ANDROID SDK BBPOS DOCUMENTATION
PT CASHLEZ WORLDWIDE INDONESIA Tbk

[Cashlez External]

Approved by: Product Owner
Version: 2.0.3.12
Classification: External Use
Date: 24 Oct, 2022

DOCUMENT INFORMATION

Document Name : Android SDK BBPOS Document v2.0.3.12

Document Status : Final

Detail Status

Release Date	Oct 24, 2022

Document Version History

FSD Version No.	Date	Content	Modified By
2.0.3.10	11 May 2022	• Debit Transfer Force card chip (DIP) only	Julian Natalino
		• Finalization	Nathania Oey
2.0.3.11	31 Aug 2022	• Updating from 2.0.3.10	Julian Natalino
2.0.3.12	24 Oct 2022	• New Payment NOBU QRIS	Julian Natalino

Document Control

Role	Name	Division
Reviewed by	Nathania Oey	IT Compliance + TW Manager
Maintained by	Julian Natalino	Technical Writer
Document Owner	Ade Setiawan	Head of IT Compliance & Product

Table of Contents

1. Introduction

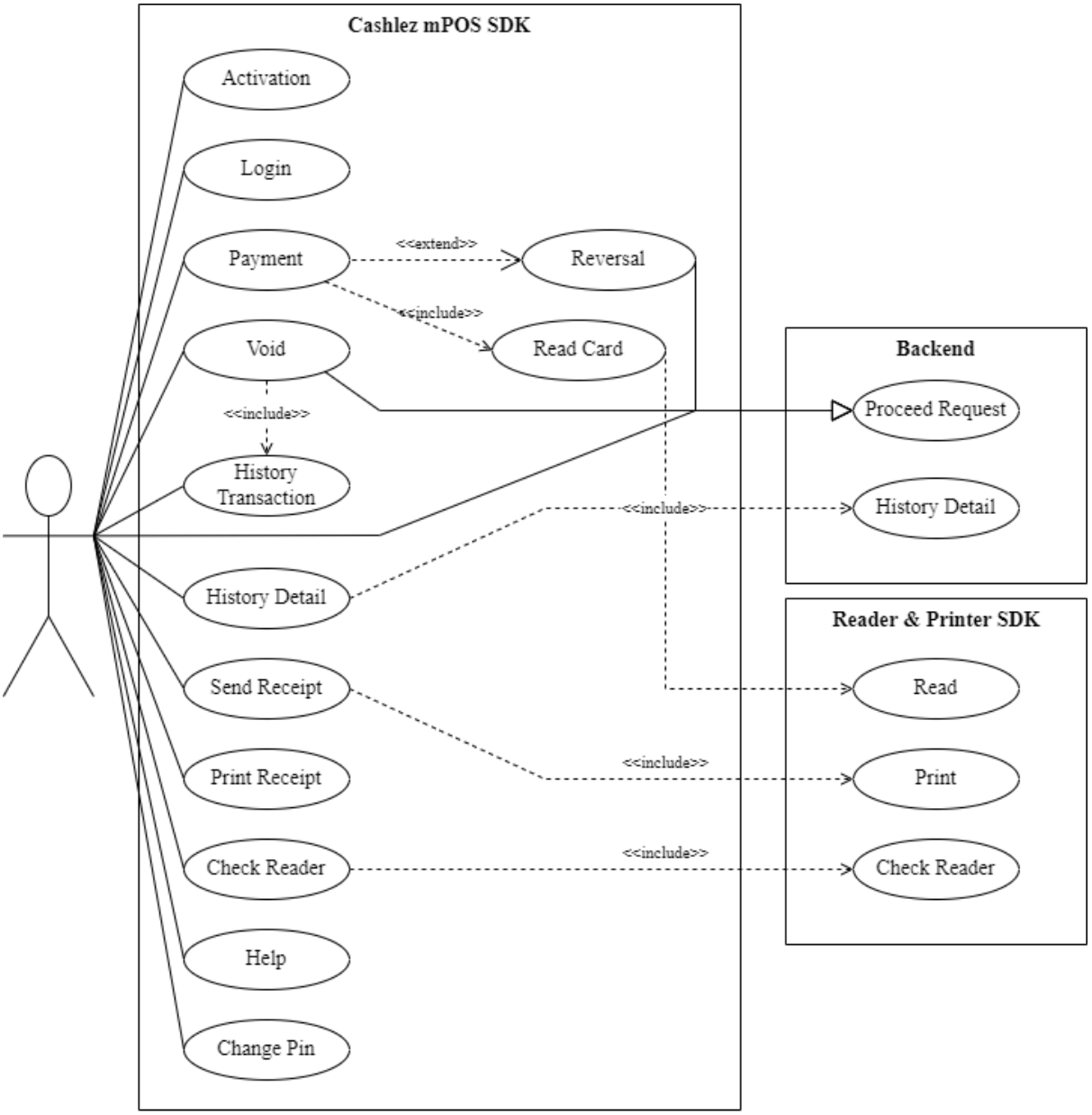

Figure 1.1 SDK Use Case Diagram

1.1. Summary

Cashlez SDK for Android is a library that allows you to accept payments in your application by leveraging Cashlez payment platform. This repository contains the SDK as well as a demo application allowing you to generate a simple payment screen and demonstrating how to use the SDK.

The following document describes the SDK integration mechanism for third party apps to use Cashlez SDK library and accept payment and how to install Cashlez SDK for Android in order to accept payments in your Android application. The integration allows Cashlez to service payment capabilities to third party apps without the need for it to be PCI DSS certified.

This type of integration requires the third-party app to include Cashlez SDK library inside. The third-party app invokes function, receives responses and listens to events from Cashlez SDK library to process payment. Below is a use case diagram of Cashlez MPOS SDK (Figure 1.1).

1.2. Requirements

The SDK is available for Android that must have the following:

1. Bluetooth version 2.0 or above
2. Google Play Service
3. API 16 or Android 4.1 (Jelly Bean)
4. GPS

1.3. Supported Reader and Printer

The following are the supported readers and printers:

1. Support printer and reader BBPOS

2. Sample App/Code

2.1 Summary

This SDK documentation includes an example app on how to use and the best practice of using the SDK. The example app is delivered with the Java source code.

Prior knowledge of Android Java programming, Gradle build and Android Studio IDE are required to understand the sample app. Knowledge in Model-View-Presenter (MVP) design pattern is also a recommendation to understand the architecture of the example app. The code snippets of the example app are used throughout the document to describe how the SDK should be used.

2.2 Availability

The link to download the example app should be available and given with the documentation, otherwise please contact your Cashlez contact person to request one. Currently Cashlez have iOS SDK and Android SDK.

2.3 Implementation of Sample App/Code

Extract the sample rar code that has been provided from the Cashlez Product Team. Then open a new project in android studio or idx, select the extracted project.

