Transfer Tokens between EOAs

In this tutorial, you will use Chainlink CCIP to transfer tokens directly from yourEOA (Externally Owned Account) to an account on a different blockchain. First, you will pay for CCIP fees on the source blockchain using LINK. Then, you will run the same example paying for CCIP fees in native gas, such as ETH on Ethereum or MATIC on Polygon.

Transferring tokens

This tutorial uses the term "transferring tokens" even though the tokens are not technically transferred. Instead, they are locked or burned on the source chain and then unlocked or minted on the destination chain. Read theToken Pools section to understand the various mechanisms that are used to transfer value across chains.

Before you begin

1. Install Node.js 18 . Optionally, you can use thenvm package to switch between Node.js versions withnvm use 18.

node-v$node-vv18.7.0 2. YourEOA (Externally Owned Account) must have both ETH and LINK tokens onEthereum Sepoliato pay for the gas fees and CCIP fees.

- Configure MetaMask to use LINK tokens
- Acquire testnet ETH and LINK fromfaucets.chain.link
- Check theSupported Networks page to confirm that the tokens you will transfer are supported for your lane. In this example, you will transfer tokens fromEthereum SepoliatoPolygon Mumbaiso check the list of supported tokenshere .
- Learn how toacquire CCIP test tokens . After following this guide, yourEOA (Externally Owned Account) should have CCIP-BnM tokens, and CCIP-BnM should appear in the list of your tokens in MetaMask.
- In a terminal, clone thesmart-contract-examples repository and change to thesmart-contract-examples/ccip/offchain/javascriptdirectory.

gitclone https://github.com/smartcontractkit/smart-contract-examples.git&&\cdsmart-contract-examples/ccip/offchain/javascript 6. Runnpm installto install the dependencies.

npminstall 7. For higher security, the examples repository imports@chainlink/env-enc . Use this tool to encrypt your environment variables at rest.

1. Set an encryption password for your environment variables.

npx env-enc set-pw 2. Runnpx env-enc setto configure a.env.encfile with the basic variables that you need to send your requests toPolygon Mumbai.

- ETHEREUM_SEPOLIA_RPC_URL: Set a URL for theEthereum Sepoliatestnet. You can sign up for a personal endpoint fromAlchemy ,Infura , or another node provider service.
- POLYGON_MUMBAI_RPC_URL: Set a URL for thePolygon Mumbaitestnet. You can sign up for a personal endpoint fromAlchemy ,Infura , or another node provider service.
- PRIVATE_KEY: Find the private key for your testnet wallet. If you use MetaMask, follow the instructions toExport a Private Key .Note: The offchain script uses your private key to sign any transactions you make such as transferring tokens.

npx env-encset

Tutorial

Transfer tokens and pay in LINK

In this example, you will transfer CCIP-BnM tokens from your EOA onEthereum Sepoliato an account onPolygon Mumbai. The destination account could be anEOA (Externally Owned Account) or a smart contract. The example shows how to transfer CCIP-BnM tokens, but you can re-use the same example to transfer other tokens as long as they are supported for yourlane .

For this example, CCIP fees are paid in LINK tokens. To learn how to pay CCIP fees in native ETH, read thePay in native section. To see a detailed description of the example code, read thecode explanation section.

To transfer tokens and pay in LINK, use the following command:

node src/transfer-tokens.js sourceChain destinationChain destinationAccount tokenAddress amount feeTokenAddress ThefeeTokenAddressparameter specifies the token address for paying CCIP fees. The supported tokens for paying fees include LINK, the native gas token of the source blockchain (ETH for Ethereum), and the wrapped native gas token (WETH for Ethereum).

Complete the following steps in your terminal:

1. Send 1,000,000,000,000,000 (0.001 CCIP-BnM) from your EOA onEthereum Sepoliato another account onPolygon Mumbai:

node src/transfer-tokens.js ethereumSepolia polygonMumbai YOUR_ACCOUNT 0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05 1000000000000000 0x779877A7B0D9E8603169DdbD7836e478b4624789Command arguments:

ArgumentExplanationnode src/transfer-tokens.jsNode.js will execute the JavaScript code inside thetransfer-tokens.jsfile.ethereumSepoliaThis specifies the source blockchain, in this case,Ethereum Sepolia.polygonMumbaiThis specifies the destination blockchain, which isPolygon Mumbaiin this case.YOUR_ACCOUNTThis is the account address on the destination blockchain. You can replace this with your account address.0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05This is the CCIP-BnM token contract address on Ethereum Sepolia. The contract addresses for each network can be found on theSupported Networks page .1000000000000000This is the amount of CCIP-BnM tokens to be transferred. In this example, 0.001 CCIP-BnM are transferred.0x779877A7B0D9E8603169DdbD7836e478b4624789Since you will pay for CCIP fees in LINK, this is the LINK token contract address onEthereum Sepolia. The LINK contract address can be found on theLink Token contracts page . 2. Once you execute the command, you should see the following logs:

