Quickstart: Transfer 10 USDC on Solana

Use the USDC token contract and third-party tools to transfer 10 USDC from one blockchain wallet to another.Suggest Edits

This step-by-step guide teaches you how to write a script to send 10 USDC between two wallets using the Solana Devnet, which is Solana's testing network.

To better understand the process and its components, you will use third-party tooling such as Javascript libraries, node services, and wallets. Completing this tutorial will help prepare you to integrate USDC payment flows within your app. Building with USDC enables you to move money near-instantly around the globe using decentralized protocols.

Note: Use Testnet Scripts as Development Framework

- As with most of our quickstarts, all API calls and transactions in this guide take place within the testnet environment; no real-world funds will be transferred.
- The sample code in this tutorial can be adapted for real-world transactions and can serve as a framework for your further development.

Tip: Consider the EVM-Compatible Alternative Tutorial

You can go through this process using an EVM compatible chain with the on-chain tutorial here .

USDC and Its Token Contract

USDC is a stablecoin backed 1:1 to the U.S dollar and operates natively on many public blockchains . Like many currencies, USDC is powered by a token contract, a programmable piece of code that manages user balances autonomously across a decentralized network.

As transactions occur, the token contract automatically updates the digital ledger, ensuring real-time tracking of funds. This mechanism allows individuals and businesses to send and receive dollars seamlessly, capitalizing on the transparency, security, and efficiency of blockchain technology.

Prerequisites

Before you begin this tutorial, you will need the following apps, tools, and accounts:

1. Circle Mint Sandbox Account
2. : an account that provides access to testnet USDC.
3. Phantom
4. :
5. An app for creating wallets, which are devices for storing USDC on chain:
6.
    1. Create two wallets, one for sending and one for receiving testnet USDC transactions on the Solana Devnet network. These will serve as your origin wallet, from which one you send USDC, and the destination wallet, which receives USDC. Keep note of the addresses for both.
7.
    1. Make sure you securely record the private key for the origin wallet.
8.
    1. Import the Solana Devnet USDC token contract
9.
    1. into your origin wallet to enable it to read the token balance. Note the contract address:
10.
    1. 4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU

Note: Access to the Testnet Scripts

Refer to our testnet scripts here . 1. Node.js 2. : JavaScript runtime environment to run code locally on your machine. 3. Web3.js 4. : JavaScript library containing functions to call the USDC token contract and initiate transactions. 5. Code Editor: A source code editor of your choice. One example, Visual Studio 6. , is available as a desktop application.

Tutorial

Follow the steps in this tutorial to transfer USDC on Solana:

Set up a Circle Mint Sandbox Account to access the USDC testnet.

1. Navigate to Circle'ssignup page
2. , enter your information in the required fields, andclick

3. "Sign Up." Completing the signup process takes you to the sandbox account Home:

4. Navigate to the APIs tab in the sidebar, andclick

5. the “CREATE A KEY” button:
6.
    1. Complete
7.
    1. the “Key Name” field.
8.
    1. Click
9.
    1. the “CREATE KEY” button. A modal title “API Key Created” will populate upon success.
10.
    1. Click
11.
    1. “SHOW” to view the value of your new key.
12.
    1. Youmust
13.
    1. click
14.
    1. the key to record a copy of its value.

Tip: Do not alter the API Key string:

You must use the entire value of your key, including the first part of the string, which is “SAND_API_KEY:”.

Gather Testnet USDC and SOL

You will need to use native gas tokens to pay the transaction (gas) fee:

1. Configure the Circle sample sandbox application by adding your API key:
2.
    1. Navigate
3.
    1. tosample-sandbox.circle.com
4.
    1. .
5.
    1. Click
6.
    1. on the settings (gear) menu in the top right-hand corner.
7.
    1. Complete
8.
    1. the field for "Your API Key" with the value from your sandbox account.
9.
    1. Click
10.
    1. on the hamburger button in the top left corner to access API methods.
11. Mint testnet USDC with a simulated bank transfer using methods in theCore Functionality
12. menu:
13.
    1. Click
14.
    1. the

