# Send a simple transaction

## Overview

This tutorial demonstrates how to send a simple transaction using ethers.js/viem and the Biconomy Smart Account with the@biconomy/account SDK.

## Prerequisites

- Node.js installed on your machine
- A Bundler url if you don't want to use the testnet one (for mumbai you can usehttps://bundler.biconomy.io/api/v2/80001/nJPK7B3ru.dd7f7861-190d-41bd-af80-6877f74b8f44
- )
- An rpc url (for mumbai can usehttps://rpc.ankr.com/polygon_mumbai
- )
- An address to send the transaction to (replace0xaddress
- )

## Step 1: Generate the config and Create Biconomy Smart Account

- viem
- ethers

```
import

{ createWalletClient }

from

"viem" ; import

{ privateKeyToAccount }

from

"viem/accounts" ; import

{ polygonMumbai }

from

"viem/chains" ; import

{ createSmartAccountClient }

from

"@biconomy/account" ;

// Your configuration with private key and Biconomy API key const config =

{ privateKey :

"your-private-key" , bundlerUrl :

"" ,

// <-- Read about this at https://docs.biconomy.io/dashboard#bundler-url } ;

// Generate EOA from private key using ethers.js const account =

privateKeyToAccount ( "0x"

+ config . privateKey ) ; const signer =

createWalletClient ( { account , chain : polygonMumbai , transport :

http ( ) , } ) ;

// Create Biconomy Smart Account instance const smartWallet =
```

await

createSmartAccountClient ( { signer , bundlerUrl : config . bundlerUrl , } ) ;

const saAddress =

await smartWallet . getAccountAddress ( ) ; console . log ( "SA Address" , saAddress ) ; import

{ ethers }

from

"ethers" ; import

{ createSmartAccountClient }

from

"@biconomy/account" ;

// Your configuration with private key and Biconomy API key const config =

{ privateKey :

"your-private-key" , bundlerUrl :

"" ,

// <-- Read about this at https://docs.biconomy.io/dashboard#bundler-url rpcUrl :

"rpc-url" , } ;

// Generate EOA from private key using ethers.js let provider =

new

ethers . providers . JsonRpcProvider ( config . rpcUrl ) ; let signer =

new

ethers . Wallet ( config . privateKey , provider ) ;

// Create Biconomy Smart Account instance const smartWallet =

await

createSmartAccountClient ( { signer , bundlerUrl : config . bundlerUrl , } ) ;

const saAddress =

await smartWallet . getAccountAddress ( ) ; console . log ( "SA Address" , saAddress ) ; Get your signer from either ethers.js or viem and create a Biconomy Smart Account instance.

## Step 2: Generate Transaction Data

const toAddress =

"0xaddress" ;

// Replace with the recipient's address const transactionData =

"0x123" ;

// Replace with the actual transaction data

// Build the transaction const tx =

{ to : toAddress , data : transactionData , } ; Specify the recipient's address and transaction data to build the simple transaction.

## Step 3: Send the Transaction and wait for the Transaction Hash

// Send the transaction and get the transaction hash const userOpResponse =

await smartWallet . sendTransaction ( tx ) ; const

{ transactionHash }

=

await userOpResponse . waitForTxHash ( ) ; console . log ( "Transaction Hash" , transactionHash ) ;

const userOpReceipt =

await userOpResponse . wait ( ) ; if ( userOpReceipt . success ==

'true' )

{ console . log ( "UserOp receipt" , userOpReceipt ) console . log ( "Transaction receipt" , userOpReceipt . receipt ) } Send the transaction using the Biconomy Smart Account and get the transaction hash. The transaction will be built into a User Operation and then sent to the Bundler.

That's it! You've successfully sent a simple transaction using ethers.js/viem and the Biconomy Smart Account. Feel free to customize this example based on your specific use case. Previous Tutorials Next Send a batch of transactions