$ node src/transfer-tokens.js ethereumSepolia polygonMumbai 0x83dC44a4C00DFf69d0A0c7c94B20b53a4933BE0A 0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05 1000000000000000 0x779877A7B0D9E8603169DdbD7836e478b4624789 Estimated fees (wei): 45004872518799270 approved router 0x0BF3dE8c5D3e8A2B34D2BEeB17ABfCeBaf363A59 to spend 1000000000000000 of token 0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05. Transaction: 0x5216b9c85d576a6d97c80edaadd890f2a7ae0132e54e7bda9a102374de3c3003 approved router 0x0BF3dE8c5D3e8A2B34D2BEeB17ABfCeBaf363A59 to spend fees 45004872518799270 of token 0x779877A7B0D9E8603169DdbD7836e478b4624789. Transaction: 0x63ca647e8fd8af177eea85ea02f18e6ece00f7c0e7d7917b050221141cd47a72

✅ 1000000000000000 of Tokens(0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05) Sent to account 0x83dC44a4C00DFf69d0A0c7c94B20b53a4933BE0A on destination chain polygonMumbai using CCIP. Transaction hash 0x7dda59032f1f157a830755264907139602f3fee73666938e12d17c26923c9e07 - Message id is 0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2

Wait for message 0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2 to be executed on the destination chain - Check the explorer https://ccip.chain.link/msg/0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2 ... 3. Analyze the logs:

- The script communicates with the router to calculate the transaction fees required to transfer tokens, which amounts to 45,004,872,518,799,270 Juels (equivalent to 0,045 LINK).
- The script engages with the Link token contract, authorizing the router contract to spend 45,004,872,518,799,270 Juels for the fees and 1000000000000000 (0.001 CCIP-BnM) from your Externally Owned Account (EOA) balance.
- The script initiates a transaction through the router to transfer 1000000000000000 (0.001 CCIP-BnM) to your account onPolygon Mumbai. It also returns the CCIP message ID.
- The script continuously monitors the destination blockchain (Polygon Mumbai) to track the progress and completion of the cross-chain transaction.
- While the script is waiting for the cross-chain transaction to proceed, open theCCIP explorer and search your cross-chain transaction using the message ID. Notice that the status isWaiting for finality.
- After several minutes (the waiting time depends on thefinality of the source blockchain), the script will complete the polling process, and the following logs will be displayed:

Message 0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2 has not been processed yet on the destination chain.Try again in 60sec - Check the explorer https://ccip.chain.link/msg/0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2

✅Status of message 0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2 is SUCCESS - Check the explorer https://ccip.chain.link/msg/0x4ea8080f2a51377247e93b5f45c00330b8c4fde3043e99847d0cab734f473df2 6. Open theCCIP explorer and use the message ID to find your cross-chain transaction. 7. The data field is empty because only tokens are transferred. The gas limit is set to 0 because the transaction is directed to an Externally Owned Account (EOA). With an empty data field, no function calls on a smart contract are expected on the destination chain.

Transfer tokens and pay in native

In this example, you will transfer CCIP-BnM tokens from your EOA onEthereum Sepoliato an account onPolygon Mumbai. The destination account could be anEOA (Externally Owned Account) or a smart contract. The example shows how to transfer LINK tokens, but you can re-use the same example to transfer other tokens as long as they are supported for yourlane .

For this example, CCIP fees are paid in Sepolia's native ETH. To learn how to pay CCIP fees in LINK, read thePay in LINK section. To see a detailed description of the example code, read thecode explanation section.

To transfer tokens and pay in native, use the following command:

node src/transfer-tokens.js sourceChain destinationChain destinationAccount tokenAddress amount Complete the following steps in your terminal:

1. Send 1,000,000,000,000,000 (0.001 CCIP-BnM) from your EOA onEthereum Sepoliato another account onPolygon Mumbai:

node src/transfer-tokens.js ethereumSepolia polygonMumbai YOUR_ACCOUNT 0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05 1000000000000000Command arguments:

ArgumentExplanationnode src/transfer-tokens.jsNode.js will execute the JavaScript code inside thetransfer-tokens.jsfile.ethereumSepoliaThis specifies the source blockchain, in this case,Ethereum Sepolia.polygonMumbaiThis specifies the destination blockchain, which isPolygon Mumbaiin this case.YOUR_ACCOUNTThis is the account address on the destination blockchain. Replace this with your account address.0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05This is the CCIP-BnM token contract address onEthereum Sepolia. The contract addresses for each network can be found on theSupported Networks page .1000000000000000This is the amount of CCIP-BnM tokens to be transferred. In this example, 0.001 CCIP-BnM are transferred. 2. After you execute the command, you should see the following logs:

$ node src/transfer-tokens.js ethereumSepolia polygonMumbai 0x83dC44a4C00DFf69d0A0c7c94B20b53a4933BE0A 0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05 1000000000000000 Estimated fees (wei): 278051815072596 approved router 0x0BF3dE8c5D3e8A2B34D2BEeB17ABfCeBaf363A59 to spend 1000000000000000 of token 0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05. Transaction: 0x85d91b16c08546ad5e8e7791cfe4ebf59cc986f805244230872487fe8ad6862d

✅ 1000000000000000 of Tokens(0xFd57b4ddBf88a4e07fF4e34C487b99af2Fe82a05) Sent to account 0x83dC44a4C00DFf69d0A0c7c94B20b53a4933BE0A on destination chain polygonMumbai using CCIP. Transaction hash 0x358c2abccd75c52aeabf5b330fda2b17a4c29bfc2612fc85a8619d22c3d1be00 - Message id is 0x1dc9f4d0e798802b19668a26b59681b97c02ad16bef3765eae6c5232f1d2e050

Wait for message 0x1dc9f4d0e798802b19668a26b59681b97c02ad16bef3765eae6c5232f1d2e050 to be executed on the destination chain - Check the explorer https://ccip.chain.link/msg/0x1dc9f4d0e798802b19668a26b59681b97c02ad16bef3765eae6c5232f1d2e050 ... 3. Analyze the logs:

- The script communicates with the router to calculate the transaction fees required to transfer tokens, which amounts to 278,051,815,072,596 wei (equivalent to 0.000278 ETH).
- The script engages with the CCIP-BnM token contract, authorizing the router contract to deduct 0.001 CCIP-BnM from your Externally Owned Account (EOA) balance.
- The script initiates a transaction through the router to transfer 0.001 CCIP-BnM tokens to your destination account onPolygon Mumbai. It also returns the CCIP message ID.
- The script continuously monitors the destination blockchain (Polygon Mumbai) to track the progress and completion of the cross-chain transaction.
- The transaction time depends on thefinality of the source blockchain . After several minutes, the script will complete the polling process and the following logs will be displayed:

Message 0xa57bd1a6a343b78e1b05a78b1725bfd43a418601ca3d2d24118c8a5f7a81e4ae is not processed yet on destination chain.Try again in 60sec - Check the explorer https://ccip.chain.link/msg/0xa57bd1a6a343b78e1b05a78b1725bfd43a418601ca3d2d24118c8a5f7a81e4ae

✅Status of message 0xa57bd1a6a343b78e1b05a78b1725bfd43a418601ca3d2d24118c8a5f7a81e4ae is SUCCESS - Check the explorer https://ccip.chain.link/msg/0xa57bd1a6a343b78e1b05a78b1725bfd43a418601ca3d2d24118c8a5f7a81e4ae 5. Open theCCIP explorer and use the message ID to find your cross-chain transaction. 6. The data field is empty because only tokens are transferred. The gas limit is set to 0 because the transaction is directed to an Externally Owned Account (EOA), so no function calls are expected on the destination chain.

Code explanation

The Javascript featured in this tutorial is designed to interact with CCIP to transfer tokens. The contract code includes several code comments to clarify each step, but this section explains the key elements.

Imports

The script starts by importing the necessary modules and data. It imports ethers.js and ABIs (Application Binary Interface) from a config file for different contracts and configurations.

Handling arguments

ThehandleArgumentsfunction validates and parses the command line arguments passed to the script.

Main function: transferTokens

This asynchronous function,transferTokensperforms the token transfer.

Initialization

The script initializes ethers providers to communicate with the blockchains in this section. It parses source and destination router addresses and blockchain selectors. A signer is created to sign transactions.

Token validity check

The script fetches a list of supported tokens for the destination chain and checks if the token you want to transfer is supported.

Building the CCIP message

A Cross-Chain Interoperability Protocol (CCIP) message is built, which will be sent to the router contract. It includes the destination account, amount, token address, and additional parameters.

Fee calculation

The script calls the router to estimate the fees for transferring tokens.

Transferring tokens

This section handles the actual transferring of tokens. It covers three cases:

- Fees paid using the native gas token:The contract makes one approval for the transfer amount. The fees are included in thevaluetransaction field.
- Fees paid using an asset different from the native gas token and the token being transferred:The contracts makes two approvals. The first approval is for the transfer amount and the second approval is for the fees.
- Fees paid using the same asset that is being transferred, but not the native gas token:The contract makes a single approval for the sum of the transfer amount and fees.

The script waits for the transaction to be validated and stores the transaction receipt.

Fetching message ID

The router'sccipSendfunction returns a message ID. The script simulates a call to the blockchain to fetch the message ID that the router returned.

Checking the status on the destination chain

The script polls the off-ramp contracts on the destination chain to wait for the message to be executed. If the message is executed, it returns the status. Otherwise, the message times out after 40 minutes.