React Native SDK

Set up the client-side React Native SDK for user-controlled wallets in your mobile application.Suggest Edits

The Circle React Native SDK enables your mobile application to provide user-controlled programmable wallets. By integrating this SDK, your users can securely input sensitive data like PINs or security answers. The SDK encrypts the request body using a secret key, protecting your users' information.

The Web and Mobile SDKs preserve the user keyshare with the individual, giving them complete control. You must use the SDKs with the user-controlled wallet product. Additionally, the Web and Mobile SDKs support only the user-controlled wallet product. At Circle, we understand the importance of end-to-end security for your application and the need to create a tailored and seamless user experience for your end-users. Hence, Circle's SDK also exposes functionality for you to customize the description and layout of UI components we serve, including:

- UI Title and Subtitle Customization:
- Modify the title and subtitle to reflect your brand identity or provide specific instructions.
- Custom PIN Code Input Layout:
- Adjust the layout and styling of the PIN code input field to align with your application's design guidelines.
- Question List Configuration:
- Set the list of security questions displayed in the User Wallet UI, allowing users to choose from a predefined set.
- SDK Initialization:
- Initialize the Web SDK by setting the endpoint server, ensuring seamless communication between your application and Circle's services.
- Predefined Error Messages:
- Customize the error messages displayed to users, providing a more personalized experience and guidance.
- ChallengeID Acceptance and Operation Retrieval:
- Easily accept the
- challengeId
- and retrieve any relevant operations within the SDK.

Tip: See theReact Native SDK UI Customization API article to customize the SDK.

Install the SDK

You can set up the React Native SDK by downloading it fromnpm .

SDK architecture

You must use Web, iOS, Android or React Native SDKs to access the user-controlled Programmable Wallet product. The SDK secures, manages, and communicates with your server to ensure your user's keyshare, always stays with them and is not exposed to your servers.

To learn more, seeSDK Architecture for User-Controlled Wallets .

SDK API references

WalletSdk

implement IWallet

IWalletSdk

interface IWalletSdk

Properties sdkVersion: SdkVersion

SDK version. Methods init: (configuration: Configuration) => Promise

Configure the Circle endpoint for SDK to connect, reject Promise if the endpoint and appId's format are invalid. setSecurityQuestions: (securityQuestions: SecurityQuestion[]) => void

Set the security question list, throw Throwable when the SecurityQuestion array is empty or contains any question whose title length is not 2~512. addListener: (listener: EventListener) => void

Register an EventListener for the app to handle events, e.g. forgot PIN. removeAllListeners: () => void

Unregister all EventListener. execute: (userToken: string, encryptionKey: string, challengeIds: string[], successCallback:

SuccessCallback, errorCallback: ErrorCallback) => void

Execute the operations by challengeId and trigger the SuccessCallback with SuccessResult after sending the request to Circle endpoint successfully. ErrorCallback will be triggered when parameters are null or invalid. executeWithKeyShare: (userToken: string, encryptionKey: string, pinCodeDeviceShare: string, challengeIds: string[], successCallback: SuccessCallback, errorCallback: ErrorCallback) => void

Execute the operations by challengeId and PIN code device share and trigger the SuccessCallback with SuccessResult after sending the request to Circle endpoint successfully. ErrorCallback will be triggered when parameters are null or invalid. getDeviceId: () => string? setBiometricsPin: (userToken: string, encryptionKey: string, successCallback: SuccessCallback, errorCallback: ErrorCallback) => void

Setup BiometricsForPin and trigger the SuccessCallback with SuccessResult after finishing the operation successfully. ErrorCallback will be triggered when parameters are null or invalid. setDismissOnCallbackMap: (map: Map\) => void

The SDK UI can be dismissed if there's an error or warning during execute() and setBiometricsPin(). App can specify ErrorCode as true in the map to dismiss. moveTaskToFront: () => void

Bring the SDK UI to foreground. After calling moveRnTaskToFront() and finishing the flow on React Native UI, e.g. forgot PIN flow, apps can back to SDK UI by calling moveTaskToFront() and continue the SDK task. moveRnTaskToFront: () => void

Bring the React Native Activity / ViewController to front. In some cases, apps may need to go back to React Native UI during execute() or setBiometricsPin, e.g., receive an error, warning or event like forgotPin. setTextConfigsMap: (map: Map\) => void

Define strings with specific configurations for special text customization. All keys are listed in A Index Table. setIconTextConfigsMap: (map: Map\>) => void

Define icon and string sets with specific configurations for icon text list item customization. All keys are listed in B Index Table. setTextConfigMap: (map: Map\) => void

Define strings with specific configurations for general string customization. It will replace values. All keys are listed in C Index Table. setImageMap: (map: Map\) => void

Define image resource for image customization. All keys are listed in D Index Table. setDateFormat: (format: DateFormat) => void

Set display date format, e.g. the answer of a security question in which inputType is datePicker. All supported formats are listed in DateFormat. The default format is “YYYY-MM-DD”. setDebugging: (debugging: boolean) => void

Android only. true: default value, check customization map value and print warn level log. false: skip checking and turn off the log. setCustomUserAgent: (userAgent: string) => void

Set custom user agent value. setErrorStringMap: (map: Map\void

Define error message for customization.

