Send an Outbound Transfer

Learn how to send USDC from a user-controlled wallet you've already created.Suggest Edits

This guide outlines initiating a currency transfer from a previously created user-controlled wallet. If you have not yet created a user-controlled wallet, go to this guide . If you do not have any tokens in your wallet, go to theinbound transfer guide.

The following steps utilize Circle's sample applications in combination with API requests that can be done via Circle's API references or cURL requests. cURL request will be provided inline, while API references will be linked from the API endpoint code text. You can find instructions on using it in thetesting via the reference pages guide.

1. Run Sample App

Once you have one of the web, iOS, or Androidsample applications set up locally, you will then:

1. Run the sample app and simulator.
2. Obtain your App ID. This can be done by one of two options1. Access the developer console and navigate to theconfigurator
3. 1. within user-controlled wallets. From there, copy the App ID.
4. 1. Make an API request toGET /config/entity
5. 1. and copy the App ID from the response body.
6. Add the App ID to the sample app.

2. Acquire a Session Token

You will start by making a request toPOST /users/token using a previously createduserId . The userToken is a 60-minute session token to initiate requests requiring a user challenge (PIN code entry). After 60 minutes, the session expires, and a new userToken must be generated via the same endpoint.

From this response, you will acquire the encryptionKey and userToken which you should provide in the respective sample app fields. Additionally, you will use the userToken in Step 3.

Node.js cURL // Import and configure the user-controlled wallet SDK const { initiateUserControlledWalletsClient } = require('@circle-fin/user-controlled-wallets'); const circleUserSdk = initiateUserControlledWalletsClient({ apiKey: '' });

const response = await circleUserSdk.createUserToken({ userId: '2f1dcb5e-312a-4b15-8240-abeffc0e3463' }); curl --request POST \ --url 'https://api.circle.com/v1/w3s/users/token' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --data ' { "userId": "2f1dcb5e-312a-4b15-8240-abeffc0e3463" } ' Response Body { "data": { "userToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCC9.eyJkZXZlbG9wZXJFbnRpdHlFbnZpcm9ubWVudCI6IlRFU1QiLCJlbnRpdHlJZCI6IjRlMDdhOGM5LTIxOTAtNDVlNC1hNjc0LWQyMGFkNjg4MWI3YyIsImV4cCjUR8i4zMmfdURw3FFcQldSbm-BUg6M7FP_fp-cs9xBbNmRZa31gMd1aKdcajJ9SvlVrfUowYfGXM3VcNF8rtTFtW-gk1-KzU4u10U35XXbbMcW1moxE0Rqx_fKotDgk2VdlTuuds5d5TiQzAXECqeCOCtNoDKktMkgIltbnLxOaRl2ReZjGt-ctD2V0DbYNO4T_ndPSUDI6qD7dXQRed5uDcezJYoha3Qj3tFGBglEnox2Y6DWTbllqjwmfTGrU8Pr0yz4jQz7suGwmiCzHPxcpYxMzYQ", "encryptionKey": "Tlcyxz7Ts9ztRLQq5+pic0MIETblYimOo2d7idV/UFM=" } }

3. Check the Wallet Balance and Acquire the Token ID

Before making an outbound transfer, you must gather the token's ID and ensure you are holding a token balance. To do this, make a request toGET /wallets passing in the wallets userToken to get the walletId .

Node.js cURL const response = await circleUserSdk.listWallets({ userToken: '', pageSize: 10 }); curl --request GET \ --url 'https://api.circle.com/v1/w3s/wallets?pageSize=10' \ --header 'accept: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' Response Body { "data": { "wallets": [{ "id": "01899cf2-d415-7052-a207-f9862157e546", "state": "LIVE", "walletSetId": "01899cf2-d407-7f89-b4d9-84d63573f138", "custodyType": "ENDUSER", "userId": "2f1dcb5e-312a-4b15-8240-abeffc0e3463", "address": "0x075e62c80e55d024cfd8fd4e3d1184834461db57", "addressIndex": 0, "blockchain": "MATIC-MUMBAI", "accountType": "SCA", "updateDate": "2023-07-28T14:41:47Z", "createDate": "2023-07-28T14:41:47Z" }] } } Next, you will make a request toGET /wallet/{id}/balances to check the balance of tokens and acquire the tokenId you intend to transfer. The token ID will be used in the following steps.