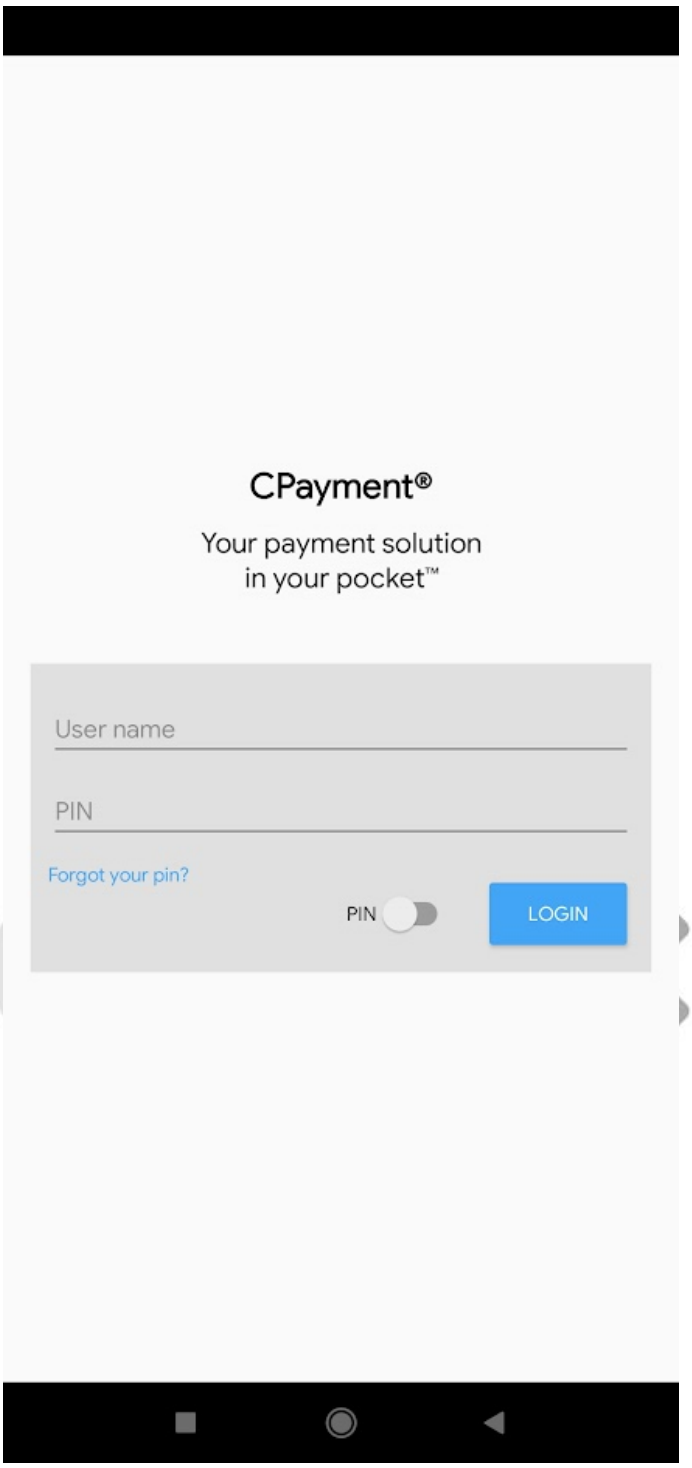

Figure 2.1 Example App Login Screen

When the import is successful and the dependencies are resolved, the module can be deployed in an android mobile phone. The example app Login screen is shown in Figure 2.4. To interact with the card reader dongle the example app must be

deployed in a real device, currently using an android emulator is not yet supported.

2.4 Implementation of Cashlez Lib or SDK

1. Download Cashlez Lib yang diberikan Tim product Cashlez.
2. Download file **github.properties** yang di berikan sama Tim Product Cashlez.

3. Buka Gradle project anda, kemudian implementasikan **github.properties** ke dalam Gradle Project.
4. Buka Gradle project anda, kemudian implementasikan Library SDK Cashlez **com.cashlez.android:cashlez:x.x.x.x**

2.5 Application Interface

In this version, the UI already revamped to a whole new fresh look. On this landing page, it has a new look and compact design. We re-design this to simplify the usage of the sample for our merchant.

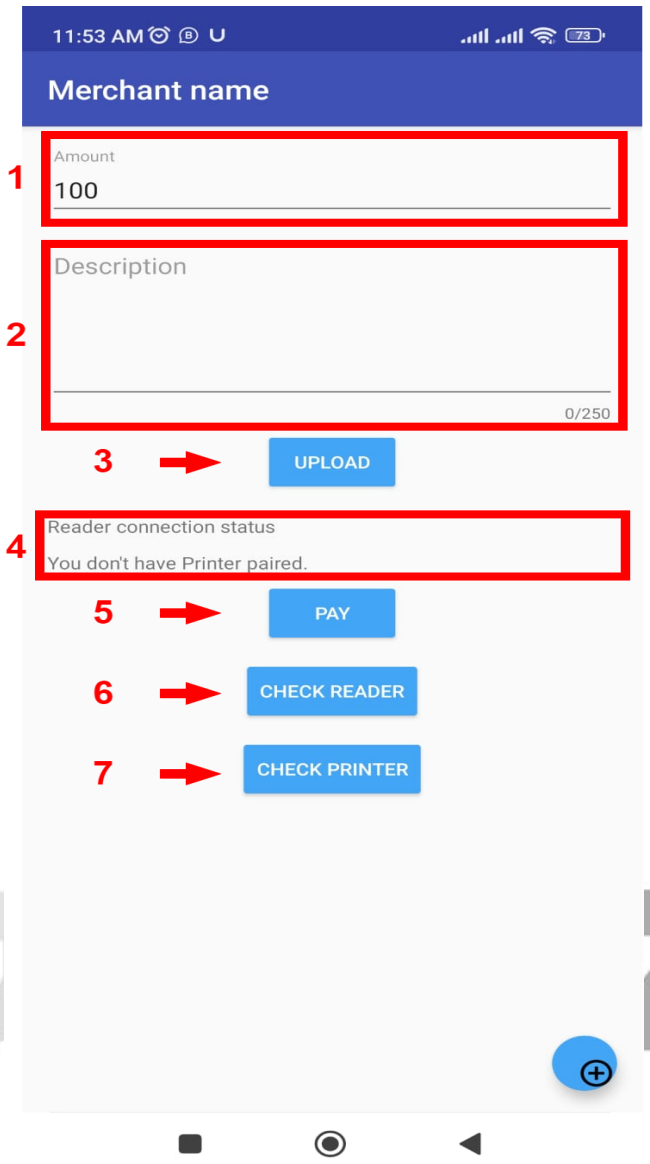

Figure 2.2 Home Page

These are the components inside this landing page based on Figure 2.5:

Table 2-1 Home Page UI Description

Home Page		
No.	**Name**	**Description**
1	Amount text box	this will add amount to pay on the payment
2	Description text area	This will add description to the payment details

3	Upload	This will upload image from local storage and put it inside to the cloud storage
4	Reader and printer status	This will return the status of the reader and printer, whenever it's connected: - if the printer is ready, it will return the status of the printer which is true. - if it's disconnected, it will return false.
5	Pay button	This button will redirect user to the payment page
6	Check reader button	Return toast alert of the reader status
7	Check printer button	Return toast alert of the printer status

