testnets)

- Simulating Interactions with Tenderly

Was this helpful? Export as PDF

Testing your integration

Testnets

We do not support testnets (e.g., Sepolia) as we have limited user engagement with them. We recommend using mainnets like Gnosis and Polygon for the following reasons:

- Limited User Engagement
- : There are few users actively utilizing testnets
- Maintenance Issues
- : Testnets often experience interruptions, especially during significant updates
- Insufficient Liquidity
- : Tokens on testnets generally lack the liquidity needed for thorough testing
- Cost-Effectiveness
- : Testing on mainnets like Gnosis and Polygon is more reliable and cost-effective

Simulating Interactions with Tenderly

You can simulate your interactions with LI.FI usingTenderly . Below are two methods to perform simulations

Using the Tenderly Interface

1. Create a Tenderly Account
2. :
3.
    - Sign up for a Tenderly account
4.
    - Create a new project in the Tenderly dashboard
5. Access the Simulation Feature
6. :
7.
    - Navigate to theSimulation Feature
8.
    - tab
9.
    - Fill in the required fields using data extracted from the quote response. Map the necessary fields from the returnedtransactionRequest
10.
    - object
11. Run the Simulation
12. :
13.
    - ClickRun Simulation
14.
    - to execute the transaction simulation

Code Implementation

This option involves using the Tenderly API for simulations

Prerequisites

- Tenderly Account
- : Ensure you have an account and project set up on Tenderly
- Tenderly API Key
- : Obtain your API key from the Tenderly dashboard
- Optional
- : Install the Tenderly CLI:
- ```
- Copy
- npm install-g @tenderly/cli

- ```

Example Code

The following example demonstrates how to implement a transaction simulation using the Tenderly API with Node.js

Network ID : Ensure you're using the correct network ID

Tenderly Headers : Include yourX-Access-Key in the API request headers ```

Copy constethers=require('ethers'); constaxios=require('axios');

// Replace with your Tenderly project details constTENDERLY_PROJECT="your-tenderly-project"; constTENDERLY_ACCESS_KEY="your-tenderly-api-key"; constTENDERLY_USER="your-tenderly-username";

// LI.FI API URL constAPI_URL='https://li.quest/v1';

// Tenderly API URL for simulation const TENDERLY_API_URL = https://api.tenderly.co/api/v1/account{TENDERLY_USER}/project{TENDERLY_PROJECT}/simulate;

// Get a quote for your desired transfer constgetQuote=async(fromChain,toChain,fromToken,toToken,fromAmount,fromAddress)=>{ constresult=awaitaxios.get(https://li.quest/v1/quote,{ params:{ fromChain, toChain, fromToken, toToken, fromAmount, fromAddress, } }); returnresult.data; }

// Simulate the transaction using Tenderly constsimulateTransaction=async(transactionRequest)=>{ constpayload={ "network_id":"100",// Use the appropriate network ID "from":transactionRequest.from, "to":transactionRequest.to, "input":transactionRequest.data, "gas":transactionRequest.gasLimit.toString(), "gas_price":transactionRequest.gasPrice.toString(), "value":transactionRequest.value.toString() };

constresponse=awaitaxios.post(TENDERLY_API_URL,payload,{ headers:{ 'X-Access-Key':TENDERLY_ACCESS_KEY } });

returnresponse.data; }

// Example usage construn=async()=>{ constfromChain='DAI'; constfromToken='USDC'; consttoChain='POL'; consttoToken='USDC'; constfromAmount='1000000'; constfromAddress='YOUR_WALLET_ADDRESS';

// Set up your wallet constprovider=newethers.providers.JsonRpcProvider('https://rpc.xdaichain.com/',100); constwallet=ethers.Wallet.fromMnemonic('YOUR_PERSONAL_MNEMONIC').connect(provider);

constquote=awaitgetQuote(fromChain,toChain,fromToken,toToken,fromAmount,fromAddress);

// Simulate transaction using Tenderly constsimulationResult=awaitsimulateTransaction(quote.transactionRequest);

console.log('Simulation Result:',simulationResult); }

run().then(()=>{ console.log('DONE!'); });

``` Last updated1 month ago
```