D4D4D4;--ch-t-background: #1E1E1E;--ch-t-lighter-inlineBackground: #1e1e1ee6;--ch-t-editor-background: #1E1E1E;--ch-t-editor-foreground: #D4D4D4;--ch-t-editor-rangeHighlightBackground: #ffffff0b;--ch-t-editor-infoForeground: #3794FF;--ch-t-editor-selectionBackground: #264F78;--ch-t-focusBorder: #007FD4;--ch-t-tab-activeBackground: #1E1E1E;--ch-t-tab-activeForeground: #ffffff;--ch-t-tab-inactiveBackground: #2D2D2D;--ch-t-tab-inactiveForeground: #ffffff80;--ch-t-tab-border: #252526;--ch-t-tab-activeBorder: #1E1E1E;--ch-t-editorGroup-border: #444444;--ch-t-editorGroupHeader-tabsBackground: #252526;--ch-t-editorLineNumber-foreground: #858585;--ch-t-input-background: #3C3C3C;--ch-t-input-foreground: #D4D4D4;--ch-t-icon-foreground: #C5C5C5;--ch-t-sideBar-background: #252526;--ch-t-sideBar-foreground: #D4D4D4;--ch-t-sideBar-border: #252526;--ch-t-list-activeSelectionBackground: #094771;--ch-t-list-activeSelectionForeground: #fffffe;--ch-t-list-hoverBackground: #2A2D2E; }

AI agent swaps on Uniswap

You can find a working code example to run locally in ourexample repository(opens in a new tab).

Requirements

- A deployed Safe Smart Account
- The AI agent is a signer on the Safe
- This example assumes, that the threshold of the Safe Smart Account is one, so the AI agent can sign autonomously.
- If you require more signatures, you have to collect those signatures programmatically of with theSafe Wallet(opens in a new tab)
- .
- This guide assumes the Safe owns WETH.
- The example repository shows how to swap ETH to WETH.

Swap on Uniswap

Here is a quick guide to get you up and running:

Setup the Safe Smart Account

Your Safe Smart Account should be deployed. Now, initialize an instance with the SafeProtocol Kit :

_10 import Safe from "@safe-global/protocol-kit"; _10 _10 const preExistingSafe = await Safe.init({ _10 provider: RPC_URL, _10 signer: AGENT_PRIVATE_KEY, _10 safeAddress: SAFE_ADDRESS, _10 });

Fetch Uniswap pool data

First, you have to fetch the pool data from Uniswap. This data provides information about the liquidity at the current and at other prices. Uniswap has a uniquePricing Math(opens in a new tab).

_26 import { _26 Address, _26 createPublicClient, _26 createWalletClient, _26 defineChain, _26 encodeFunctionData, _26 http, _26 PublicClient, _26 } from "viem"; _26 _26 // Fetch slot0 data (current price, tick, etc.) _26 const slot0 = (await publicClient.readContract({ _26 address: poolAddress, _26 abi: POOL_ABI, _26 functionName: "slot0", _26 })) as any; _26 _26 // Fetch liquidity _26 const liquidity = (await publicClient.readContract({ _26 address: poolAddress, _26 abi: POOL_ABI, _26 functionName: "liquidity", _26 })) as any; _26 _26 const sqrtPriceX96 = BigInt(slot0[0]); _26 const tick = slot0[1];

Execute Swap

Now, you can setup your Safe Smart Account and send a swap transaction to Uniswap:

_104 import { _104 FeeAmount, _104 Pool, _104 Route, _104 SwapRouter, _104 CurrencyAmount, _104 TradeType, _104 Percent _104 } from "@uniswap/v3-sdk"; _104 import { Token, SwapOptions } from "@uniswap/sdk-core"; _104 import JSBI from "jsbi"; _104 import { OperationType, MetaTransactionData } from "@safe-global/types-kit"; _104 _104 // Set up viem clients and accounts _104 const account = privateKeyToAccount(AGENT_PRIVATE_KEY as 0x{string}); _104 _104 const publicClient = createPublicClient({ _104 transport: http(RPC_URL!) _104 }); _104 const walletClient = createWalletClient({ _104 transport: http(RPC_URL!) _104 }); _104 _104 const chainId = (await publicClient.getChainId()); _104 _104 // Example Values for WETH/USDC Uniswap Pool on Sepolia: _104 const WETH_ADDRESS = "0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2"; _104 const USDC_ADDRESS = "0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48"; _104 const USDC_ETH_POOL_ADDRESS = "0x88e6A0c2dDD26FEEb64F039a2c41296FcB3f5640"; _104 const SWAP_ROUTER_ADDRESS = "0xE592427A0AEce92De3Edee1F18E0157C05861564"; // Uniswap V3 Router _104 const INPUT_AMOUNT = "100000000000"; // Amount of ETH to swap to USDC _104 const OUTOUT_AMOUNT = "0"; // 0 USDC _104 _104 // Define token details _104 const USDC = new Token(chainId, USDC_ADDRESS, 6, "USDC", "USD Coin"); _104 const WETH = new Token(chainId, WETH_ADDRESS, 18, "WETH", "Wrapped Ether"); _104 _104 const callDataApprove = encodeFunctionData({ _104 abi: WETH_ABI, _104 functionName: "approve", _104 args: [SWAP_ROUTER_ADDRESS, INPUT_AMOUNT], _104 }); _104 _104 const safeApproveTx: MetaTransactionData = { _104 to: WETH_ADDRESS, _104 value: "0", _104 data: callDataApprove, _104 operation: OperationType.Call, _104 }; _104 _104 const options: SwapOptions = { _104 slippageTolerance: new Percent(50, 10_000), // 50 bips, or 0.50% _104 deadline: Math.floor(Date.now() / 1000) + 60 * 20, // 20 minutes from the current Unix time _104 recipient: SAFE_ADDRESS, _104 }; _104 _104 const poolInfo = await fetchPoolData(publicClient, USDC_ETH_POOL_ADDRESS); _104 _104 // Create the pool object _104 const pool = new Pool(_104 WETH, _104 USDC, _104 FeeAmount.MEDIUM, _104 JSBI.BigInt(poolInfo.sqrtPriceX96.toString()), _104 JSBI.BigInt(poolInfo.liquidity.toString()), _104 poolInfo.tick _104); _104 _104 const swapRoute = new Route([pool], WETH, USDC); _104 _104 const uncheckedTrade = Trade.createUncheckedTrade({ _104 tradeType: TradeType.EXACT_INPUT, _104 route: swapRoute, _104 inputAmount: CurrencyAmount.fromRawAmount(WETH, _104 INPUT_AMOUNT _104), _104 outputAmount: CurrencyAmount.fromRawAmount(USDC, OUTOUT_AMOUNT), _104 }); _104 _104 const methodParameters = SwapRouter.swapCallParameters(_104 [uncheckedTrade], _104 options _104); _104 _104 const safeSwapTx: MetaTransactionData = { _104 to: SWAP_ROUTER_ADDRESS, _104 value: methodParameters.value, _104 data: methodParameters.calldata, _104 operation: OperationType.Call, _104 }; _104 _104 const safeTx = await preExistingSafe.createTransaction({ _104 transactions: [safeApproveTx, safeSwapTx], _104 onlyCalls: true, _104 }); _104 _104 // You might need to collect more signatures here, depending on the threshold _104 _104 const txResponse = await preExistingSafe.executeTransaction(safeTx); _104 await publicClient.waitForTransactionReceipt({ _104 hash: txResponse.hash as 0x{string}, _104 }); _104 _104 console.log(Deposit and approve transaction: [{txResponse.hash}]); Now your AI agent executed a swap on Uniswap.

Next steps

You can find more information about Swaps on Uniswap in theirdocs about swaps(opens in a new tab).

If you have a technical question about Safe Smart Accounts, feel free to reach out onStack Exchange(opens in a new tab) with the safe-core tag.

AI agent swaps on CoW Swap Glossary Was this page helpful?

Report issue