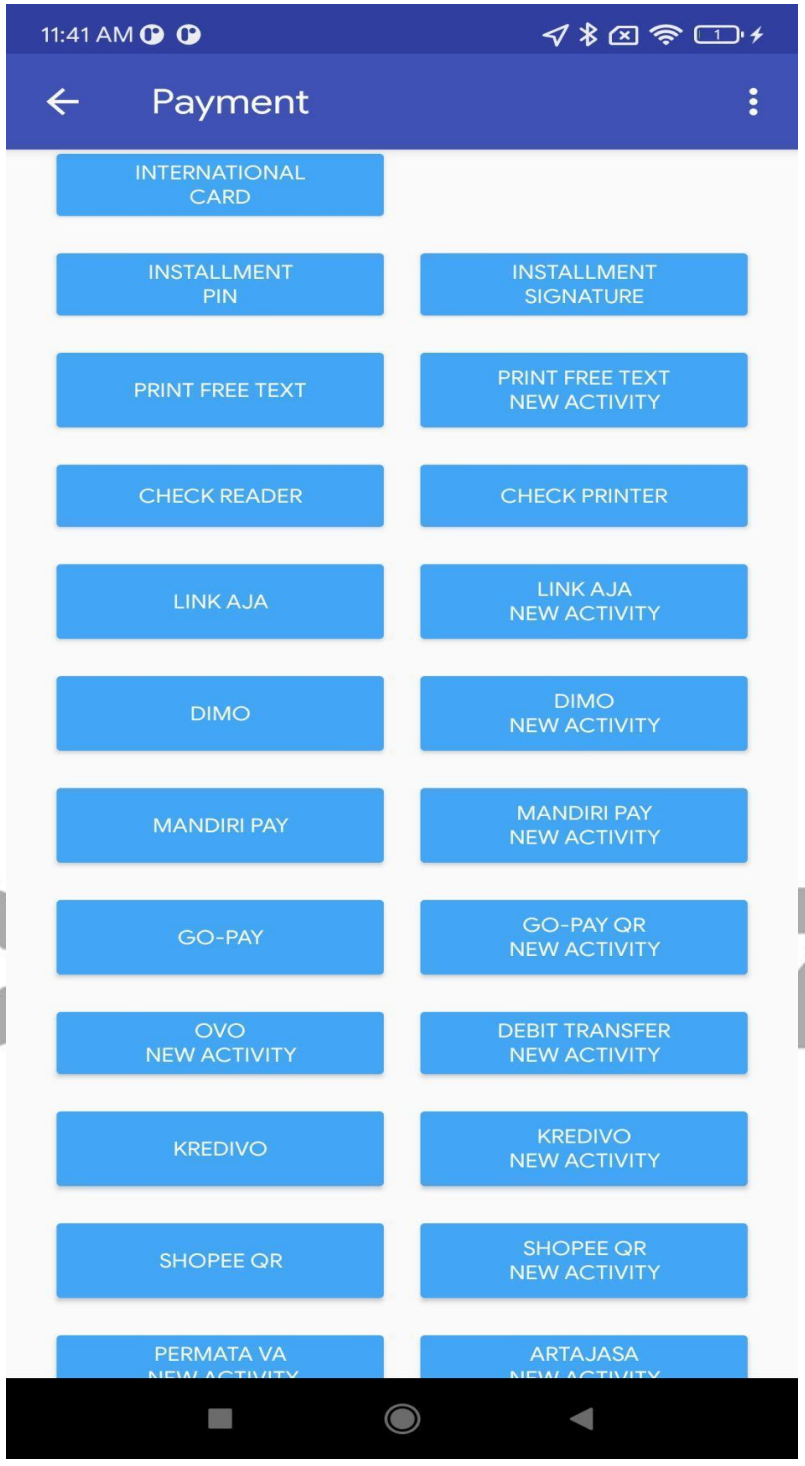

Figure 2.3 Payment Page

When redirected to the payment page, it will show the options for payment, and also the amount and payment description. Based on Figure 2.6. Several mandatory fields taken from the home page will appear on the payment page such as amount text, description text, printer, and reader status.

For each payment we have different UI, these are the list of our payment

Table 2-2 Payment List

Payment List	
Payment Options	**Payment Name**
International Card	Debit/Credit Card
Cash	Cash Money
Debit Transfer New Activity	Mini ATM bersama (Bank Transfer)
LinkAja New Activity	Payment QRIS LinkAja
Go-Pay QR New Activity	Payment QRIS Go-Pay
OVO New Activity	Push to Pay OVO
Artajasa New Activity	VA (ATM Bersama)
Kredivo New Activity	Payment Paylater Kredivo QR
Shopee QR New Activity	Payment QRIS ShopeePay
Permata VA New Activity	Permata (ATM Bersama)
BCA VA New Activity	BCA VA
Vospay New Activity	Push to Pay (paylater)
Nobu QRIS New Activity	Payment QRIS Nobu

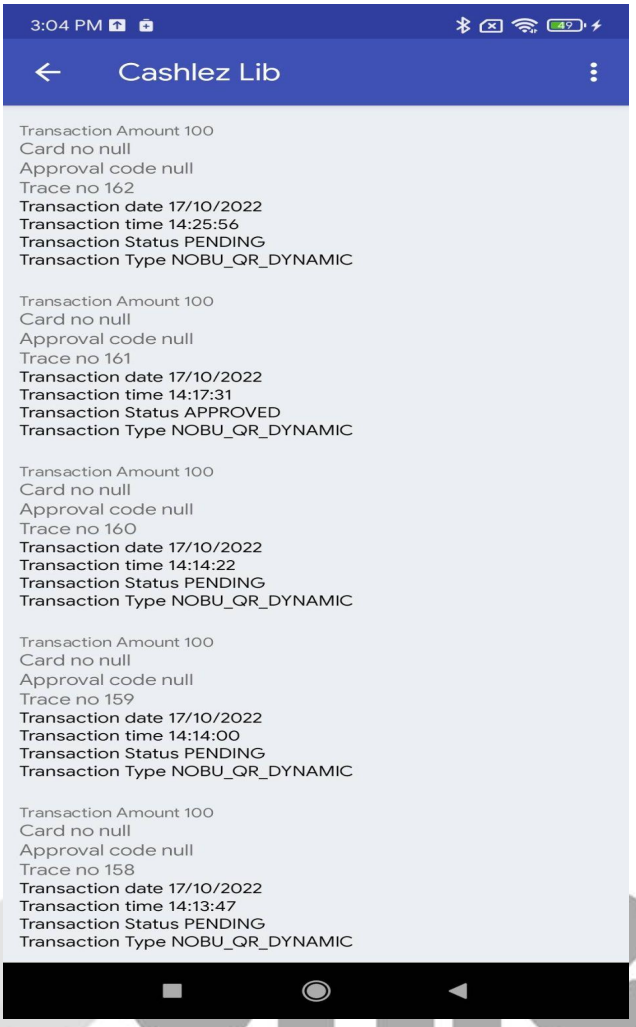

Figure 2.5 Payment History

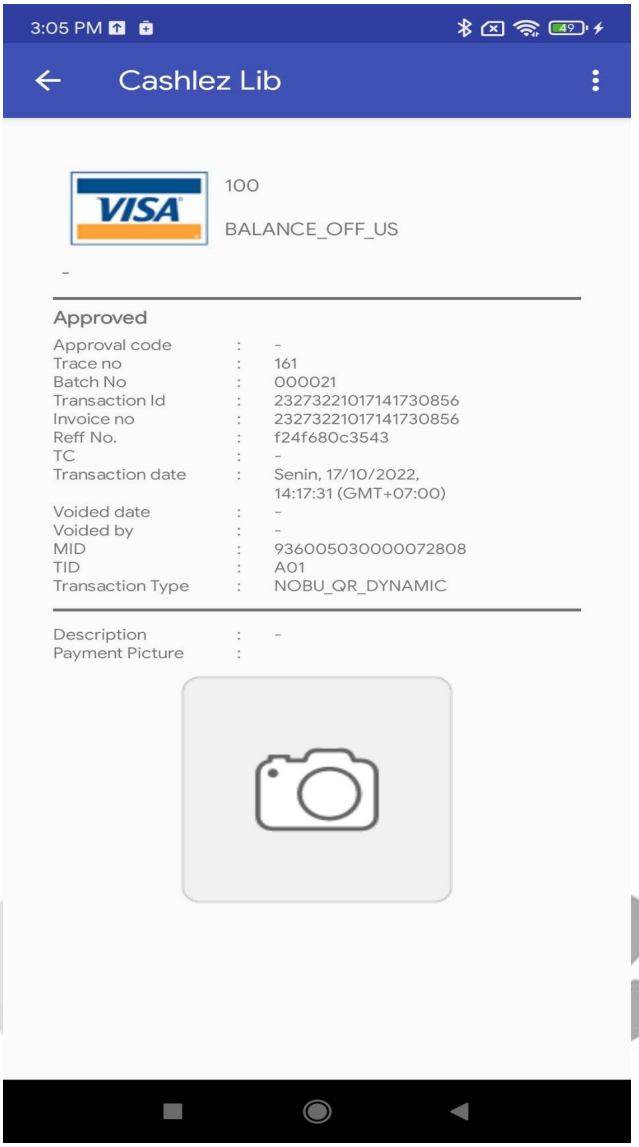

Figure 2.6 Payment History Detail

Table 2-6 Payment History Detail

Payment History Detail		
No.	**Name**	**Description**
1	Void Payment	To void Payment
2	Print	To Print Receipt Payment
3	Send Receipt	To Send Receipt Payment

3. Implementation

3.1 Settings

The following are the settings required:

1. Turn on Bluetooth on Your Device.
2. Turn on Location Service.
3. Bluetooth between Device BBPOS and your device pairing. The SDK will automatically find and use one reader and printer available in the Bluetooth paired list.

implementation 'com.cashlez.android:cashlez:2.0.3.12'

3.2 Programming Model

The programming model for each service of the SDK uses a service class to call functions and a service interface to do asynchronous callbacks. For example, the login service will have a service class called **CLLoginHandler** that has methods to do functions and **ICLLoginService** service interface to be implemented with the response handling.