15.
    1. POST /businessAccount/banks/wires
16.
    1. menu item.
17.
    1.
        1. Click
18.
    1.
        1. the
19.
    1.
        1. PREFILL FORM
20.
    1.
        1. button and then choose
21.
    1.
        1. US BANK ACCOUNT
22.
    1.
        1. to add a simulated bank account for testing.
23.
    1.
        1. Click
24.
    1.
        1. the
25.
    1.
        1. MAKE API CALL
26.
    1.
        1. button.
27.
    1.
        1. Copy the value for the
28.
    1.
        1. id
29.
    1.
        1. key.
30.
    1. Click
31.
    1. the
32.
    1. GET /businessAccount/banks/wires/{id}/instructions
33.
    1. menu item.
34.
    1.
        1. Complete
35.
    1.
        1. the
36.
    1.
        1. Account Id
37.
    1.
        1. field using the value of the
38.
    1.
        1. id
39.
    1.

        1. key returned from the previous step.
40.
    1.
        1. Click
41.
    1.
        1. the
42.
    1.
        1. MAKE API CALL
43.
    1.
        1. button.
44.
    1.
        1. From the successful response, record the value of the
45.
    1.
        1. trackingRef
46.
    1.
        1. and
47.
    1.
        1. accountNumber
48.
    1.
        1. fields.
49.
    1. Click
50.
    1. the
51.
    1. POST /mocks/payments/wire
52.
    1. menu item.
53.
    1.
        1. Complete
54.
    1.
        1. the
55.
    1.
        1. Tracking Ref
56.
    1.
        1. and
57.
    1.
        1. Account Number
58.
    1.
        1. fields with the values you recorded from the previous step.
59.
    1.
        1. Complete
60.
    1.
        1. the
61.
    1.
        1. Amount
62.
    1.
        1. field with a dollar amount you choose for your mock payment.
63.
    1.

        1. Click
64.
    1.
        1. the
65.
    1.
        1. MAKE API CALL
66.
    1.
        1. button.
67. Add the SOL wallet address to your address book:
68.
    1. You can use a REST client to make a POST request to the address recipients endpoint. The sandbox app does not yet support
69.
    1. SOL
70.
    1. blockchain:cURL
71.
    1. curl -X POST \
72.
    1. --url https://api-sandbox.circle.com/v1/addressBook/recipients \
73.
    1. --header 'accept: application/json' \
74.
    1. --header "Authorization: Bearer " \
75.
    1. --header 'content-type: application/json' \
76.
    1. --data '{"idempotencyKey": "", "address": "", "chain": "SOL", "metadata": {"email": "", "bns": "", "nickname": "" } }'
77.
    1. Record the value of the
78.
    1. id
79.
    1. from the successful response.
80. Click
81. the
82. POST /payouts
83. menu item.
84.
    1. Complete
85.
    1. the
86.
    1. Amount
87.
    1. field with a dollar amount you choose for your mock payment.
88.
    1. Complete
89.
    1. the
90.
    1. id
91.
    1. field with the values you recorded from the previous step.
92.
    1. Click
93.
    1. the
94.
    1. MAKE API CALL
95.
    1. button.
96. Request testnet SOL:
97.
    1. In a new browser tab or window,navigate
98.

1. to the Sol Faucet athttps://solfaucet.com/

99.

1. .

100.

1. Complete

101.

1. the

102.

1. Enter Solana account address

103.

1. field with your Fantom deposit address from step 2.

104.

1. Below the wallet address input field,select

105.

1. the amount of SOL tokens you want.

106.

1. Select

107.

1. the Devnet environment. Your SOL should automatically be air dropped into your wallet upon success.

108. Make payouts in USDC from thePayouts API
109. menu:
110.

1. Fromsample-sandbox.circle.com

111.

1. ,click

112.

1. the Payounts APIs menu item.

113.

1. Click

114.

1. the

115.

1. POST /payouts

116.

1. menu item.

117.

1. Complete

118.

1. the

119.

1. Amount

120.

1. field.

121.

1. Set

122.

1. the

123.

1. Currency

124.

1. to

125.

1. USD

126.

1. .

127.

1. Complete

128.

1. the

129.

1. Destination

130.

1. field with the recipient ID that corresponds to your destination wallet address.

131.

1. Select

132.

1. address_book

133.

