Circle SDKs

Use our SDKs to quickly get your codebase up and running with Circle APIsSuggest Edits

Circle's SDKs are developer toolkits that enable you to integrate your code with Circle APIs. Because our SDKs are open source, users like you can help us to improve them. If you find bugs or missing features, we encourage you to file an issue or contribute back to thenode-sdk .

Note: This page contains code snippets and examples that can be run as is or incorporated into your apps.

Circle offers SDKs for Typescript, Java and Python. Java and Python SDKs are currently in beta.

Installing the SDK

You can run one of the two commands below from your project directory to install the SDK.

Node Java (Beta) Python (Beta) npm install @circle-fin/circle-sdk --save

or

yarn add @circle-fin/circle-sdk //Add the following dependency to your pom.xml. Make sure to specify the version of the artifactcom.circle circle replace-me-with-latest-artifact-ver python3 -m pip install circle-sdk

Configuration

TypeScript Java (Beta) Python (Beta) import { Circle, CircleEnvironments } from "@circle-fin/circle-sdk";

const circle = new Circle("", CircleEnvironments.sandbox // API base url); import com.circle.client.Circle;

Circle.getInstance() .setBasePath(Circle.SANDBOX_BASE_URL) .setApiKey(""); import circle

Configure global settings

circle.base_url = circle.Environment.SANDBOX_BASE_URL circle.api_key = '

List balances

TypeScript Java (Beta) Python (Beta) async function listBalances() { const balancesRes = await circle.balances.listBalances(); } import com.circle.client.ApiException; import com.circle.client.api.BalancesApi; import com.circle.client.model.ListBalancesResponse;

// Create API driver BalancesApi api = new BalancesApi();

// Send request try { ListBalancesResponse resp = api.listBalances(); System.out.println(resp); } catch (ApiException e) { System.out.println("Failed to fetch balances"); System.out.println(e); } import circle from circle.apis.tags import balances_api

Choose API driver

api_instance = balances_api.BalancesApi()

Send request

try: api_response = api_instance.list_balances() pprint(api_response) except circle.ApiException as e: print("Exception when calling BalancesApi->list_balances: %s\n" % e)

Create a crypto payment

TypeScript Java (Beta) Python (Beta) async function createCryptoPayment() { const createCryptoPaymentRes = await circle.paymentIntents.createPaymentIntent({ idempotencyKey: "5c6e9b91-6563-47ec-8c6d-0ce1103c50b3", amount: { amount: "3.14", currency: "USD", }, settlementCurrency: "USD", paymentMethods: [{ chain: "ETH", type: "blockchain", },], }); } import com.circle.client.ApiException; import com.circle.client.Circle; import com.circle.client.api.CryptoPaymentIntentsApi; import com.circle.client.model.Chain; import com.circle.client.model.CreatePaymentIntentRequest; import com.circle.client.model.CreatePaymentIntentResponse; import com.circle.client.model.CryptoPaymentsMoney; import com.circle.client.model.PaymentIntentCreationRequest; import com.circle.client.model.PaymentMethodBlockchain;

import java.util.Arrays; import java.util.UUID;

// Create API driver CryptoPaymentIntentsApi api = new CryptoPaymentIntentsApi();

// Prepare request data CreatePaymentIntentRequest reqBody = new CreatePaymentIntentRequest(new PaymentIntentCreationRequest() .idempotencyKey(UUID.randomUUID()) .amount(new CryptoPaymentsMoney() .amount("45.19") .currency(CryptoPaymentsMoney.CurrencyEnum.USD)) .settlementCurrency(PaymentIntentCreationRequest.SettlementCurrencyEnum.USD) .paymentMethods(Arrays.asList(new PaymentMethodBlockchain() .chain(Chain.ETH) .type(PaymentMethodBlockchain.TypeEnum.BLOCKCHAIN))));

// Make API request and print response try { CreatePaymentIntentResponse resp = api.createPaymentIntent(reqBody); System.out.println(resp); // Print error if API request failed } catch (ApiException e) { System.out.println("Failed to make API call"); System.out.println(e); } from uuid import uuid4 import circle from circle.apis.tags import crypto_payment_intents_api from circle.model.create_payment_intent_request import CreatePaymentIntentRequest from circle.model.crypto_payments_money import CryptoPaymentsMoney from circle.model.currency import Currency from circle.model.payment_method_blockchain import PaymentMethodBlockchain from circle.model.chain import Chain

Configure global settings

circle.base_url = circle.Environment.SANDBOX_BASE_URL circle.api_key = '

Choose API driver

api_instance = crypto_payment_intents_api.CreatePaymentIntent()

body param

req_body = CreatePaymentIntentRequest(amount=CryptoPaymentsMoney(amount='35.14',currency=Currency.USD),idempotencyKey=str(uuid4()),settlementCurrency=Currency.USD,paymentMethods=[PaymentMethodBlockchain(chain=Chain.ETH,type="blockchain")])

try: api_response = api_instance.create_payment_intent(body=req_body) print(api_response) except circle.ApiException as ex: print("Exception when calling payment_intents_api->create_payment_intent: %s\n" % ex)

Get a crypto payment

TypeScript Java (Beta) Python (Beta) async function getCryptoPayment(id: string) { const cryptoPayment = await circle.paymentIntents.getPaymentIntent(id); } import com.circle.client.ApiException; import com.circle.client.api.CryptoPaymentIntentsApi; import com.circle.client.model.PaymentIntent;

CryptoPaymentIntentsApi api = new CryptoPaymentIntentsApi(); UUID paymentIntetntId = replace_with_payment_intent_id; // Send request try { PaymentIntent resp = api.getPaymentIntent(id); System.out.println(resp); } catch (ApiException e) { System.out.println("Failed to fetch payment intent"); System.out.println(e); } import circle from circle.apis.tags import payment_intents_api

api_instance = payment_intents_api.PaymentIntentsApi()

payment_intent_id = replace_with_payment_intent_id

Send request

try: api_response = api_instance.get_payment_intent(path_params={'id':payment_intent_id}) pprint(api_response) except circle.ApiException as e: print("Exception when calling PaymentIntentsApi->get_payment_intent: %s\n" % e) Updated4 months ago * Table of Contents * * Installing the SDK * * Configuration * * List balances * * Create a crypto payment * * Get a crypto payment