3.2.1 Models

3.2.1.1. ICLLoginResponse

Table 3-1 ICLLoginResponse

ICLLoginResponse		
Name	**Type**	**Deskripsi**
userName	String	
CLMerchant	Models data CLMerchant	
CLPaymentCapability	Models data CLPaymentCapability	

3.2.1.2. TransactionType

TransactionType is a requirement to execute the type of transaction required

Table 3-2 TransactionType

TransactionType	
Name	**Value**
CASH	CASH
CREDIT	CREDIT
DEBIT	DEBIT
CREDIT_OR_INTERNATIONAL	CREDIT_OR_INTERNATIONAL
TCASH_QR	LINK AJA
MINIATM_TRANSFER	MINIATM_TRANSFER
OVO_PUSH_TO_PAY	OVO_PUSH_TO_PAY
GOPAY_QR	GOPAY_QR
KREDIVO_QR	KREDIVO_QR
SHOPEEPAY_QR	SHOPEEPAY_QR
VA_TRANSFER	VA_TRANSFER
VOSPAY	VOSPAY
NOBU_QR_DYNAMIC	NOBU_QR_DYNAMIC

3.2.1.3. ICLPayment

Table 3-3 ICLPayment

ICLPayment		
Name	**Type**	**Description**
amount	String	Required
TransactionType	TransactionType	Required
CLCardProcessingMode	CLCardProcessingMode	Required for card payment
image	String	optional
description	String	optional
phoneNo	String	optional
merchantTransactionID	String	optional

billID	String	optional
email	String	optional

3.2.1.4. ICLPaymentResponse

Table 3-4 ICLPaymentResponse

ICLPaymentResponse		
Name	**Data Type**	**Description**
userId	String	
batchNo	String	
cardNo	String	
refNo	String	
totalAmount	String	
bankName	String	
hpNo	String	
transDate	String	
transTime	String	
invoiceNo	String	
transDesc	String	
transactionId	String	
footerReceiptMerchant	String	
clientTransactionTimeZone	String	
transactionType	TransactionType (enum)	
userName	String	
merchantTransactionId	String	
responseCode	String	
aid	String	
approvalCode	String	
traceNo	String	
cardHolderName	String	

cardType	String	
applicationLabel	String	
approvedCurrencyCode	String	
transactionStatus	Integer	
AIDICC	String	
terminalVerificationResults	String	
applicationCryptogram	String	
footerReceiptBank	String	
merchant	CLMerchant	
readerCompanion	CLReaderCompanion	
bankSetting	CLBankSetting	
verificationMode	CLVerificationMode	
securityType	JSONServiceDTO.SECURITY_TYPE	
signature	Bitmap	
signatures	String	
itemImage	Bitmap	
ItemImage	String	
transactionRequestId	Long	
maskedPAN	String	
appStatus	String	
qrCodeContent	String	
transactionNameEnum	CLTransactionNameEnum	
transferDetail	CLTransferDetail	
emailAddress	String	
emailAddressChecked	boolean	
HPChecked	boolean	
hideLocation	String	
errorCode	String	

errorMessage	String	
hostResponseCode	String	
hostErrorMessage	String	
voidedDate	String	
voidedTime	String	
voidedBy	String	
appBankRefId	String	
appBankName	String	
appBankCode	String	
appDiscountAmount	String	
appLoyaltyName	String	
appLoyaltyType	String	
showRememberInput	boolean	
rememberMobileNo	boolean	
rememberEmail	boolean	
customerName	String	
customerMobilePhone	String	
customerEmail	String	
receiptHeaderLogo	CLReceiptHeaderLogo	
merchantLogo	String	
installmentCode	String	
installmentTenor	String	
installmentMonthlyAmount	long	
installmentName	String	
total	String	
cashTendered	String	The Cash Paid. Only for Cash
change	String	The Cash Change. Only for Cash
roundingType	String	
roundingTarget	String	
roundingValue	String	

posPaymentData	CLPosPaymentData	
authenticationId	String	
paymentName	String	
locationModel	LocationModel	
billId	String	
vaNumber	String	
expireDate	String	
responseContainer	String	
longitude	String	
latitude	String	
altitude	String	
tid	String	
mid	String	

3.2.1.5. ICLErrorResponse

Table 3-5 CLErrorResponse

ICLErrorResponse	
Name	**Type**
errorCode	Integer
errorMessage	String
hostErrorCode	Integer
hostErrorMessage	String
htppStatusCode	Integer

3.3 Login and Activation

The section shows how to log in and activate using the Android SDK library. To sign into the app, the user first gets authentication credentials from the mobile user. These credentials can be the user's username and PIN and authentication belongs to Cashlez mobile user. After a successful login user can perform all the object functions contained in this android SDK.

3.3.1 Login

The following classes and interfaces are used to log in and do activation from the

SDK. Login flow can be seen in Figure 3.1.

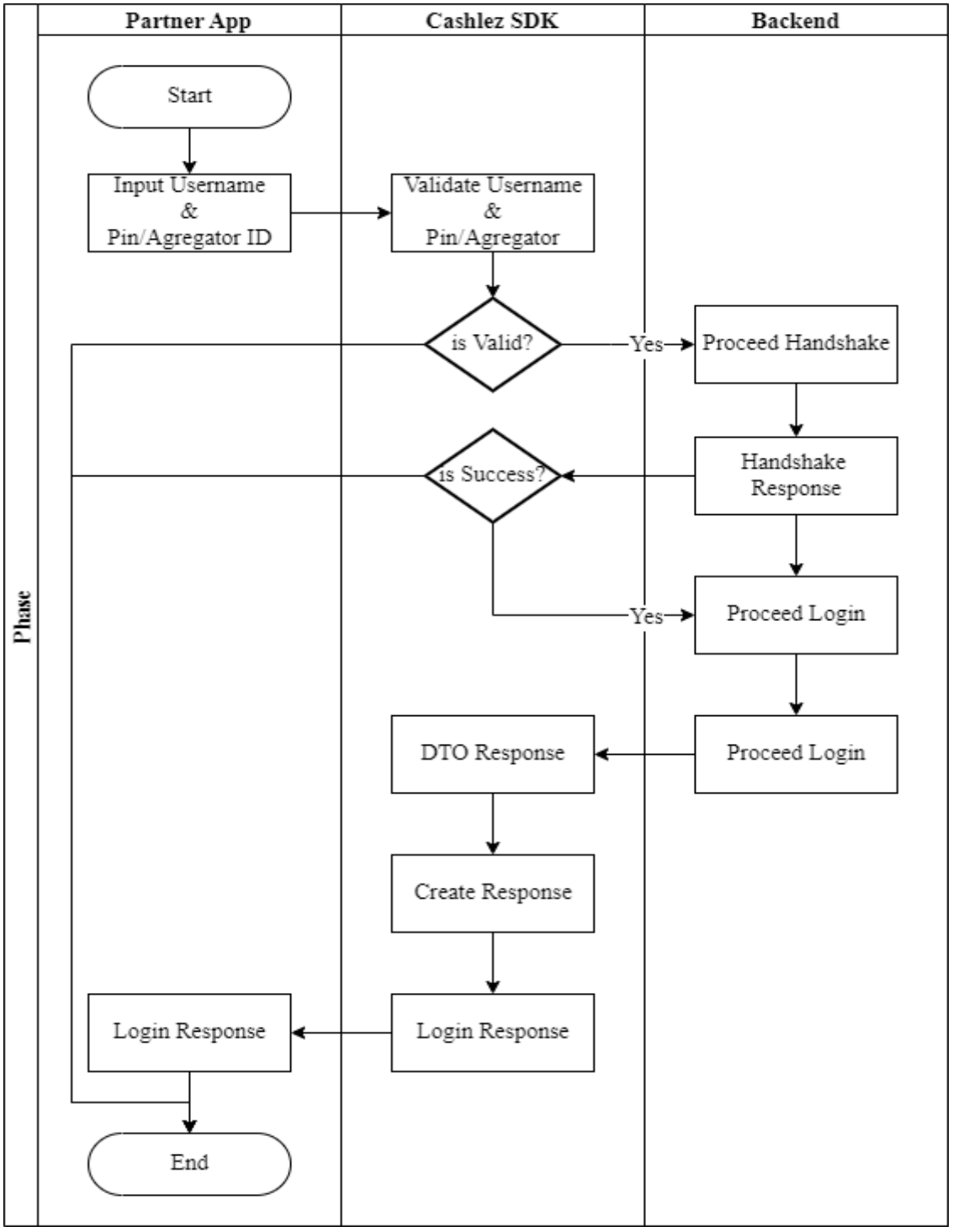

Figure 3.1 Login Flow

3.3.1.1 Login with PIN

Login with the usual validation username and password before processing the payment. The login process is provided in **CLLoginHandler**, set the user name (Username) and PIN contained in the **CLLoginHandler** before using them as parameters in the Login method. If the login process is successful then the callback is **onLoginSuccess** and can be seen in **ICLLoginService**, otherwise if the login process fails then the callback is **onLoginError** and can be seen in **ICLLoginService**.