Node.js cURL const response = await circleUserSdk.getWalletTokenBalance({ userToken: '' }); curl --request GET \ --url 'https://api.circle.com/v1/w3s/wallets/{id}/balances' \ --header 'accept: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' Response Body { "data": { "tokenBalances": [{ "token": { "id": "38f2ad29-a77b-5a44-be05-8d03923878a2", "blockchain": "MATIC-MUMBAI", "tokenAddress": "0x0fa8781a83e46826621b3bc094ea2a0212e71b23", "standard": "ERC20", "name": "USD Coin (PoS)", "symbol": "USDC", "decimals": 6, "isNative": false, "updateDate": "2023-06-29T06:41:32Z", "createDate": "2023-06-29T06:41:32Z" }, "amount": "10", "updateDate": "2023-10-11T20:13:33Z" }] } }

4. Estimate the Cost of Transferring the token (Optional)

To estimate the fees for the transaction to transfer tokens, make a request toPOST transactions/transfer/estimateFee .

Node.js cURL const response = await circleUserSdk.estimateTransferFee({ userToken: '', amount: ['.01'], destinationAddress: '0xEb9614D6d001391e22dDbbEA7571e9823A469c1f', tokenId: '38f2ad29-a77b-5a44-be05-8d03923878a2', walletId: '01899cf2-d415-7052-a207-f9862157e546' }); curl --request POST \ --url 'https://api.circle.com/v1/w3s/transactions/transfer/estimateFee' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' \ --data ' { "amounts": ["."01"], "destinationAddress": "0xEb9614D6d001391e22dDbbEA7571e9823A469c1f", "tokenId": "38f2ad29-a77b-5a44-be05-8d03923878a2", "walletId": "01899cf2-d415-7052-a207-f9862157e546" } Response Body { "data": { "low": { "gasLimit": "21000", "baseFee": "2.456220277", "priorityFee": "1.022783914", "maxFee": "5.935224468" }, "medium": { "gasLimit": "21000", "baseFee": "2.456220277", "priorityFee": "2.655282857", "maxFee": "7.567723411" }, "high": { "gasLimit": "21000", "baseFee": "2.456220277", "priorityFee": "15.986229693", "maxFee": "20.898670247" } } }

5. Initiate a Blockchain Transfer

Make a request toPOST /user/transactions/transfer to initiate a blockchain transfer from a specified walletId to a blockchain address destinationAddress . This call returns a challengeId , used within the sample app, that prompts users to enter their PIN code to authorize the transfer.

If you do not have a wallet to use as a destination for the transfer, you can create another User-Controlled Wallet by stepping throughcreate your first wallet or send funds to any other blockchain wallet such asMetamask . Node.js cURL const response = await circleUserSdk.createTransaction({ userToken: '', amounts: ['.01'], destinationAddress: '0x6E5eAf34c73D1CD0be4e24f923b97CF38e10d1f3', tokenId: '38f2ad29-a77b-5a44-be05-8d03923878a2', walletId: '01899cf2-d415-7052-a207-f9862157e546', fee: { type: 'level', config: { feeLevel: 'MEDIUM' } } }); curl --request POST \ --url 'https://api.circle.com/v1/w3s/user/transactions/transfer' \ --header 'Content-Type: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' \ --data ' { "userId": "2f1dcb5e-312a-4b15-8240-abeffc0e3463", "idempotencyKey": "607a0972-17f9-4d56-8ca3-a0e94adc3210", "amounts": ["."01"], "destinationAddress": "0x6E5eAf34c73D1CD0be4e24f923b97CF38e10d1f3", "tokenId": "38f2ad29-a77b-5a44-be05-8d03923878a2", "walletId": "01899cf2-d415-7052-a207-f9862157e546", "feeLevel": "MEDIUM" }' Response Body { "data": { "challengeId": "0d1b5f41-1381-50af-983b-f54691415158" } }

6. Authorize transfer from the sample app

Using the sample application, enter the userToken and secretKey returned from Step 2. Also, enter the challengeId returned from Step 5.

At this point, you should be ready to execute your first transfer through the sample app. ClickExecute in the sample app to continue.

The sample application takes you through the authentication process, which includes the user entering their PIN code to authorize the transfer.

7. Check the Transfer Status

As the transfer state changes and ultimately completes, Circle sends notifications to asubscribed endpoint . You can find a list of all possible states in theAsynchronous States and Statuses guide . The Webhook notification will be similar to the one below.

