Cross-Chain Suapps

It is often the case that Suapps need to connect to two (or more!) different blockchains in order to be effective.

suavex-foundry

The other option is to spoof Ethereum L1 (or chains like it) using Anvil, a tool which is also well suited to making your cross-chain life easier.

For this reason, we maintain a fork of foundry which specifically implements suavex_call . The historical context for this tool can be found in this forum post .

The idea is that, instead of running your own L1 node, you simply run Anvil via an RPC provider, and use that to test any croos chain interactions:

1. clone the repo & go into its directory:

git clone https://github.com/flashbots/suavex-foundry cd suavex-foundry 1. run anvil with cargo:

export

RPC_URL

https://REPLACE_WITH_YOUR_MAINNET_RPC_URL cargo run --bin anvil -- -p

8555 --chain-id 1

-f

RPC_URL The creator of suavex-foundry wrote an example Suapp called unisuapp , which features a simple intent-based public mempool and solver-driven intent execution using Uniswap v2. Unisuapp illustrates well how to use suavex-foundry to test Suapps that work across chains. You can get it running by following these steps:

1. Clone the repo and enter its directory

git clone https://github.com/zeroXbrock/unisuapp cd unisuapp 1. Populate .env 2. , which uses pre-funded default accounts

echo

"L1_CHAIN_ID=1 L1_KEY=ac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80
L1_RPC_URL=http://localhost:8555
SUAVE_KEY=91ab9a7e53c220e6210460b65a7a3bb2ca181412a8a7b43ff336b3df1737ce12
SUAVE_RPC_URL=http://localhost:8545"

.env 1. Install solidity dependencies & build contracts

forge install forge build 1. Install NPM dependencies

bun install 1. Run suave-geth 2. with the external-whitelist flag:

suave-geth --suave.dev --suave.eth.external-whitelist = '*' 1. Run the script with DEPLOY set to deploy the contracts. Deployed contract address will be saved to addresses.json 2. :

DEPLOY

true bun run index.ts 1. Run the script again without deploying to see the logic at work:

bun run index.ts

Kurtosis

There is a kurtosis setup for the private Order Flow Auction Suapp example : this example demonstrates the flexibility and power of SUAVE, as it happens across two different chains, which creates some infrastructure and testing challenges. Kurtosis + Docker is one way to make this slightly easier and ensure that you have all the services running that you need to execute and test all your Suapp's intended actions. Edit this page Previous Using SUAVE Testnets Next SUAVE Concepts