3.3.1.2 Login with Aggregator

Aggregator login is a different type of login from normal login, using aggregator data to log in. It's easier than regular login so there's no need to set a username and PIN, just set up **doLoginAggregator**. If the login process is successful then the callback is **onLoginSuccess** and can be seen in **ICLLoginService**, otherwise if the login process fails then the callback is onLoginError and can be seen in **ICLLoginService**.

3.3.1.3 ICLLoginHandler

The **ICLLoginHandler** class is used to login using the SDK. There are two ways to log in (Table 3.1): log in using PIN and with Aggregator Login. Login with pin is the authentication used as in Cashlez App, each user has its own pin. Login with aggregator login can be used if a third-party application wants to log in on behalf of their user.

Table 3.6 ICLLoginHandler Methods

```
void doLogin(String userName, String pin);
void doLogin(String serverPublicKey, String clientPublicKey, String mobileUserId, String aggregatorId);
```

Table 3-7 CLLoginHandler

CLLoginHandler	
Methods	**Description**
doLogin(String userName, String pin);	Login process using PIN
doLogin(String serverPublicKey, String clientPublicKey, String mobileUserId, String aggregatorId);	Login process using Aggregator

3.3.1.4 ICLLoginService

ICLLoginService is a protocol provided by **ICLLoginHandler**. It will return a login response through the delegate method whenever it success or error.

Make sure that protocol is placed in class and set delegate from **CLLoginHandler** before doing login.

If activation success, then **ICLLoginService** returns and will show to the main menu.

```
onStartActivation(String mobileUpdateURL);
```

If Login success, then **ICLLoginService** returns and will show to the main menu.

```
onLoginSuccess(CLLoginResponseclLoginResponse);
```

And If authentication failed system will show an alert error message on **onLoginError**.

```
onLoginError(CLErrorResponseerrorResponse);
```

In **CLErrorResponse** If there is an error in this class it will show the reason why the error occurred like **errorCode**, **hostErrorCode**, or **httpStatusCode**.

Table 3-8 ICLLoginService

ICLLoginService	
Methods	**Description**
onStartActivation(String mobileUpdateUrl);	Function is used if the activation is successful
onLoginSuccess(CLLoginResponse response)	Callback / Reverse login process is successful
onLoginError(CLErrorResponse error)	Callback / Reverse login process is successful

3.3.2 Forgot PIN

Forgot PIN feature is provided for resetting PIN so it can be used again for login. it can send to the server and the server will send an email which is registered in the username account (Figure 3.2).

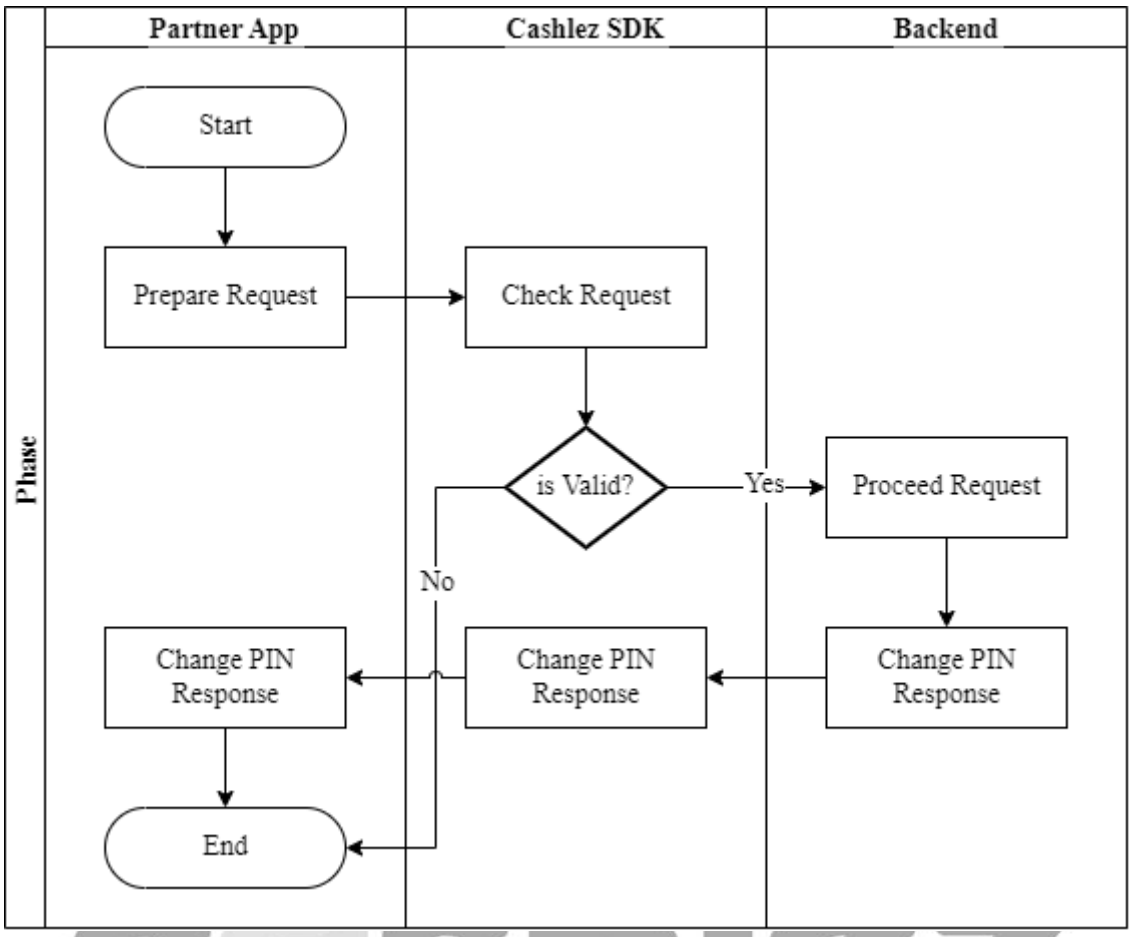

Figure 3.2 Forgot PIN Flow

3.3.2.1 ICLManagePasswordHandler

ICLManagePassword class main to do forgot pin function and this doChangePassword this method as execution

```
void doChangePassword(String userName);
```

Table 3-9 CLManagePasswordHandler

ICLManagePassworHandler	
Methods	**Description**
doChangePassword(String userName)	this function is used to process forget the pin

3.3.2.2 ICLManagePasswordService

ICLManagePasswordService is a protocol provided by **ICLManagePasswordHandler**. This will return the forgot PIN response via the delegation method every time it is successful or wrong. Make sure the protocol is placed in the class and set the delegation from **ICLManagePasswordHandler** before forgot PIN.

