Integration

Smart Accounts are created deterministically using the CREATE2 opcode, meaning the smart account address can be identified even before it is deployed. It will also maintain the same address across different chains.

To integrate a Biconomy Smart Account , you'll need the bundler url . If you also want to use the paymaster, you'll need Paymaster api key .

Installation

First, install the following packages for initializing the Smart Account.

- npm
- yarn
- pnpm

```
npm install @biconomy/account yarn add @biconomy/account pnpm add @biconomy/account
```

Integration

Specify PRIVATE_KEY , BUNDLER_URL , and RPC_URL (for ethers) to create a smart account.

- Viem
- Ethers
- Alchemy

```
import

{ Hex , createWalletClient , http }

from

"viem" ; import

{ privateKeyToAccount }

from

"viem/accounts" ; import

{ polygonMumbai }

from

"viem/chains" ; import

{ createSmartAccountClient }

from

"@biconomy/account" ;

// ----- 1. Generate EOA from private key const account =

privateKeyToAccount ( "0x"

+

"PRIVATE_KEY" ) ; const client =

createWalletClient ( { account , chain : polygonMumbai , transport :

http ( ) , } ) ; const eoa = client . account . address ; console . log (EOA address: { eoa } ) ;

// ------ 2. Create biconomy smart account instance const smartAccount =

await

createSmartAccountClient ( { signer : client , bundlerUrl :

"" ,
// <-- Read about this at https://docs.biconomy.io/dashboard#bundler-url biconomyPaymasterApiKey :
"" ,
// <-- Read about at https://docs.biconomy.io/dashboard/paymaster } ) ; const smartAccountAddress =
await smartAccount . getAccountAddress ( ) ; console . log ( "Smart Account Address" , smartAccountAddress ) ; import
{ ethers }
from
"ethers" ; import
{ createSmartAccountClient }
from
"@biconomy/account" ;
let provider =
new
ethers . JsonRpcProvider ( "RPC_URL" ) ; let signer =
new
ethers . Wallet ( "PRIVATE_KEY" , provider ) ;
const biconomySmartAccountConfig =
{ signer : signer , bundlerUrl :
"" ,
// <-- Read about this at https://docs.biconomy.io/dashboard#bundler-url biconomyPaymasterApiKey :
"" ,
// <-- Read about at https://docs.biconomy.io/dashboard/paymaster } ;
const smartAccount =
await
createSmartAccountClient ( biconomySmartAccountConfig ) ;
const smartAccountAddress =
await smartAccount . getAccountAddress ( ) ; console . log ( "Smart Account Address" , smartAccountAddress ) ; import
{ ethers }
from
"ethers" ; import
{ polygonMumbai }
from
"viem/chains" ; import
{ createSmartAccountClient }
from
"@biconomy/account" ; import
{ WalletClientSigner }
from
```

"@alchemy/aa-core" ;

const account =

privateKeyToAccount ("0x"

+

"PRIVATE_KEY") ; const client =

createWalletClient ({ account , chain : polygonMumbai , transport :

http ( ) , }) ; const alchemySigner =

new

WalletClientSigner (client ,

"json-rpc") ; const smartAccount =

await

createSmartAccountClient ({ signer : alchemySigner , bundlerUrl : config . bundlerUrl , biconomyPaymasterApiKey : config . biconomyPaymasterApiKey , }) ; const saAddress =

await smartAccount . getAccountAddress ( ) ; console . log ("SA Address" , saAddress) ; In this doc, the following values are used by default:

- ECDSA Validation Module. Other validationmodules
- can be enabled based on the specific use case.
- The default entry pointaddress
- is used for the chainId.
- An index with a value 0, but you can also specify a custom index value in the configuration to generate multiple smart accounts with the same signer.

CREATE2 allows us to enable flows where users could even send funds to an undeployed wallet, which could then be used to pay for the gas for deployment. A smart account will be deployed with the first UserOp execution. Edit this page Previous Smart Accounts Next Methods