, etherscan:{ apiKey:{ goerli:"YOUR-API-KEY-HERE", //highlight-start optimisticGoerli:"YOUR-API-KEY-HERE", //highlight-end } } };

```

Using the contract address you just deployed, verify it:

```

Copy npx hardhat verify --network optimism-goerli 0xC4e508cEe84499958a84C3562e92bD9e71d7D38a 0x68Db1c8d85C09d546097C65ec7DCBFF4D6497CbF

```

If you run into any errors likeProviderError: Too Many Requests , then replace the public RPC url inhardhat.config.js with another one from https://chainlist.org/ or use your own private RPC with a provider like Infura or Alchemy.

Executing the Transaction

You should try the following steps on your own deployed contracts. For the lazy ones, you can just use these contracts we've deployed already:

- SourceGreeter.sol
- DestinationGreeter.sol
-
```

Mint TEST Tokens

First, you will need some TEST tokens. Recall that the destination contract requires a payment > 0 TEST in order to update its greeting.

Since you'll be updating the greeting from the origin chain, you will need to acquire some TEST tokens on the origin chain.

You can use Etherscan to call functions on (verified) contracts. Go to theTEST Token on Etherscan and click on the "Write Contract" button.

A new tab will show up with all write functions of the contract. Connect your wallet, switch to the Goerli network, and enter the parameters for themint function:

- account
- :
- amount
- : 10000000000000000000
-
    - 10 TEST. You can actually mint however much you want.
- *
-

Approve TEST Tokens

Tokens will move fromUser's wallet =>SourceGreeter =>Connext =>DestinationGreeter .

The user must first approve a spending allowance of the TEST ERC20 to theSourceGreeter contract. Therequire clause starting on line 39 checks for this allowance.

Again, on the Etherscan page for the TEST token, enter the parameters for theapprove function:

- spender
- : 0x9Af84578B89FcA019580af02326388987A074ca1
-
    - This is the address ofSourceGreeter
-
    - .
- *
- amount
- : 10000000000000000000
-

Then "Write" to theapprove function.

ExecutexUpdateGreeting

Similarly to the approval function for TEST, navigate to theSourceGreeter contract on Etherscan. Fill out thexUpdateGreeting function parameters and "Write" to the contract.

Let's walk through the different parameters.

- xUpdateGreeting
- (payableAmount): 0.03
-
    - This is the native gas that you're sending into thexcall
-
    - . This value must match what you pass in asrelayerFee
-
    - , but note that it's in ETH units here and wei units inrelayerFee
-
    - .
- *
- target
- : 0xC4e508cEe84499958a84C3562e92bD9e71d7D38a
-
    - The address ofDestinationGreeter
-
    - .
- *
- destinationDomain

- : 1735356532
-
    - The Domain ID of the destination chain. You can find a mapping of Domain IDs here. For this example,DestinationGreeter
-
    - is deployed to Optimism-Goerli.
- *
- newGreeting
- : hello chain!
-
    - Whatever string you want to update the greeting to.
- *
- amount
- : 1000000000000000000
-
    - The amount of TEST tokens to pay. We send 1 TEST here.
- *
- relayerFee
- : 30000000000000000
-
    - 0.03 goerli ETH, in wei units. Just a conservative estimate for relayers on testnet.
-
    - IMPORTANT!
-
    - This is a fee paid to relayers, which are off-chain agents that help execute the final leg of the cross-chain transfer on destination. Relayers get paid in the origin chain's native asset. This is whySourceGreeter
-
    - passes the fee like so:
-
    - ```
-
    - Copy
-
    - connext.xcall{value:relayerFee}(...)
-
    - ```
- *
-

As a xApp developer, you have some tools available to estimate what thisrelayerFee should be. For now, there are offchain methods for doing so - check out the guide onEstimating Fees .

Track the xcall

After executingupdateGreeting , you can useConnextscan (testnet) to check the status of thexcall . Just search up the transaction hash from the execution transaction.

Note that if yourrelayerFee was too low, the explorer will prompt you to increase it.

CheckDestinationGreeter

DestinationGreeter should be updated in just a few minutes (because this call is unauthenticated!). Cross-chain calls are not always this fast - see our guide on Authentication.

Head over to theDestinationGreeter contract on Etherscan. This time, we'll go to theRead Contract tab and look at the value ofgreeting . It has updated!

Send a couple more updates fromSourceGreeter but make it a different string. At some point, your TEST allowance toHelloSource will run out and you'll need to do the approval dance again.

Congrats! You've gone cross-chain!

Next Steps

- Trytracking the status
- of anxcall
- after you send it.
- Learn aboutauthentication
- and important security considerations.
- See hownested xcalls

- can open up infinite cross-chain possibilities.
- Fork thexApp Starter Kit
- (includes code for this example) and build your own xApp.
-

Previous Introduction Next Guides Last updated9 months ago On this page *Quickstart (Greeter) * Introduction * Prerequisites * Create a new project * Source Contract * Target Contract * Executing the Transaction * Next Steps

Edit on GitHub