USDC quick start guide

USDC provides the ability to transfer dollars over the Arbitrum network using a smart contract. The smart contract enables users to send, receive, and store dollars on-chain with a wallet.

This guide will walk you through using theviem framework to build a simple app that enables a user to connect their wallet and interact with the blockchain by sending a USDC transaction from their address.

Prerequisites

Before you start building the sample app to perform a USDC transfer, ensure you meet the following prerequisites:

1. Node.js and npm
2. : Ensure that you have Node.js and npm installed on your machine. You can download and install Node.js fromnodejs.org
3. . npm comes with Node.js.
4. MetaMask
5. : Install the MetaMask browser extension and set up your wallet. Ensure that your wallet is funded with:
6.
   - Some native gas tokens (e.g., ETH on the Sepolia network) to cover transaction fees.
7.
   - USDC tokens for the transfer. (USDC Testnet Faucet
8.
   -)
9. Project Setup
10. : Create a new project directory and initialize it with npm:

mkdir usdc-transfer-app cd usdc-transfer-app npm init -y 1. Dependencies 2. : Install the required dependencies using the following command:

npm

install react@^18.2.0 react-dom@^18.2.0 @types/react@^18.0.27 @types/react-dom@^18.0.10 @vitejs/plugin-react@^3.1.0 typescript@^5.0.3 vite@^4.4.5 This will set up your development environment with the necessary libraries and tools for building a React application with TypeScript and Vite.

Installation

To install viem run the following command.

npm i viem

Setup public client

The public client is used to interact with your desired blockchain network.

import

{ http , createPublicClient }

from

'viem' ; import

{ arbitrumSepolia }

from

'viem/chains' ;

const publicClient =

createPublicClient ({ chain : arbitrumSepolia , transport :

http ( ) , }) ;

Setup wallet client

The wallet client is used to interact with Arbitrum accounts to retrieve accounts, execute transactions, and sign messages.

```
import
{ createWalletClient }
from
'viem' ; import
{ arbitrumSepolia }
from
'viem/chains' ;
const walletClient =
createWalletClient ( { chain : arbitrumSepolia , transport :
custom ( window . ethereum ! ) , } ) ;
```

Define USDC contract details

Define the USDC contract address and ABI (Application Binary Interface). The ABI specifies the functions available in the contract. (The USDC Token Contract Address referenced in the code is on Ethereum Sepolia)

```
const
USDC_CONTRACT_ADDRESS
=
'0x75faf114eafb1BDbe2F0316DF893fd58CE46AA4d' ; const
USDC_ABI
=
[ { constant :
false , inputs :
[ {
name :
'_to' ,
type :
'address'
} , {
name :
'_value' ,
type :
'uint256'
} , ] , name :
'transfer' , outputs :
[ {
name :
'' ,
```

type :

'bool'

}] , type :

'function' , } ,] ;

Connect wallet

Create a function to connect the user's wallet and retrieve their account address.

const

connect

=

async

( )

=>

{ const

[address]

=

await walletClient . requestAddresses ( ) ; setAccount (address) ; } ;

Send transaction

Create a function to send the USDC transfer transaction. This function encodes the transfer function data and sends the transaction using the wallet client.

const data =

encodeFunctionData ({ abi :

USDC_ABI , functionName :

'transfer' , args :

[to , valueInWei] , }) ;

const hash =

await walletClient . sendTransaction ({ account , to :

USDC_CONTRACT_ADDRESS , data , }) ;

Wait for transaction receipt

Use the public client to wait for the transaction receipt, which confirms that the transaction has been mined.

useEffect (( )

=>

{ (async

( )

=>

{ if

(hash)

```
{ const receipt =

await publicClient . waitForTransactionReceipt ( { hash } ) ; setReceipt ( receipt ) ; } } ) ( ) ; } ,

[ hash ] ) ;
```

