Python SDK

Set up the server-side Python SDK in your application.Suggest Edits

Use the Python SDK to interact with Circle Web3 Services APIs, such as embedding secure wallets in your applications or interacting with smart contracts on the Web3 Services platform.

Prerequisites

- Own an activeCircle Developer Account
- .
- Generate an API Key
- to use in requests to Circle APIs.
- Generate an entity secret
- to use in requests with the Smart Contract Platform SDK and Developer-Controlled Programmable Wallets SDK.

Available SDKs

The Python SDK includes the following application-specific SDKs:

- User-controlled Wallets SDK
- Developer-controlled Wallets SDK
- Smart Contract Platform SDK

For details, see theProgrammable Wallets andSmart Contract Platform guides.

Install SDKs

Usepip to install all SDKs.

Unset pip install circle-developer-controlled-wallets pip install circle-smart-contract-platform pip install circle-user-controlled-wallets

Configure clients

To use an SDK, you must configure a client.

User-controlled programmable wallets client

The following code sample shows how to import the client and configure it to use your API key:

Python from circle.web3 import utils

client = utils.init_user_controlled_wallets_client(api_key="Your API KEY") The following code sample shows how to create a transaction using the client:

Python import uuid from circle.web3 import user_controlled_wallets

generate a user id

user_id = str(uuid.uuid4())

create user

api_instance = user_controlled_wallets.UsersAndPinsApi(client) try: request = user_controlled_wallets.CreateUserRequest(user_id=user_id) api_instance.create_user(request) except user_controlled_wallets.ApiException as e: print("Exception when calling UsersAndPinsApi->create_user: %s\n" % e)

get user

try: response = api_instance.get_user(id=user_id) print(response.data) except user_controlled_wallets.ApiException as e: print("Exception when calling UsersAndPinsApi->get_user: %s\n" % e)

get user token

try: request = user_controlled_wallets.GenerateUserTokenRequest.from_dict({"userId": user_id}) response = api_instance.get_user_token(request) print(response.data.user_token) except user_controlled_wallets.ApiException as e: print("Exception when calling UsersAndPinsApi->get_user_token: %s\n" % e)

Developer-controlled programmable wallets client

The following code example shows how to import the client and configure it to use your API key and entity secret:

Python from circle.web3 import utils

client = utils.init_developer_controlled_wallets_client(api_key="Your API KEY", entity_secret="Your entity secret") The following code sample demonstrates how to create a transaction using the client:

Python from circle.web3 import developer_controlled_wallets

api_instance = developer_controlled_wallets.WalletSetsApi(client)

create wallet sets

try: request = developer_controlled_wallets.CreateDeveloperWalletSetRequest.from_dict({ "name": "my_wallet_set" }) api_instance.create_wallet_set(request) except developer_controlled_wallets.ApiException as e: print("Exception when calling WalletSetsApi->create_wallet_set: %s\n" % e)

list wallet sets

try: response = api_instance.list_wallet_sets() for wallet_set in response.data.wallet_sets: print(wallet_set.id) except developer_controlled_wallets.ApiException as e: print("Exception when calling WalletSetsApi->list_wallet_sets: %s\n" % e)

Smart Contract Platform client

The following code samples demonstrate how to import the client and configure it to use your API key and entity secret:

Python from circle.web3 import utils

client = utils.init_smart_contract_platform_client(api_key="Your API KEY", entity_secret="Your entity secret") The following code sample shows how to use the client to create a smart contract:

Python from circle.web3 import smart_contract_platform

api_instance = smart_contract_platform.DeployImportApi(client)

import contract

try: request = smart_contract_platform.ImportContractRequest.from_dict({ "name": "UChildERC20Proxy", "address": "0x2791Bca1f2de4661ED88A30C99A7a9449Aa84174", "blockchain": "MATIC" }) response = api_instance.import_contract(request) print(response) except smart_contract_platform.ApiException as e: print("Exception when calling DeployImportApi->import_contract: %s\n" % e)

Client configuration options

Each SDK client accepts the following configuration parameters:

Parameter Required? Description api_key Yes API Key used to authenticate with Circle APIs. entity_secret Yes Your configured entity secret.

Required for Developer-controlled Programmable Wallets and Smart Contract Platform clients to auto-generate entitySecretCiphertext . host No Base URL that overrides the default URL of https://api.circle.com/v1/w3s . user_agent No Custom user agent request header. If provided, it is prepended to the default user agent header. Updated14 days ago * Table of Contents * * Prerequisites * * Available SDKs * * Install SDKs * * Configure clients * * * User-controlled programmable wallets client * * * Developer-controlled programmable wallets client * * * Smart Contract Platform client * * * Client configuration options