Create Your First User-Controlled Wallet

Use API requests and Circle's sample app to create a user-controlled wallet.Suggest Edits

This guide outlines initializing and creating a user-controlled smart contract account (SCA) wallet by setting their PIN code and security questions. It utilizes Circle's sample application 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 are linked from the API endpoint text. Instructions on using the API references are in thisguide .

If you prefer to create an externally owned account (EOA) wallet, take note of the callout where an API parameter must be altered. To learn more about the two account types, please see theaccount types guide .

Prerequisites

1. Create adeveloper account and acquire an API key in the console
2. .
3. Install theWeb3 Services SDKs
4. , which is currently only available for Node.js. (optional
5.)
6. Set up one of the web, iOS, or Androidsample applications
7. locally.

1. Configure and Run the 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.

App ID

AKA Application ID is a unique identifier assigned to your application. It serves as a key that allows you to configure and manage various settings specific to your User-Controlled Wallet integration. The App ID is essential for identifying your application and enabling communication with the Circle Platform APIs. View of the sample app on launch

2. Create a User

Make a request toPOST /users to create a userId . This represents the user's account and all associated wallets, assets, and transactions. The userId is recommended to be in the UUID format.

We recommend that you maintain a mapping to associate the end-user profile usernames with the userId provided to our service/end-point. You can use a local database to maintain this mapping. 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.createUser({ userId: '2f1dcb5e-312a-4b15-8240-abeffc0e3463' }); curl --request POST \ --url 'https://api.circle.com/v1/w3s/users' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --data ' { "userId": "2f1dcb5e-312a-4b15-8240-abeffc0e3463" } ' If the request is successful, you will receive an empty response body.

Response Body {}

3. Acquire a Session Token

Next, you will need to acquire a session token. To do this, you will make a request to thePOST /users/token using the previously created userId in Step 2. The userToken is a 60-minute session token, which is used to initiate requests that require a userchallenge (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 4.

Node.js cURL 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_fKotDgk2VdITuuds5d5TiQzAXECqeCOCtNoDKktMkglltbnLxOaRl2ReZjGt-ctD2V0DbYNO4T_ndPSUDI6qD7dXQRed5uDcezJYoha3Qj3tFGBglEnox2Y6DWTbllqjwmfTGrU8Pr0yz4jQz7suGwmiCzHPxcpYxMzYQ", "encryptionKey": "Tlcyxz7Ts9ztRLQq5+pic0MIETblYimOo2d7idV/UFM=" } }

4. Initialize the User's Account and Acquire the Challenge ID

You have two options to initialize your user's account:

For this guide, we will use option one to create a user and a wallet simultaneously.

1 POST /user/initialize : Initialize a user account and create a wallet This call generates wallets for the specified blockchains at the time of account creation. Use this method if you know which blockchain the wallet will be created on. 2 POST /user/pin : Initialize the user account This call generates an account without creating a wallet. Use this method if you are unsure when creating an account on which blockchain the wallet will be created on. Make a request toPOST /user/initialize using the userToken returned from Step 3. This call returns aChallenge ID, which is used with the Circle Programmable Wallet SDK to have the user set their PIN code and security questions.

Make sure to provide a Testnet blockchain such as ETH-SEPOLIA , MATIC-MUMBAI , and AVAX-FUJI .

Externally Owned Account (EOA)

If you prefer to create an EOA, change the account type parameter to "accountType": "EOA" . If an account type is not provided an EOA wallet will be created. Node.js cURL const response = await circleUserSdk.createUserPinWithWallets({ userToken: '', accountType: 'SCA', blockchains: ['MATIC-MUMBAI'] }); curl --request POST \ --url 'https://api.circle.com/v1/w3s/user/initialize' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' \ --data ' { "idempotencyKey": "49e3f455-60a2-4b5e-9e9e-9400b86e5f34", "accountType": "SCA", "blockchains": ["MATIC-MUMBAI"] } ' Response Body { "data": { "challengeId": "0d1b5f41-1381-50af-983b-f54691415158" } }

5. Create a Wallet in the Sample App

At this point, you should be ready to execute your first request through the sample app. Once you've entered the required fields indicated in Step 4, clickExecute to continue.

The sample application takes you through the end user initialization process, which includes setting up the user's PIN code and security questions and having the user confirm their configuration.

6. Check User Status

Once you have completed all the steps in the sample app, you can then check the user status by making a request toGET /user providing the userToken to retrieve the status of the user's account.

To understand the current state of the user, inspect the following values:

1. PIN Status
2. : This parameter indicates whether the end-user has successfully set a 6-digit PIN. If the user has set the PIN successfully, the
3. pinStatus
4. value will be
5. enabled
6. .
7. Security Question Status
8. : This parameter provides information about the user's recovery method status, specifically related to the defined security questions. If the end-user has successfully established a recovery method by defining their security questions, the
9. securityQuestionStatus
10. will be set to
11. enabled
12. .

Additional information provided will include the number of failed attempts for both the pinStatus and the security questions. If the end-user enters an incorrect PIN or security answers more than three times, the pin entry or recovery method will be locked, and they will need to wait 30 minutes for it to be unlocked. Node.js cURL const response = await circleUserSdk.getUserStatus({ userToken: '' }); curl --request GET \ --url 'https://api.circle.com/v1/w3s/user' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --header 'X-User-Token: ' Response Body { "data": { "id": "2f1dcb5e-312a-4b15-8240-abeffc0e3463", "status": "ENABLED", "createDate": "2023-07-26T15:27:32Z", "pinStatus": "ENABLED", "pinDetails": { "failedAttempts": 0 }, "securityQuestionStatus": "ENABLED", "securityQuestionDetails": { "failedAttempts": 0 } } }

7. Check Wallet Status

Additionally, you can make an API request toGET /wallets using the userToken to see the user's newly created wallets.

Node.js cURL const response = await circleUserSdk.listWallets({ userToken: '' }); curl --request GET \ --url 'https://api.circle.com/v1/w3s/wallets' \ --header 'accept: application/json' \ --header 'content-type: 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" }] } } Updated16 days ago

What's Next Congratulations! You've successfully set up your first user-controlled wallet. To learn how to send tokens to the wallet see the next guide. Receive an Inbound Transfer * Table of Contents * * Prerequisites * * 1. Configure and Run the Sample App* * 2. Create a User * * 3. Acquire a Session Token* * 4. Initialize the User's Account and Acquire the Challenge ID* * 5. Create a Wallet in the Sample App * * 6. Check User Status * * 7. Check Wallet Status