SdkVersion

interface SdkVersion

Properties rn: string

Programmable Wallet React Native SDK version. native: string

Programmable Wallet Native SDK version.

Configuration

interface Configuration

Properties endpoint: string

The endpoint SDK connects. appId: string

Application ID, retrieved from Circle Web3 Services Console. settingsManagement?: SettingsManagement

Extra settings for SDK

SettingsManagement

interface SettingsManagement

Properties enableBiometricsPin: boolean

Enable biometrics to protect PIN code.

SecurityQuestion

class SecurityQuestion

Properties title: string

The question string. inputType: InputType

The input type of the question. Support text input and timePicker. Constructor constructor(title: string, inputType?: InputType)

Initialize a SecurityQuestion, use InputType.text as default value.

InputType

Enumerates the types of security questions.

JavaScript enum InputType { text = 'text', datePicker = 'datePicker', }

SuccessResult

interface SuccessResult

Properties result: ExecuteResult

Execute result. warning?: ExecuteWarning

Execute warning.

Error

interface Error

Properties code: string

Error code. message: ExecstringuteWarning

Error message.

SuccessCallback

type SuccessCallback

Type declaration (result: SuccessResult) => void

Callback when the operation is executed successfully.

ErrorCallback

type ErrorCallback

Type declaration (error: Error) => void

The callback is triggered when a failure occurs in operation or is canceled by the user.

ExecuteResult

interface ExecuteResult

Properties resultType: ExecuteResultType

The type of the operation that the challenge represents. status: ExecuteResultStatus

The status of the execution. data?: ExecuteResultData

The data of the execution.

ExecuteResultData

interface ExecuteResultData

Properties signature?: string

The signature for SIGN_MESSAGE and SIGN_TYPEDDATA.

ExecuteResultType

Enumerates the types of challenges supported

JavaScript enum ExecuteResultType { UNKNOWN = 'UNKNOWN', SET_PIN = 'SET_PIN', RESTORE_PIN = 'RESTORE_PIN', SET_SECURITY_QUESTIONS = 'SET_SECURITY_QUESTIONS', CREATE_WALLET = 'CREATE_WALLET', CREATE_TRANSACTION = 'CREATE_TRANSACTION', ACCELERATE_TRANSACTION = 'ACCELERATE_TRANSACTION', CANCEL_TRANSACTION = 'CANCEL_TRANSACTION', CONTRACT_EXECUTION = 'CONTRACT_EXECUTION', SIGN_MESSAGE = 'SIGN_MESSAGE', SIGN_TYPEDDATA = 'SIGN_TYPEDDATA', SET_BIOMETRICS_PIN = 'SET_BIOMETRICS_PIN', }

ExecuteResultStatus

Enumerates the possible statuses for a challenge

JavaScript enum ExecuteResultStatus { UNKNOWN = 'UNKNOWN', PENDING = 'PENDING', IN_PROGRESS = 'IN_PROGRESS', COMPLETE = 'COMPLETE', FAILED = 'FAILED', EXPIRED = 'EXPIRED', }

EventListener

type EventListener

Type declaration (event: ExecuteEvent) => void

Event listener for receiving ExecuteEvent.

