Account Abstraction

Effortlessly create and manage smart accounts for your users with just a few lines of code, using our native account abstraction. Additionally, SCWs offer enhanced control and programmability, enabling features like those listed below.

- Gas Abstraction: Cover transaction fees for users, or allow users to pay for their own
- transactions using ERC-20 tokens.
- Batch Transactions: Perform multiple transactions in a single call.
- Automated Transactions: Allow users to automate actions, like swapping ETH to USDT when ETH hits
- 2,700.
- Set Spending Limits: Allow users to set tailored spending limits.

Our native AA integration streamlines your setup, allowing you to create and manage smart accounts using your favorite libraries like Viem, Ethers, and Web3.js. With this, you don't need to rely on third party packages to effortlessly create ERC-4337 compatible Smart Contract Wallets (SCWs), and give users the ability to perform batch transactions and efficiently manage gas sponsorship.

Web3Auth's native account abstraction gives you the flexibility to choose your preferred account abstraction provider, configure your bundler client, and integrate your paymaster. For more insights into how ERC-4337 works and its components,check out our detailed blog post.

note This is a paid feature and the minimumpricing plan to use this SDK in a production environment is theGrowth Plan . You can use this feature in Web3Auth Sapphire Devnet network for free.

Installationâ

To use native account abstraction, you'll need to install the@web3auth/account-abstraction-provider , which allows you to create and interact with Smart Contract Wallets (SCWs). This provider simplifies the entire process by managing the complex logic behind configuring the account abstraction provider, bundler, and preparing user operations.

```
npm install --save @web3auth/account-abstraction-provider
```

Configureâ

When instantiating the Account Abstraction Provider, you can pass configuration objects to the constructor. These configuration options allow you to select the desired Account Abstraction (AA) provider, as well as configure the bundler and paymaster, giving you flexibility and control over the provider.

```
import

{ AccountAbstractionProvider , SafeSmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new
```

AccountAbstractionProvider ({ config :

{ chainConfig , smartAccountInit :

new

SafeSmartAccount ( ) , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/11155111/rpc?apikey= { pimlicoAPIKey } , } , } , }) ; Please note this is the important step for setting the Web3Auth account abstraction provider.

- [Configure Smart Account provider]
- [Configure Bundler]
- [Configure Sponsored Paymaster]
- [Configure ERC-20 Paymaster]

Configure Smart Account Providerâ

Web3Auth offers the flexibility to choose your preferred Account Abstraction (AA) provider. Currently, we support Safe, Kernel, Biconomy, and Trust.

- Safe
- Kernel
- Biconomy
- Trust

import

{ AccountAbstractionProvider , SafeSmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , smartAccountInit :

new

SafeSmartAccount ( ) , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/11155111/rpc?apikey= { pimlicoAPIKey } , } , } , }) ; import

{ AccountAbstractionProvider , KernelSmartAccount , } from

"@web3auth/account-abstraction-provider" ; const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , smartAccountInit :

new

KernelSmartAccount ( ) , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/11155111/rpc?apikey= { pimlicoAPIKey } , } , } , }) ; import

{ AccountAbstractionProvider , BiconomySmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , smartAccountInit :

new

BiconomySmartAccount ( ) , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/11155111/rpc?apikey= { pimlicoAPIKey } , } , } , }) ; import

{ AccountAbstractionProvider , TrustSmartAccount , }

from

"@web3auth/account-abstraction-provider" ; const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , smartAccountInit :

new

TrustSmartAccount ( ) , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/11155111/rpc?apikey= { pimlicoAPIKey } , } , } , }) ;

Configure Bundlerâ

Web3Auth enables you to configure your bundler and define the paymaster context. The bundler aggregates the UserOperations and submits them on-chain via a global entry point contract.

Bundler support is not limited to the examples belowâyou can use any bundler of your choice.