Final step: build your USDC transfer sample app

Now that you understand the core components for programmatically performing your first USDC transaction, create the followingindex.tsx andindex.html files to build a sample app. This app will enable you to send USDC from one wallet to another. Ensure that your wallet is funded with both the native gas token and USDC.

```
index.tsx import

React ,

{ useEffect , useState }

from

'react' ; import

ReactDOM

from

'react-dom/client' ; import

{ http , type Address , type Hash , type TransactionReceipt , createPublicClient , createWalletClient , custom , stringify ,
encodeFunctionData , }

from

'viem' ; import

{ arbitrumSepolia }

from

'viem/chains' ; import

'viem/window' ;

const publicClient =

createPublicClient ( { chain : arbitrumSepolia , transport :

http ( ) } ) ;

const walletClient =

createWalletClient ( { chain : arbitrumSepolia , transport :

custom ( window . ethereum ! ) } ) ;

const

USDC_CONTRACT_ADDRESS

=

'0x75faf114eafb1BDbe2F0316DF893fd58CE46AA4d' ; const

USDC_ABI

=

[ { constant :

false , inputs :

[ {
```

```
name :
'_to' ,
type :
'address'
} , {
name :
'_value' ,
type :
'uint256'
} , ] , name :
'transfer' , outputs :
[ {
name :
'' ,
type :
'bool'
} ] , type :
'function' , } , ] ;
function
Example ( )
{ const
[ account , setAccount ]
= useState < Address
      ( ) ; const
[ hash , setHash ]
= useState < Hash
      ( ) ; const
[ receipt , setReceipt ]
= useState < TransactionReceipt
      ( ) ;
const addressInput =
React . createRef < HTMLInputElement
      ( ) ; const valueInput =
React . createRef < HTMLInputElement
      ( ) ;
const
connect
```

```
=
async
( )
=>
{ const
[ address ]
=
await walletClient . requestAddresses ( ) ; setAccount ( address ) ; } ;
const
sendTransaction
=
async
( )
=>
{ if
( ! account )
return ; const to = addressInput . current ! . value
as
Address ; const value = valueInput . current ! . value
as
{ number } ; const valueInWei =
BigInt ( value )
*
BigInt ( 10
**
6 ) ;
// Assuming USDC has 6 decimals
const data =
encodeFunctionData ( { abi :
USDC_ABI , functionName :
'transfer' , args :
[ to , valueInWei ] , } ) ;
const hash =
await walletClient . sendTransaction ( { account , to :
USDC_CONTRACT_ADDRESS , data , } ) ; setHash ( hash ) ; } ;
useEffect ( ( )
=>
{ ( async
( )
=>
{ if
( hash )
{ const receipt =
await publicClient . waitForTransactionReceipt ( { hash } ) ; setReceipt ( receipt ) ; } } ) ( ) ; } ,
[ hash ] ) ;
if
( account )
{ return
( <
        < div
        Connected :
{ account } < / div
        < input ref = { addressInput } placeholder = "address"
/
        < input ref = { valueInput } placeholder = "value (USDC)"
/
        < button onClick = { sendTransaction }
        Send < / button
        { receipt &&
( < div
        Receipt :
< pre
        < code
        { stringify ( receipt ,
null ,
2 ) } < / code
        < / pre
        < / div
        ) } < /
        ) ; } return
< button onClick = { connect }
        Connect
Wallet < / button
        ; }
```

ReactDOM . createRoot (document . getElementById ('root')

as

HTMLElement) . render (< Example

/

) ; index.html

< html

lang

" en "

< head

< meta

charset

" UTF-8 "

/> < meta

name

" viewport "

content

" width=device-width, initial-scale=1.0 "

/> < title

USDC Transfer Sample App </ title

</ head

< body

< h1

USDC Transfer Sample App </ h1

< div

id

" root "

</ div

< script

type

" module "

src

" /index.tsx "

```
</ script

    </ body

    </ html
```

By combining theseindex.tsx andindex.html files, you will have a complete setup that allows you to perform a USDC transfer from your wallet. Simply connect your wallet, input the recipient's address and the amount of USDC to transfer, and click the “Send” button to execute the transaction. You will receive a transaction receipt once the transaction is confirmed on the blockchain. Edit this page Last updatedonJan 27, 2025

Previous Contribute third-party docs Next Covalent