ExecuteEvent

Enumerates the possible event

JavaScript enum ExecuteEvent { forgotPin = 'forgotPin', }

ErrorCode

Enumerates the types of error code

JavaScript enum ErrorCode { unknown = '-1', success = '0', apiParameterMissing = '1', apiParameterInvalid = '2', forbidden = '3', unauthorized = '4', retry = '9', customerSuspended = '10', pending = '11', invalidSession = '12', invalidPartnerId = '13', invalidMessage = '14', invalidPhone = '15', walletIdNotFound = '156001', tokenIdNotFound = '156002', transactionIdNotFound = '156003', walletSetIdNotFound = '156004', notEnoughFounds = '155201', notEnoughBalance = '155202', exceedWithdrawLimit = '155203', minimumFundsRequired = '155204', invalidTransactionFee = '155205', rejectedOnAmlScreening = '155206', tagRequired = '155207', gasLimitTooLow = '155208', transactionDataNotEncodedProperly = '155209', fullNodeReturnedError = '155210', walletSetupRequired = '155211', lowerThenMinimumAccountBalance = '155212', rejectedByBlockchain = '155213', droppedAsPartOfReorg = '155214', operationNotSupport = '155215', amountBelowMinimum = '155216', wrongNftTokenIdNumber = '155217', invalidDestinationAddress = '155218', tokenWalletChainMismatch = '155219', wrongAmountsNumber = '155220', userAlreadyExisted = '155101', userNotFound = '155102', userTokenNotFound = '155103', userTokenExpired = '155104', invalidUserToken = '155105', userWasInitialized = '155106', userHasSetPin = '155107', userHasSetSecurityQuestion = '155108', userWasDisabled = '155109', userDoesNotSetPinYet = '155110', userDoesNotSetSecurityQuestionYet = '155111', incorrectUserPin = '155112', incorrectDeviceId = '155113', incorrectAppId = '155114', incorrectSecurityAnswers = '155115', invalidChallengeId = '155116', invalidApproveContent = '155117', invalidEncryptionKey = '155118', userPinLocked = '155119', securityAnswersLocked = '155120', walletIsFrozen = '155501', maxWalletLimitReached = '155502', walletSetIdMutuallyExclusive = '155503', metadataUnmatched = '155504', userCanceled = '155701', launchUiFailed = '155702', pinCodeNotMatched = '155703', insecurePinCode = '155704', hintsMatchAnswers = '155705', networkError = '155706', biometricsSettingNotEnabled = '155708', deviceNotSupportBiometrics = '155709', biometricsKeyPermanentlyInvalidated = '155710', biometricsUserSkip = '155711', biometricsUserDisableForPin = '155712', biometricsUserLockout = '155713', biometricsUserLockoutPermanent = '155714', biometricsUserNotAllowPermission = '155715', biometricsInternalError = '155716', }

TextConfig

Data-only class for text customization. class TextConfig

Properties text?: string

Text to display. gradientColors?: string[]

Array of Gradient text color. Only used by TextsKey.enterPinCodeHeadline, TextsKey.securityIntroHeadline, TextsKey.newPinCodeHeadline textColor?: string

Text color. font?: string

Font. Constructor constructor(text?: string, gradientColorsOrTextColor: string[] | string, font?: string)

TextConfig

Data-only class for icon text list item customization. class IconTextConfig

Properties image: ImageSourcePropType

The image source for customization. textConfig: TextConfig

Text config for text customization. Constructor constructor(image: ImageSourcePropType, textConfig: TextConfig)

TextsKey

Enum for setTextConfigsMap(). SeeA Index Table .

IconTextsKey

Enum for setIconTextConfigsMap(). SeeB Index Table .

TextKey

Enum for setTextConfigMap(). SeeC Index Table .

ImageKey

Enum for setImageMap(). SeeD Index Table .

DateFormet

Enum for setDateFormat().

JavaScript enum DateFormat { YYYYMMDD_HYPHEN = 'yyyy-MM-dd', DDMMYYYY_SLASH = 'dd/MM/yyyy', MMDDYYYY_SLASH = 'MM/dd/yyyy', }

SampleCode

JavaScript import * as React from 'react'; import { WalletSdk } from '@circle-fin/w3s-pw-react-native-sdk';

export default function App() { const [endpoint, setEndpoint] = React.useState('https://api.circle.com/v1/w3s/'); const [appId, setAppId] = React.useState(''); const [enableBiometricsPin, setEnableBiometricsPin] = React.useState(false); const [userToken, setUserToken] = React.useState(''); const [encryptionKey, setEncryptionKey] = React.useState(''); const [challengeId, setChallengeId] = React.useState('');

```
React.useEffect(() => {
   // Regester EventListener
   WalletSdk.addListener((event: any) => {
  if(event == ExecuteEvent.forgotPin){
        WalletSdk.moveRnTaskToFront();
        // forgot PIN flow in React Native UI
      }
   });
_initSdk();
   _setSecurityQuestions();
   _setDismissOnCallbackMap();
   _setTextConfigsMap();
   _setIconTextConfigsMap();
   _setTextConfigMap();
   _setErrorStringMap();
```

```
  _setImageMap();
  _setDateFormat();
  _setDebugging();
  return () => {
    // Removes listeners once unmounted
    WalletSdk.removeAllListeners();
  };
```