Webhook Request Body { "subscriptionId": "d4c07d5f-f05f-4fe4-853d-4dd434806dfb", "notificationId": "acab8c14-92ae-481a-8335-6eb5271da014", "notificationType": "transactions.outbound", "notification": { "id": "ad3f40ae-9c0e-52cf-816f-91838850572a", "blockchain": "MATIC-MUMBAI", "tokenId": "38f2ad29-a77b-5a44-be05-8d03923878a2", "walletId": "01899cf2-d415-7052-a207-f9862157e546", "sourceAddress": "0x7b777eb80e82f73f118378b15509cb48cd2c2ac3", "destinationAddress": "0x6e5eaf34c73d1cd0be4e24f923b97cf38e10d1f3", "transactionType": "OUTBOUND", "custodyType": "ENDUSER", "state": "COMPLETE", "amounts": ["0.01"], "nfts": null, "txHash": "0x535ff240984f54e755d67cdc9c79c88768fe5997955f09f3a66b4d1126810900", "blockHash": "0xa4c5c79500240f3ae3f4e5c5f641198b7c698d83b7539ac4e8cf2d3f5f49bdfd", "blockHeight": 41100000, "networkFee": "0.07037500047405219", "firstConfirmDate": "2023-10-11T21:08:28Z", "operation": "TRANSFER", "userId": "c266945c-f440-4537-85cf-a16b6e33b0cc", "abiParameters": null, "createDate": "2023-10-11T21:08:13Z", "updateDate": "2023-10-11T21:08:37Z" }, "timestamp": "2023-10-11T21:08:13Z", "version": 2 } Alternatively, you can pollGET /transactions using the userId or userToken associated with your user.

Node.js cURL const response = await circleUserSdk.listTransactions({ userToken: '' }); curl --request GET \ --url 'https://api.circle.com/v1/w3s/transactions' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' Response Body { "data": { "transactions": [{ "id": "ad3f40ae-9c0e-52cf-816f-91838850572a", "blockchain":

"MATIC-MUMBAI", "tokenId": "38f2ad29-a77b-5a44-be05-8d03923878a2", "walletId": "01899cf2-d415-7052-a207-f9862157e546", "sourceAddress": "0x7b777eb80e82f73f118378b15509cb48cd2c2ac3", "destinationAddress": "0x6e5eaf34c73d1cd0be4e24f923b97cf38e10d1f3", "transactionType": "OUTBOUND", "custodyType": "ENDUSER", "state": "COMPLETE", "amounts": ["0.01"], "nfts": null, "txHash": "0x535ff240984f54e755d67cdc9c79c88768fe5997955f09f3a66b4d1126810900", "blockHash": "0xa4c5c79500240f3ae3f4e5c5f641198b7c698d83b7539ac4e8cf2d3f5f49bdfd", "blockHeight": 41100000, "networkFee": "0.07037500047405219", "firstConfirmDate": "2023-10-11T21:08:28Z", "operation": "TRANSFER", "userId": "c266945c-f440-4537-85cf-a16b6e33b0cc", "abiParameters": null, "createDate": "2023-10-11T21:08:13Z", "updateDate": "2023-10-11T21:08:37Z" }, { "id": "81cf790a-ed95-5d41-b7bd-c4e15390eef6", "blockchain": "MATIC-MUMBAI", "tokenId": "38f2ad29-a77b-5a44-be05-8d03923878a2", "walletId": "01899cf2-d415-7052-a207-f9862157e546", "sourceAddress": "0x48520ff9b32d8b5bf87abf789ea7b3c394c95ebe", "destinationAddress": "0x7b777eb80e82f73f118378b15509cb48cd2c2ac3", "transactionType": "INBOUND", "custodyType": "ENDUSER", "state": "COMPLETE", "amounts": ["10"], "nfts": null, "txHash": "0x5121f9efec29d4d661ffb0b777727d1f5ba7b5bc286ac4891c82f7b1b80a9485", "blockHash": "0xba7984dbe7423827b5fd175a636552ae85401c3f2a0c5cdda934a37d6652ac49", "blockHeight": 41098635, "networkFee": "0.001911870000955935", "firstConfirmDate": "2023-10-11T20:13:33Z", "operation": "TRANSFER", "userId": "c266945c-f440-4537-85cf-a16b6e33b0cc", "abiParameters": null, "createDate": "2023-10-11T20:13:33Z", "updateDate": "2023-10-11T20:13:45Z" }] } } Updated16 days ago

What's Next Congratulations! You've received your first transaction to your user-controlled wallet. To learn how wallet recovery works, go to Reset Account Pin Code * Table of Contents * * 1. Run Sample App * * 2. Acquire a Session Token * * 3. Check the Wallet Balance and Acquire the Token ID* * 4. Estimate the Cost of Transferring the token (Optional)* * 5. Initiate a Blockchain Transfer * * 6. Authorize transfer from the sample app * * * 7. Check the Transfer Status