- Pimlico
- ZeroDev

import

{ AccountAbstractionProvider , SafeSmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

```
"Ethereum Sepolia Testnet" , blockExplorerUrl :
"https://sepolia.etherscan.io" , ticker :
"ETH" , tickerName :
"Ethereum" , logo :
"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;
const accountAbstractionProvider =
new
AccountAbstractionProvider ( { config :
{ chainConfig , smartAccountInit :
new
SafeSmartAccount ( ) , bundlerConfig :
{ // Get the pimlico API Key from dashboard.pimlico.io url :
https://api.pimlico.io/v2/ { chainId } /rpc?apikey= { pimlicoAPIKey } , } , } , } ) ; import
{ AccountAbstractionProvider , KernelSmartAccount , }
from
"@web3auth/account-abstraction-provider" ;
const chainConfig =
{ chainNamespace :
CHAIN_NAMESPACES . EIP155 , chainId :
"0xaa36a7" , rpcTarget :
"https://rpc.ankr.com/eth_sepolia" , displayName :
"Ethereum Sepolia Testnet" , blockExplorerUrl :
"https://sepolia.etherscan.io" , ticker :
"ETH" , tickerName :
"Ethereum" , logo :
"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;
const accountAbstractionProvider =
new
AccountAbstractionProvider ( { config :
{ chainConfig , smartAccountInit :
new
KernelSmartAccount ( ) , bundlerConfig :
{ // Get the ZeroDev API from https://dashboard.zerodev.app/ url :
https://rpc.zerodev.app/api/v2/bundler/ { projectId } , } , } , } ) ;
```

Configure Paymasterâ

You can configure the paymaster of your choice to sponsor gas fees for your users, along with the paymaster context. The paymaster context lets you set additional parameters, such as choosing the token for ERC-20 paymasters, defining gas policies, and more.

Sponsored Paymasterâ

- Pimlico
- ZeroDev

```
import

{ AccountAbstractionProvider , SafeSmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ( { config :

{ chainConfig , smartAccountInit :

new

SafeSmartAccount ( ) , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/ { chainId } /rpc?apikey= { pimlicoAPIKey } , } , paymasterConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/ { chainId } /rpc?apikey= { pimlicoAPIKey } , } , } , } ) ; import
```

```
{ AccountAbstractionProvider , SafeSmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :
```

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , smartAccountInit :

new

SafeSmartAccount ( ) , bundlerConfig :

{ // Get the ZeroDev API from https://dashboard.zerodev.app/ url :

https://rpc.zerodev.app/api/v2/bundler/ { projectId } , } , paymasterConfig :

{ // Get the ZeroDev API from https://dashboard.zerodev.app/ url :

https://rpc.zerodev.app/api/v2/paymaster/ { projectId } , } , } , }) ;

ERC-20 Paymasterâ

When using an ERC-20 paymaster, ensure you include the approval transaction, as Web3Auth does not handle the approval internall.

For Pimlico, you can specify the token you want to use in the paymasterContext. If you want to set up sponsorship policies, you can define those in the paymasterContext as well.Checkout the supported tokens for ERC-20 Paymaster on Pimlico.

import

{ AccountAbstractionProvider , SafeSmartAccount , }

from

"@web3auth/account-abstraction-provider" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/ { chainId } /rpc?apikey= { pimlicoAPIKey } , paymasterContext :

{ token :

"SUPPORTED_TOKEN_CONTRACT_ADDRESS" , } , } , smartAccountInit :

new

SafeSmartAccount ( ) , paymasterConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/ { chainId } /rpc?apikey= { pimlicoAPIKey } , } , } , }) ;

Set upâ

Once you have configured your AccountAbstractionProvider, you can now plug it in your Web3Auth Modal instance. If you are using the external wallets like MetaMask, Coinbase, etc, you can define whether you want to use the AccountAbstractionProvider, or EthereumPrivateKeyProvider by setting theuseAAWithExternalWallet inWeb3AuthOptions .

If you are settinguseAAWithExternalWallet totrue , it'll create a new Smart Account for your user, where the signer/creator of the Smart Account would be the external wallet.

If you are settinguseAAWithExternalWallet tofalse , it'll skip creating a new Smart Account, and directly use the external wallet to sign the transactions.

