FinGo SDK

Android Developer Guide

Version 1.0 **4 May 2021**

Developed by KAN Developments

USER MANUAL
TABLE OF CONTENTS

Page #

1.0 GENERAL INFORMATION

1.1 System Overview

This is an Android SDK that provides third party apps to integrate with the Fingo device for the different available Fingo operations,

1. Enrollment
2. Identification
3. Payment
4. Refund

1.2 Authorized Use Permission

1.2.1 Technical Information

FinGo Technical Director	Mr. Andy Horton
Fingo CEO	Nick Dryden
KAN MD	Khaled Abdalla
KAN Technical	Gohary Ahmed
Release Date	4 May 2021

1.2.2 Confidentiality Level

High		**Medium**		**Low**		**Public**	
					X		

1.3 Points of Contact

For Inquiries please contact us on

1. **kabdalla@kan4u.com**

2.0 PRE-REQUISITE

1. Pre-Knowledge in Android Development
2. FingoSDK (.aar) file
3. Set of dependencies that needs to be included in the project's build.gradle file
    a. NDK Setup
        i. defaultConfig {
            minSdkVersion 23
            ……..

            ndk {
                abiFilters 'x64', 'armeabi'
            }
        }
    b. EventBus
        i. 'org.greenrobot:eventbus:3.2.0'
    c. Networking
        i. 'com.squareup.retrofit2:retrofit:2.9.0'
        ii. 'com.squareup.okhttp3:logging-interceptor:4.2.1'
    d. GSON
        i. 'com.squareup.retrofit2:converter-gson:2.4.0'

3.0 SDK FEATURES

3.1 Initialization

At startup of the app Fingo SDK needs to be initialized to be able to proceed with api usage.

3.1.1 Fingo SDK Init

1) Provide Context for initializing the FingoSDK as follows:

```
FingoErrorCode fingoInitErrorCode = FingoSDK.initialize(@context);
Log.i(TAG, fingoInitErrorCode);
```

2) After step 1 is finished error code is returned indicating the initialization status
   i. H1_OK if initialization is successful
   ii. Any other error code if initialization failed

3.1.2 Setting FingoParams

1) Create FingoParams object and specify the required parameters

```
FingoParams fingoParams = new FingoParams();
fingoParams.setCloudUrl("cloud_url");
fingoParams.setPartnerId("partner_id");
fingoParams.setMerchantId("merchant_id");
fingoParams.setTerminalId("terminalID");
fingoParams.setApiKey("api_key");
fingoParams.setTemplateKeySeed("template_key_seed");
```

2) Pass the specified Fingoparams to the fingo sdk during to set this params into the fingo SDK.

```
FingoErrorCode fingoParamsErrorCode = FingoSDK.setFingoParams(fingoParams);
```

3) After setting “FingoParams” into the SDK the sdk will return if the params r valid or not in the fingoParamsErrorCode.

3.2 Identification API Reference

Identification api’s includes (Enrollment and Identify).

3.2.1 Enrollment

1) User can enroll a finger using the finger’s vein ID on the fingo cloud, the cloud will provide a unique deterministic finger vein ID for the finger of choice.

2) During the enrollment process if the user’s finger is already enrolled on the cloud, the stored vein ID will be returned immediately in the api response.

3) Enrollment process consists of three finger enroll captures and one verification capture, so total of 4 scans for the enrollment process to be completed.

4) Same Finger should be used during the whole enrollment process.

3.2.2 Identify

1) After a successful enrollment on the fingo cloud, user can be identified by the enrolled finger unique biometric template (vein_id).

2) Identification process consists of one finger capture for the identification process to be completed.

3) If the user is already enrolled on the cloud, his unqiue vein ID will be returned from the cloud, if the user is not enrolled the cloud will return and error. (check the errors section).

3.3 Payment API Reference

Identification api's includes (Enrollment and Identify).

3.3.1 Payment

1) After a successful enrollment on the fingo cloud, user can perform payments on the fingo cloud, given that the user already have a predefined card data stored on the cloud..