The **ICLManagePasswordService** interfaces has methods/callbacks:

- When forgot PIN is success

onManagePasswordSuccess

- When forgot PIN is failed

onManagePasswordError

Table 3-10 ICLManagePasswordService

ICLManagePasswordService	
Methods	**Description**
onManagePasswordSuccess(CLManageResponse response);	This function used if forgot pin process is success
onManagePasswordError(CLErrorResponse error);	This function used if forgot pin process return failed

3.3.3 Activation

The activation service is used to do activation of a new user. The activation may not be necessary in some settings. Figure 3.3 shows the usage of activation service in the example app. Please notice the usage of **ICLActivationService** and **CLActivationHandler**

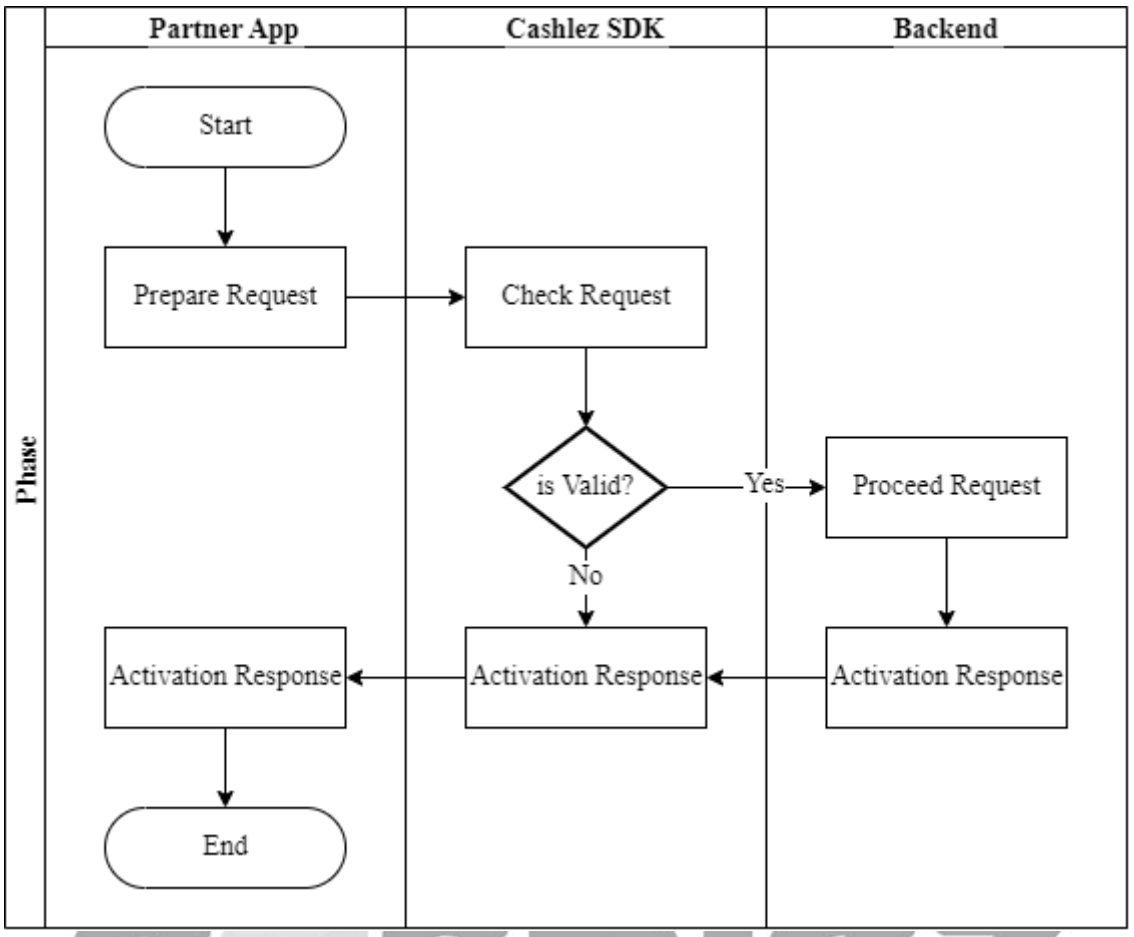

Figure 3.3 Activation Flow

3.3.3.1 ICLActivationHandler

ICLActivationHandler is main class to do user activation and this doActivate this method as execution

```
void doActivate(String activationCode);
```

Table 3-11 CLActivationHandler

ICLActivationHandler	
Methods	**Description**
doActivate(String activationCode)	this function is used to process activation

3.3.3.2 ICLActivationService

ICLActivationService is a protocol provided by I**CLActivationHandler**. It will return a response through delegate method whenever its success or error.

Make sure that protocol is placed in class and set delegate from **CLActivationHandler** before sending the data.

If the activation success will get a response

```
onActivationSuccess(CLResponse response);
```

and if fail will get error response

```
onActivationError(CLErrorResponseerrorResponse);
```

Table 3-12 ICLActivationService

ICLActivationService	
Methods	**Description**
onActivationSuccess(CLResponse response);	Callback if activation process is success
onActivationError(CLErrorResponse error)	Callback if activation process is failed

3.4 Payments and Void

Users can do the transaction depending on payment capability they got when they were doing the login (**CLLoginResponse**). for this version, SDK provided some payment like:

A. Card Payment

Card Payment			
No.	**Payment Method**	**Category**	**Void Status**
1.	Debit/Credit Card	Card Payment	Available
2.	Debit Transfer	Transfer	-

B. Payment Cash

Table 3-13 Payment Cash

Payment Cash			
No.	**Payment Method**	**Category**	**Void Status**
1	Cash	-	Available

C. QRIS

Table 3-14 ICLActivationService

QRIS		
No.	**Payment Method**	**Void Status**
1.	ShopeePay	Voided Available payment On-us
2.	Link Aja	Voided Available payment On-us
3.	Gopay	-

D. Virtual Account

Table 3-15 Virtual Account

Virtual Account			
No.	**Payment Method**	**Category**	**Void Status**
1.	BCA VA	BCA	-
2.	Permata VA	Permata	-
3.	Artajasa VA	ATM Bersama	-

E. Push to Pay

Table 3-16 Push to Pay

Push to Pay			
No.	**Payment Method**	**Category**	**Void Status**
1.	OVO	OVO Push to Pay	Available
2.	Vospay	Paylater	Available

F. Paylater QR

Table 3-17 Paylater QR

Paylater QR		
No.	**Payment Method**	**Void Status**
1.	Kredivo	-

3.4.1 Payments

The **CLPaymentHandler** class has the functions to do payment and setting up the necessary preconditions. This **ICLPaymentService** protocol interface is used to accept payment responses from the SDK. Below is Payments Flow (Figure 3.4). Communication between classes must use the **CLPayment** class.