Configure Web3Auth Instanceâ

import

{ EthereumPrivateKeyProvider }

from

"@web3auth/ethereum-provider" ; import

{ AccountAbstractionProvider , SafeSmartAccount }

from

"@web3auth/account-abstraction-provider" ; import

{ Web3Auth }

from

"@web3auth/modal" ;

const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0xaa36a7" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , displayName :

"Ethereum Sepolia Testnet" , blockExplorerUrl :

"https://sepolia.etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const accountAbstractionProvider =

new

AccountAbstractionProvider ({ config :

{ chainConfig , bundlerConfig :

{ // Get the pimlico API Key from dashboard.pimlico.io url :

https://api.pimlico.io/v2/11155111/rpc?apikey= { pimlicoAPIKey } , } , smartAccountInit :

new

SafeSmartAccount ( ) , } }) ;

const privateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig } , }) ;

const web3auth =

new

Web3Auth ({ clientId :

"YOUR_WEB3AUTH_CLIENT_ID" , web3AuthNetwork :

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider , accountAbstractionProvider // This will allow you to use EthereumPrivateKeyProvider for // external wallets, while use the AccountAbstractionProvider // for Web3Auth embedded wallets. useAAWithExternalWallet :

false , }) ;

Configure Signerâ

The Web3Auth AccountAbstractionProvider is compatible with popular signer SDKs, including ethers, web3, and viem. You can choose your preferred package to configure the signer.

You can retreive the provider to configure the signer from Web3Auth instance.

- Viem
- Ethers
- Web3

import

{ createWalletClient }

from

"viem" ;

// Use your Web3Auth instance to retreive the provider. const provider = web3auth . provider ;

const walletClient =

createWalletClient ({ transport :

custom (provider) , }) ; import

{ ethers }

from

"ethers" ;

// Use your Web3Auth instance to retreive the provider. const provider = web3auth . provider ;

const ethersProvider =

new

ethers . providers . Web3Provider (provider) ; const signer =

await ethersProvider . getSigner ( ) ; import

Web3

from

"web3" ;

// Use your Web3Auth instance to retreive the provider. const provider = web3auth . provider ;

const web3 =

new

Web3 (provider) ;

Smart Account Addressâ

Once, you have set up the signer, you can use it to retreive the user's smart account address.

- Viem
- Ethers
- Web3

// Use walletClient instance from previous step const addresses =

await walletClient . getAddresses ( ) ;

const smartAccountAddress = addresses [0] ; const eoaAddress = addresses [1] ; // User signer from previous step const smartAccountAddress = signer . getAddress ( ) ; // Use web3 instance from previous step const addresses =

await web3 . eth . getAccounts ( ) ;

const smartAccountAddress = addresses [0] ; const eoaAddress = addresses [1] ;

Send Transactionâ

You can use your signer to submit on-chain transactions, while the Account Abstraction Provider handles the creation and submission of the UserOperation. You only need to passto ,data , andvalue ; other parameters will be discarded and overridden by the provider.

The bundler client determinesmaxFeePerGas andmaxPriorityFeePerGas to prevent transaction rejections. If you'd like to specify custom values, you can useAccountAbstractionProvider.bundlerClient to create and send the transaction.

Smart Accounts are basically smart contract deployed on chain. Therefore, the user's first transaction also deploys the smart contract for their wallet.

- Viem
- Ethers
- Web3

// Convert 1 ether to WEI format const amount = web3 . utils . toWei (1) ;

// Submits a user operation to the blockchain const receipt =

await web3 . eth . sendTransaction ({ to :

"DESTINATION_ADDRESS" , value : amount , // This will perform the transfer of ETH data :

"0x" , }) ; // Convert 1 ether to WEI format const amount = ethers . parseEther ("1.0") ;

// Submits a user operation to the blockchain const transaction =

await signer . sendTransaction ({ to :

"DESTINATION_ADDRESS" , value : amount , // This will perform the transfer of ETH data :

"0x" , }) ;

// Wait for the transaction to be mined const receipt =

await transaction . wait ( ) ; // Convert 1 ether to WEI format const amount =

parseEther ("1") ;

// Submits a user operation to the blockchain const hash =

await walletClient . sendTransaction ({ to :

"DESTINATION_ADDRESS" , value : amount , // This will perform the transfer of ETH data :

"0x" , }) ;

// Wait for the transaction to be mined const receipt =

await publicClient . waitForTransactionReceipt ({ hash }) ;

Advanced Smart Account Operationsâ

To learn more about supported transaction methods, and how to perform batch transactionscheckout our detailed documentation of AccountAbstractionProvider . Edit this page Previous React Hooks Next Whitelabel