}, []); ... return ({ _executeSdk(); }} > Execute { _setBiometricsPin(); }} > Set Biometrics Pin) } JavaScript ... const _initSdk = async () => { try { // Set endpoint, app ID and extra settings await WalletSdk.init({ endpoint, appId, settingsManagement: { enableBiometricsPin: enableBiometricsPin } }); } catch (e: any) { toast(e.message); return; } };

const _setSecurityQuestions = () => { WalletSdk.setSecurityQuestions([new SecurityQuestion('What was your childhood nickname?', InputType.text), new SecurityQuestion('When is your favorite date?', InputType.datePicker), new SecurityQuestion('What is the name of your first pet?'), new SecurityQuestion('What is your favorite country?'),]); };

const _setDismissOnCallbackMap = () => { const map = new Map(); map.set(ErrorCode.userCanceled, true); map.set(ErrorCode.networkError, true); WalletSdk.setDismissOnCallbackMap(map); };

const _setTextConfigsMap = () => { let gradientColors = ['#05184b', '#21bad5'] as Array; const map = new Map(); map.set(TextsKey.newPinCodeHeadline, [new TextConfig('ENTER your new ', '#a6183f'), new TextConfig('W3s PIN', undefined, gradientColors, 'Roboto'),]); map.set(TextsKey.securityIntroLink, [new TextConfig('==Learn more=='), new TextConfig('https://www.circle.com/en/legal/privacy-policy'),]); WalletSdk.setTextConfigsMap(map); };

const _setImageMap = () => { const imageMap = new Map(); imageMap.set(ImageKey.naviBack, require('../../assets/image/ic_back.png')); imageMap.set(ImageKey.naviClose, require('../../assets/image/ic_close.png')); imageMap.set(ImageKey.securityIntroMain, require('../../assets/image/grab_confirm_main_icon.png')); imageMap.set(ImageKey.selectCheckMark, { uri: 'https://drive.google.com/uc?id=1UTX1tFnECuj1k3U1bggH0g5YvH_Ens5M', }); imageMap.set(ImageKey.dropdownArrow, { uri: 'https://drive.google.com/uc?id=1PR3yYpk4AmsCAlUM8nw6C3y-RAXNjGMv', }); imageMap.set(ImageKey.errorInfo, { uri: 'https://drive.google.com/uc?id=1UJjinISU6ZHO0fAoKaZofBMgItaAn9kS', }); imageMap.set(ImageKey.securityConfirmMain, { uri: 'https://drive.google.com/uc?id=16OkP3VzEjLICOifNKRQi3FMJntV9F-n-', }); WalletSdk.setImageMap(imageMap); }; const _setIconTextConfigsMap = () => { const map = new Map>(); map.set(IconTextsKey.securityConfirmationItems, [new IconTextConfig(require('../../assets/image/ic_intro_item0_icon.png'), new TextConfig('This is the only way to recover my account access. ')), new IconTextConfig(require('../../assets/image/ic_intro_item1_icon.png'), new TextConfig('Circle won’t store my answers so it’s my responsibility to remember them.')), new IconTextConfig(require('../../assets/image/ic_intro_item2_icon.png'), new TextConfig('I will lose access to my wallet and my digital assets if I forget my answers. ')),]); WalletSdk.setIconTextConfigsMap(map); }; const _setTextConfigMap = () => {}; const _setErrorStringMap = () => {}; const _setDateFormat = () => {}; const _setDebugging = () => {};

// execute const _executeSdk = async () => { try { let { result } = await WalletSdk.execute(userToken, encryptionKey, [challengeId,]); console.log({result.resultType}, {result.status}, {result.data?.signature}); } catch (e: any) { console.log(e.message); } }; // setBiometricsPin const _setBiometricsPin = async () => { try { let { result } = await WalletSdk.setBiometricsPin(userToken, encryptionKey); console.log({result.resultType}, {result.status}); } catch (e: any) { console.log(e.message); } }; Updated5 days ago * Table of Contents * * Install the SDK * * SDK architecture * * SDK API references * * * WalletSdk * * * IWalletSdk * * * SdkVersion * * * Configuration * * * SettingsManagement * * * SecurityQuestion * * * InputType * * * SuccessResult * * * Error * * * SuccessCallback * * * ErrorCallback * * * ExecuteResult * * * ExecuteResultData * * * ExecuteResultType * * * ExecuteResultStatus * * * EventListener * * * ExecuteEvent * * * ErrorCode * * * TextConfig * * * TextConfig * * * TextsKey * * * IconTextsKey * * * TextKey * * * ImageKey * * * DateFormet * * * SampleCode