2) Payment process consists of one finger capture for the payment process to be completed.

3) If the payment failed the cloud will return error (Refer to payment error section), if the payment is successful the below payment data will be returned from the cloud.

    a Transaction ID

    b Gateway Transaction ID

    c Gateway Auth Code

    d Masked Card Number

    e Merchant ID

    f Transaction Timestamp

3.3.2 Refund

1) After a successful payment on the fingo cloud, user can refund the already made transaction partial refund or full refund, given that the user already have a predefined card data stored and available payment to refund on the cloud.

2) Refund process consists of one finger capture for the refund process to be completed.

3) If the refund failed the cloud will return error (Refer to payment error section), if the refund is successful the below refund data will be returned from the cloud.

    a Transaction ID

    b Gateway Transaction ID

    c Gateway Auth Code

    d Masked Card Number

    e Merchant ID

    f Transaction Timestamp

3.4 Logger API

3.4.1 Usage

1. This API allow the third party app to register for fingo request and response logs at the initialization time of the SDK.
2. First third-party app needs to make use of the exposed interface of the lib FingoRequestLogger, either by implementing the interface or using it as a parameter
3. Once the FingoRequestLogger is setup you will need to initialize the FingoSDK using the logger

```
FingoSDK.initialize(@context, new FingoRequestLogger() {
    @Override
    public void onLogDataAvailable(String s) {
        Log.d(TAG, "onLogDataAvailable: " + s);
    }
});
```

4. Now the third-party app will receive callbacks whenever the SDK tries to go online and contact the Fingo cloud. (THIS API IS USED FOR DEBUG PURPOSES ONLY)

3.5 Examples

1) First create a FingoListener that will be responsible for the events received from the SDK api's and will provide callbacks with results.

```
FingoContract.FingoListener fingoListener = new FingoContract.FingoListener(){

@Override
public void onProcessingStarted() {
    Log.i(TAG, "onProcessingStarted");
}

@Override
public void onDisplayTextRequested(DisplayTextRequested displayTextRequested)
```

```
{
    Log.d(TAG, "onDisplayTextRequested: " + displayTextRequested.getType());
    Log.d(TAG, "onDisplayTextRequested: " + displayTextRequested.getText());
    Log.d(TAG, "onDisplayTextRequested: " + displayTextRequested.getCode());
}

@Override
public void onIdentifyData(IdentifyData identifyData) {
    Log.d(TAG, "onOnlineIdentifyData: " + identifyData.getMemberId());
    Log.d(TAG, "onOnlineIdentifyData: " + identifyData.getVeinId());
    Log.d(TAG, "onOnlineIdentifyData: " +
identifyData.getVerificationTemplate());
    Log.d(TAG, "onOnlineIdentifyData: " +
identifyData.getEnrolmentTemplate());
}

@Override
public void onPaymentData(PaymentData paymentData, FingoErrorResponse
fingoErrorResponse) {
    Log.d(TAG, "onOnlinePaymentData: " + paymentData.getTransactionId());
    Log.d(TAG, "onOnlinePaymentData: " + paymentData.getGatewayAuthCode());
    Log.d(TAG, "onOnlinePaymentData: " +
paymentData.getGatewayTransactionId());
    Log.d(TAG, "onOnlinePaymentData: " + paymentData.getMaskedCardNumber());
    Log.d(TAG, "onOnlinePaymentData: " + paymentData.getTimestamp());
    Log.d(TAG, "onOnlinePaymentData: " +
fingoErrorResponse.getFingoErrorList().get(0).getErrorCode());
    Log.d(TAG, "onOnlinePaymentData: " +
fingoErrorResponse.getFingoErrorList().get(0).getErrorMessage());
}

@Override
public void onProcessingFinished(ProcessingFinished processingFinished) {
    Log.d(TAG, "onProcessingFinished: " + processingFinished.getText());
    Log.d(TAG, "onProcessingFinished: " + processingFinished.getErrorName());
    Log.d(TAG, "onProcessingFinished: " + processingFinished.getErrorCode());
    Log.d(TAG, "onProcessingFinished: " + processingFinished.getStatus());
}
}
```