    1. from the
134.
    1. Destination Type
135.
    1. drop-down.
136.
    1. Complete
137.
    1. the
138.
    1. Beneficiary Email
139.
    1. with your recipient email address.
140.
    1. Click
141.
    1. the
142.
    1. MAKE API CALL
143.
    1. button.
144.
    1. Record
145.
    1. the value of the
146.
    1. id
147.
    1. for your payout.
148. Monitor your payouts from thePayouts API
149. menu:
150.
    1. From[sample-sandbox.circle.com](sample-sandbox.circle.com)
151.
    1. ,click
152.
    1. the Payounts APIs menu item.
153.
    1. Click
154.
    1. the
155.
    1. GET /payouts/{id}
156.
    1. menu item.
157.
    1. Complete
158.
    1. the
159.
    1. Payout Id
160.
    1. field with the value of your payout ID from the previous step.
161.
    1. Click
162.
    1. the
163.
    1. MAKE API CALL
164.
    1. button to check the status of your payout.

Install Web3.js library to enable API calls

Install Web3.js by running the command below at the command line.

javasc // Type this command to install libraries npm install @solana/web3.js @solana/spl-token bs58

Customize sample JavaScript code by inputting variables

In your text editor, create a new JavaScript file called send10.js . Review the commented code below to understand its structure. Copy and paste it into your new file:

JavaScript // Write JavaScript Code const bs58 = require('bs58'); const { Connection, Keypair, PublicKey, clusterApiUrl, } = require("@solana/web3.js");

const { getOrCreateAssociatedTokenAccount, transfer } = require("@solana/spl-token"); // Replace these with your own keys and desired transfer amount const PRIVATE_KEY = ""; // Your private key in Base58 encoding const RECEIVER_PUBLIC_KEY = ""; // Receiver's public key const TRANSFER_AMOUNT = 10000000; // 10 amount of USDC to transfer (in smallest unit) // The address of the USDC token on Solana Devnet const USDC_DEV_PUBLIC_KEY = "4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU"; // Convert the private key from Base58 to a byte array and create a Keypair const senderPrivateKeyBytes = bs58.decode(PRIVATE_KEY); const senderKeypair = Keypair.fromSecretKey(senderPrivateKeyBytes); // Create a new connection to the Solana Devnet const connection = new Connection(clusterApiUrl("devnet"), "confirmed"); (async () => { try { // Fetch the sender's USDC token account const senderTokenAccount = await getOrCreateAssociatedTokenAccount(connection, senderKeypair, new PublicKey(USDC_DEV_PUBLIC_KEY), senderKeypair.publicKey,); // Fetch or create the receiver's associated token account for USDC const receiverTokenAccount = await getOrCreateAssociatedTokenAccount(connection, senderKeypair, new PublicKey(USDC_DEV_PUBLIC_KEY), new PublicKey(RECEIVER_PUBLIC_KEY), true, // Allow creating a token account for the receiver if it doesn't exist); // Perform the transfer const signature = await transfer(connection, senderKeypair, senderTokenAccount.address, receiverTokenAccount.address, senderKeypair.publicKey, TRANSFER_AMOUNT,); // Log the transaction signature console.log(Transaction signature: {signature}); console.log(You can verify the transaction on https://explorer.solana.com/tx{signature}?cluster=devnet); } catch (error) { console.error("Error performing the transfer:", error); } })();

Modify the Code

Replace the value of the PRIVATE_KEY variable with your secure private key, which must be in Base58 format. Replace the value of the RECEIVER_PUBLIC_KEY variable with the receiver’s public key.

Note: Transfer amount format

The set transfer amount value of 10000000 is equal to 10 USDC. This setting is because ISDC uses a 6 decimal place format, and the amount value should be in the smallest unit. Save the Javascript file.

Execute JavaScript code to initiate USDC transfer

From the terminal command line, enter the following command to execute the code and transfer USDC:

// Type node filename.js to run code node filename.js You should see a similar output to the following code if your file executes successfully:

```
node send10.js
```

Transaction signature: dsLFw3rtDEWT2b3vpzournXspeTbeEZvzVAh3bmNaLzrqQB7nWrgj4rdh4j9gVb51p4mkDm1ivFUgkuC6Q7aehT You can verify the transaction on https://explorer.solana. Check the status of your USDC transaction by visiting theSolana Devnet block explorer .

Conclusion

By completing this quickstart guide for USDC, you have started to use the essential skills of programmable money. You can adapt the script provided in this tutorial to enable real-world transactions and empower your own app development.

Now, you can begin to apply this knowledge to create frictionless financial experiences to empower users with seamless, secure global transactions. Updated3 months ago