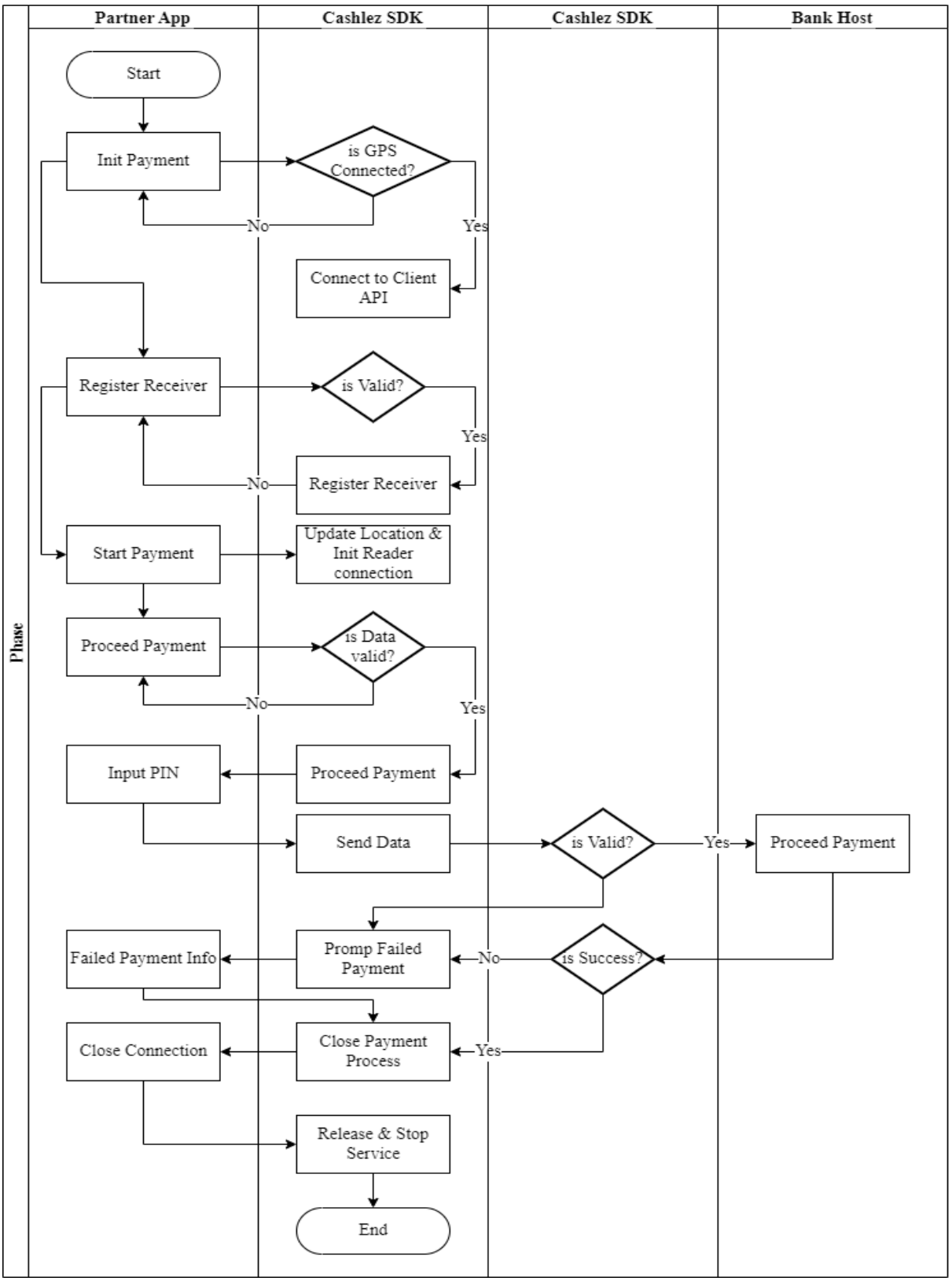

Figure 3.4 Payments Flow

3.4.1.1 ICLPaymentHandler

ICLPaymentHandler is a class for handling payment transactions, reader connection, and GPS location (Table 3.2). Before doing payment, make sure it updates the location because location data is needed for payment transactions. Then make sure the reader companion is connected for payment transactions using a card.

Table 3-18 ICLPaymentHandler Methods

ICLPaymentHandler	
Methods	**Description**
doConnectionLocationProvider()	This function updates payment locations and must be called before payment transaction.
doStartPayment(ICLPaymentService)	This function to start payment transaction
doProceedCashPayment(CLPayment payment)	This function to process Cash payment
doProceedInstallmentPayment(CLPayment payment);	This function to process Card payment Installment.
doProceedPayment(CLPayment payment)	This function to process Card payment by using location set during doStartPayment
doProceedDebitTransferPayment(CLPayment payment);	This function to process Debit transfer payment
doConfirmDebitTransferPayment(Boolean isCancelled);	This function to confirm Debit Transfer payment transaction Detail
doCheckReaderCompanion();	Automatically connect to reader
doCheckPrinterCompanion()	Automatically connect to printer
doProceedSignature(String signature)	This function is to send signature for signature verification payment
doStopUpdateLocation();	To stop requesting location updates
doUnregisterReceiver();	To check unregister receiver
doCloseCompanionConnection();	This function to disconnect between reader with mobile phone
doPrint(CLPaymentResponse paymentResponse);	To print receipt
doPrintFreeText(ArrayList<CLPrintObject> freeText;	To print receipt with free text
doLogout()	To exit from app
doCancelTransaction();	To Cancel Transaction Payment

doProceedGoPayPayment(CLPaymentpayment	To generate Transaction QRIS
doCheckGoPayStatus(CLGoPayQRResponse goPayQRResponse);	To check Status Transaction QRIS (**Pending/Success**)
doPrintGoPay(CLGoPayQRResponse goPayQRResponse);	To Print Receipt Transaction after payment success
doPrintQRGopay(Bitmap bitmap);	To Print QRIS Gopay
doProceedKredivoPayment(CLPayment payment);	To generate Transaction QR Paylater
doCheckKredivoStatus(CLKredivoResponse kredivoPayQRResponse);	To check Status Transaction QR Paylater (**Pending/Success**)
doPrintKredivo(CLKredivoResponse kredivoPayQRResponse);	To Print Receipt after Payment Success
doPrintShopeePayReceipt(CLShopeePayQrResponse paymentResponse);	To Print Receipt after Payment Success
doProceedShopeePayQr(CLPayment payment);	To generate Transaction Shopee Pay QRIS
doInquiryShopeePayQr(CLShopeePayQrResponse paymentResponse);	To Check Status QRIS ShopeePay
doPrintQRContent(Bitmap qrValue);	To Print QRIS Shopee Pay
doPrintNobuQRReceipt(CLPaymentResponse paymentResponse);	To print Receipt Nobu QRIS after payment successed
doPrintNobuQRCode(Bitmap bitmap);	To Print QRIS Nobu
doProceedNobuQRPayment(CLPayment payment);	To Generate Reception Nobu QRIS
doInquiryNobuQRPayment(CLPaymentResponse paymentResponse);	To Check Status QRIS Nobu

3.4.1.2 ICLPaymentService

ICLPaymentService is a protocol provided by **ICLPaymentHandler**. it will return a response through the delegate method whenever it's success or failed. make sure that protocol is placed in class and set a delegate from **ICLPaymentHandler** before sending the data. the ICLPaymentService interface has methods/callbacks**.**

Table 3-19 ICLPaymentService

ICLPaymentService	
Methods	**Description**
onReaderSuccess(CLReaderCompanion reader);	this callback is called when is reader found
onReaderError(CLErrorResponse error);	this callback is called when is reader not found/error
onPrinterSuccess(CLPrinterCompanion printercompanion);	callback when success printing receipt transaction
onPrinterError(CLErrorResponse error);	callback when fail printing receipt transaction
onInsertCreditCard(CLPaymentResponse paymentResponse);	callback when system accept payment with insert credit card
onInsertOrSwipeDebitCard(CLPaymentResponse paymentresponse);	callback when system accept payment with insert/swipe debit card
onSwipeDebitCard(CLPaymentResponse paymentresponse);	callback when cashlez reader recognize a debit card has been swiped
onRemoveCard(String removeCard)	callback when reader ask card to be removed
onProvideSignatureRequest(CLPaymentResponse paymentresponse);	callback when signature has to be submitted
onProvideSignatureError(CLErrorResponse error);	callback when signature is failed or error
onSignatureTimeout(CLErrorResponse error);	callback when cashlez reader give a timeout during provide signature to server
onPaymentTimeout(CLErrorResponse error);	callback when transaction request received request timeout, check last transaction to confirm transactionStatus
onPaymentDebitTransferRequestConfirmation(CLTransferDetail detail);	callback is called to return transfer detail and ask confirmation
onCashPaymentSuccess(CLPaymentResponse response)	Callback status with cash payment transaction is success
onCashPaymentError(CLErrorResponse)	callback status with cash payment transaction is error/fail

onPaymentError(CLErrorResponse error);	callback status when transaction status is error/fail
onPaymentSuccess(CLPaymentResponse response);	callback status when transaction status is success
onQROnReaderTimeout()	
onUpdateHardwareProgress(double percentage);	callback status progress to update reader
onGetHardwareInfoSuccess(Hashtable<String, String> data	callback to read info hardware is success
onGetHardwareInfoError(CLErrorResponse error)	callback to read info hardware is fail/Error
onUpdateHardwareFirmawareSuccess(String message)	callback to update hardware Firmware reader/printer is success
onUpdateHardwareFirmawareError(CLErrorResponse error)	callback to update hardware Firmware reader/printer is error/fail
onUpdateHardwareConfigurationSuccess(String message)	callback to updateHardwareConfiguration reader/printer is success
onUpdateHardwareConfigurationError(CLErrorResponse error);	callback to updateHardwareConfiguration reader/printer is error
onGoPaySuccess(CLGoPayQRResponse qrResponse);	callback when generate QR Payment is success
onGoPayError(CLErrorResponse errorResponse);	callback when generate QR Payment is fail or error
onCheckGoPayStatusSuccess(CLGoPayQRResponse paymentResponse);	callback when check status transaction success
onCheckGoPayStatusError(CLErrorResponse errorResponse);	callback when check status transaction is fail or error
onKredivoSuccess(CLKredivoResponse response);	callback when generate QR payment is success
onKredivoError(CLErrorResponse errorResponse);	callback when generate QR is fail or error

onCheckKredivoStatusSuccess(CLKredivoResponse response);	callback when check status transaction success
onCheckKredivoStatusError(CLErrorResponse errorResponse);	callback when check status transaction is fail or error
onShopeePayQrSuccess(CLShopeePayQrResponse paymentResponse);	callback when generate QRIS payment is success
onShopeePayQrError(CLErrorResponse errorResponse);	callback when generate QRIS is fail or error
onShopeePayQrCheckStatusSuccess(CLShopeePayQrResponse paymentResponse);	callback when check status transaction success
onShopeePayQrCheckStatusError(CLErrorResponse errorResponse);	callback when check status transaction is fail or error
onShopeePayQrVoidSuccess(CLVoidResponse paymentResponse);	callback when void transaction is success
onShopeePayQrVoidError(CLErrorResponse errorResponse);	callback when void transaction is error
onNobuQRPaymentSuccess(CLPaymentResponse paymentResponse);	callback when generate QRIS payment is success
onNobuQRPaymentError(CLErrorResponse errorResponse);	callback when generate QRIS is fail or error
onNobuQRInquiryPaymentSuccess(CLPaymentResponse paymentResponse);	callback when check status transaction success
onNobuQRInquiryPaymentError(CLErrorResponse errorResponse);	callback when check status transaction is fail or error
onNobuQRVoidPaymentSuccess(CLVoidResponse voidResponse);	callback when void transaction is success
onNobuQRVoidPaymentError(CLErrorResponse errorResponse);	callback when generate QRIS payment is success