1) Create a presenter that will be responsible to access the SDK api's and will provide callbacks with results

```
fingoPresenter = new FingoPresenter(@activity, fingoListener);
```

2) To invoke the enrollment API

```
fingoPresenter.enroll(TIMEOUT);
```

3) To invoke the identify API

```
fingoPresenter.identify(TIMEOUT);
```

4) To invoke the payment API

```
PosData posData = new PosData("2", "SomeLocation");
fingoPresenter.payment(100, Currency.GBP, 0,posData, TIMEOUT);
```

5) To invoke the refund API

```
TerminalData terminalData = new TerminalData();
terminalData.setLocation("Cairo");
fingoPresenter.refund(100, "8c04ad1b-e1e8-4752-b50c-e3c9dc70ad11",
"96577222", terminalData, TIMEOUT);
```

6) To cancel the finger vein capture session

```
fingoPresenter.cancel();
```

7) To check if the capture session is cancelled by the SDK

```
fingoPresenter.isOperationCancelled()
```

8) For identification and enrollment API's, the results will be received in the

```
onIdentifyData(IdentifyData identifyData)
```

9) For payment and refund API's, the results will be received in the

```
```

```
onPaymentData(PaymentData paymentData, FingoErrorResponse
fingoErrorResponse)
```

4.0 SDK CODES

4.1 DisplayRequested Msg Code

The below error codes are for the onDisplayTextRequested callback, the callback returns a message in plain English and a unique code that represents this message.

Error Code	Error Message

2000	Please Insert your finger in the scanner
2001	Identifying finger vein please wait
2002	First finger vein scan is successful \n Please remove your finger
2003	Second finger vein scan is successful \n Please remove your finger
2004	Third finger vein scan is successful \n Please remove your finger
2005	Enrollment template generated successfully
2006	Finger vein identification successful
2007	Finger vein enrollment successful
2008	Payment Declined
2009	Payment Accepted
2010	Refund Declined
2011	Refund Accepted
2012	Operation Cancelled

4.2 Error Codes

4.2.1 Enroll & Identify

FVAE Error Codes

1.1 Return Code of FVAE

Return code (Dec)	Return code (Hex)	Return value	Cause and Solution
0	0x000	FV_CORE_OK	[Cause] ·Operation succeeded. [Solution] ·N/A
1	0x001	FV_CORE_ERR	[Cause] ·An internal error not specified in this table occurred. [Solution] ·Terminate and then reinitialize the FVAE module. ·Confirm that the installation is correctly done. ·Restart Windows.
31	0x01F	FV_CORE_ERR_INVALID_ARG	[Cause] ·An invalid argument is specified. [Solution] ·Confirm that input arguments to the function is correct. ·Check the log and identify the invalid argument.
32	0x020	FV_CORE_ERR_INVALID_HANDLE	[Cause] ·An invalid handle is specified. [Solution] ·Confirm that the specified handle is correct.

Return code (Dec)	Return code (Hex)	Return value	Cause and Solution
			·Check the log and identify the invalid handle.
111	0x06F	FV_CORE_ERR_UNABLE_TO_CAPTURE	[Cause] ·Failed to capture a FV image. [Solution] ·Check the log and identify where the error occurred. ·See (2)(3)(4) for detailed solutions
112	0x070	FV_CORE_ERR_UNABLE_TO_CREATE_TEMPLATE	[Cause] ·Failed to create a FV template. [Solution] ·Check the log and identify where the error occurred. ·See (2)(3)(4) for detailed solutions.
113	0x071	FV_CORE_ERR_UNABLE_TO_GET_TEMPLATE	[Cause] ·Failed to obtain a created FV template. [Solution] ·Check the log and identify where the error occurred. ·See (2)(3)(4) for detailed solutions.
114	0x072	FV_CORE_ERR_TIMEOUT	[Cause] ·Timeout occurred during the capture operation. [Solution] ·Confirm that the specified value of timeout is long enough. ·Confirm that zero is not specified to the value of timeout.
115	0x073	FV_CORE_ERR_CANCEL	[Cause] ·[OK] button is pressed by user while capturing a FV image. [Solution] ·N/A
161	0x0A1	FV_CORE_ERR_UNABLE_TO_INIT_FRAMEWORK	[Cause]

Return code (Dec)	Return code (Hex)	Return value	Cause and Solution
162	0x0A2	FV_CORE_ERR_UNABLE_TO_GET_FRAMEWORK_INFO	・Failed to initialize the Hitachi Secure BioAPI Runtime Libraries. [Solution] ・Check the log and identify where the error occurred. ・See (2)(3)(4) detailed solutions.
163	0x0A3	FV_CORE_ERR_UNABLE_TO_LOAD_BSP	
165	0x0A5	FV_CORE_ERR_UNABLE_TO_QUERY_UNITS	
166	0x0A6	FV_CORE_ERR_UNABLE_TO_ATTACH_BSP	
167	0x0A7	FV_CORE_ERR_ALREADY_INITIALIZED	[Cause] ・The FVAE module is already initialized. [Solution] ・Terminate and then reinitialize the FVAE module.
171	0x0AB	FV_CORE_ERR_UNABLE_TO_DETACH_BSP	[Cause] ・Failed to terminate the Hitachi Secure BioAPI Runtime Libraries. [Solution] ・Check the log and identify where the error occurred. ・See (2)(3)(4) for detailed solutions.
172	0x0AC	FV_CORE_ERR_UNABLE_TO_UNLOAD_BSP	
173	0x0AD	FV_CORE_ERR_UNABLE_TO_TERMINATE_FRAMEWORK	
231	0x0E7	FV_CORE_ERR_MORE_THAN_ONE_CANDIDATE	[Cause] ・Sequential identification failed. More than one candidate is found. [Solution] ・Retry the sequential identification match with the same finger, or try the second sequential identification match using a different finger.
232	0x0E8	FV_CORE_ERR_NOT_MATCHED	[Cause] ・Sequential identification failed. No candidate is found. [Solution] ・If genuine users keep getting rejected, see the chapter 3 "Application Note" of the *User Guide* for solutions.
261	0x105	FV_CORE_ERR_TOO_MANY_TEMPLATES	[Cause] ・Too many templates are added under the FVAE handle. [Solution]

Return code (Dec)	Return code (Hex)	Return value	Cause and Solution
			・Confirm that the number of templates added to the specified handle is not exceeding 100,000.
1011	0x3F3	FV_CORE_ERR_LICENSE_EXPIRED	[Cause] ・The license key for using the FVAE is expired. [Solution] ・Confirm that the license key is correctly specified in the function. ・Obtain a new license key.
1021	0x3FD	FV_CORE_ERR_OS_NOT_SUPPORTED	[Cause] ・The operating system is not supported by the FVAS. [Solution] ・Run the FVAS on a supported platform. See the *User Guide* for system requirements.

4.2.2 Payment & Refund

Error Codes

Code	Description
1208	The account does not have a default payment card set.
1221	Could not identify biometric template for the finger. Ensure that you have enrolled against the partner using the inserted finger.
1226	The specified merchant does not exist.
1227	The selected currency is currently not supported.
1229	No suitable payment gateway found, cannot complete transaction.
1230	Transaction couldn't get successfully authorised.
1231	Payment gateway took too long to respond.
1232	Account does not allow the selected payment gateway..
1241	Payment gateway could not verify the validity of the card.
9999	Bad input during capture, attempt enrolment again.
9998	Irrecoverable error. Follow FinGo Driver troubleshooting guidelines.

THE END OF THE DOCUMENT
THANKS