3.4.1.3 ICLArtajasaVAHandler

ICLArtajasaVAHandler is a class for handling payment transactions **ARTAJASA VA**, reader connection and GPS location, before doing payment, make sure it updates the location because location data is needed for payment transactions. then make sure the reader companion is connected for payment transactions.

Table 3-20 ICLArtajasaVAHandler

ICLArtajasaVAHandler	
Methods	**Description**
doStartArtajasaVAHandler();	this function is used to start with VA
doStopArtajasaVAHAndler();	this function is used to stop VA activity
doResumeArtajasaVAHandler();	this function is used to resume VA Activity
doProceedArtajasaVAPayment(CLPayment payment, LocationUpdater locationupdate, LocationModel locationModel)	this function is used to process transaction payment Artajasa VA with location as parameter to remove the need of invoking doStartVaHandler beforehand
doProceedArtajasaVAPayment(CLPayment payment);	this function is used to process transaction payment Artajasa VA
doCheckStatusVA(CLPaymentResponse artajasaVAResponse)	this function is used to check status transaction VA
doPrintArtajasaVA(CLPaymentResponse artajasaVAResponse)	this function is used to print receipt after payment success

3.4.1.4 ICLArtajasaVAService

ICLArtajasaVAService is a protocol provided by **ICLArtajasaVAHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLArtajasaVAHandler** before sending the data. The **ICLArtajasaService** interface has methods/callbacks.

Table 3-21 ICLArtajasaService

ICLVaService	
Methods	**Description**
onArtajasaGenerateVASuccess(CLPaymentResponse paymentResponse)	callback when generate vanumber is succes

onArtajasaGenerateVAError(CLErrorResponse errorResponse)	callback when generate vanumber is fail/error
onArtajasaCheckStatusSuccess(CLPaymentResponse paymentResponse)	callback when status transaction va is success
onArtajasaCheckStatusError(CLErrorResponse errorResponse)	callback when status transaction va is error/fail
onPrinterSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success
onPrinterError(CLErrorResponse error)	callback printing receipt is error/fail

3.4.1.5 ICLBcaVaHandler

ICLBcaVaHandler is a class for handling payment transactions **BCA VA**, reader connection and GPS location, before doing payment, make sure it updates the location because location data is needed for payment transactions. then make sure the reader companion is connected for payment transactions.

Table 3-22 ICLBcaVaHandler

ICLBcaVaHandler	
Methods	**Description**
doStartBcaVaHandler();	this function is used to start with VA
doStopBcaVaHandler();	this function is used to stop VA activity
doResumeBcaVaHandler();	this function is used to resume VA Activity
doBcaVaCheckStatus(CLPaymentResponse paymentResponse)	this function is used to check status transaction VA
doProceedBcaVaPayment(CLPayment payment);	this function is used to process transaction payment BCA VA
doPrintBcaVaReceipt(CLPaymentResponse paymentResponse)	this function is used to print receipt after payment success

3.4.1.6 ICLBcaVaService

ICLBcaVaService is a protocol provided by **ICLBcaVaHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLBcaVaHandler** before sending the data. The **ICLBcaVaService** interface has methods/callbacks.

Table 3-23 ICLBcaVaService

ICLVaService	
Methods	**Description**
onBcaVaGenerateSuccess(CLPaymentResponse paymentResponse)	callback when generate vanumber is succes
onBcaVaGenerateError(CLErrorResponse errorResponse)	callback when generate vanumber is fail/error
onBcaVaCheckStatusSuccess(CLPaymentResponse paymentResponse)	callback when status transaction va is success
onBcaVaCheckStatusError(CLErrorResponse errorResponse)	callback when status transaction va is error/fail
onPrinterSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success
onPrinterError(CLErrorResponse error)	callback printing receipt is error/fail

3.4.1.7 ICLPermataVAHandler

ICLPermataVAHandler is a class for handling payment transactions **Permata VA**, reader connection and GPS location, before doing payment, make sure it updates the location because location data is needed for payment transactions. then make sure the reader companion is connected for payment transactions.

Table 3-24 ICLPermataVAHandler

ICLVaHandler	
Methods	**Description**
doStartPermataVAHandler();	this function is used to start with VA
doStopPermataVAHandler();	this function is used to stop VA activity
doResumePermataVAHandler();	this function is used to resume VA Activity
doPermataCheckStatusVA(CLPaymentResponse permataVAResponse)	this function is used to check status transaction VA
doProceedPermataVAPayment(CLPayment payment);	this function is used to process transaction payment Permata VA
doPrintPermataVaReceipt(CLPaymentResponse permataVAResponse)	this function is used to print receipt after payment success

3.4.1.8 ICLPermataVAService

ICLPermataVAService is a protocol provided by **ICLPermataVAHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLPermataVAHandler** before sending the data. The **ICLPermataVAService** interface has methods/callbacks.

Table 3-25 ICLPermataVAService

ICLVaService	
Methods	**Description**
onPermataGenerateVASuccess(CLPaymentResponse paymentResponse)	callback when generate vanumber is succes
onPermataGenerateVAError(CLErrorResponse errorResponse)	callback when generate vanumber is fail/error
onPermataCheckStatusSuccess(CLPaymentRespons e paymentResponse)	callback when status transaction va is success
onPermataCheckStatusError(CLErrorResponse errorResponse)	callback when status transaction va is error/fail
onPrinterSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success
onPrinterError(CLErrorResponse error)	callback printing receipt is error/fail

3.4.1.9 ICLGoPayQRHandler

ICLGoPayQRHandler is a class for handling payment transaction **GOPAY** reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. Then make sure the reader companion is connected for payment transactions.

Table 3-26 ICLGoPayQRHandler

ICLGoPayQRHandler	
Methods	**Description**
doStartGoPayHandler()	this function is used to start with QRISPayment
doResumeGoPayHandler()	this function is used to resume activity QRISPayment
doStopGoPayHandler()	this function is used to stop activity QRISPayment

doProceedGoPayPayment(CLPayment payment, LocationUpdater locationUpdate, LocationModel locationmodel)	this function is used to process transaction payment QRISPayment (Gopay) with location as parameter to remove the need of invoking doStartGoPayHandler beforehand
doCheckGoPayQRStatus(CLPaymentResponse paymentresponse)	this function is used to check status transaction payment QRISPayment (Gopay)
doProceedGoPayPayment(CLPayment payment)	this function is used to process transaction payment QRISPayment (Gopay)
doPrintQRContent(Bitmap qrCode)	this function to process print qrcode
doPrintGoPay(CLPaymentResponse paymentresponse)	this function to process print receipt after status transaction Approved (100)

3.4.1.10 ICLGoPayQRService

ICLGoPayQRService is a protocol provided by **ICLGoPayQRHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLGoPayQRHandler** before sending the data. The **ICLGoPayQRService** interface has methods/callbacks.

Table 3-27 ICLGoPayQRService

ICLGoPayQRService	
Methods	**Description**
onGoPayQRSuccess(CLPaymentResponse qrResponse)	Callback when generate qrpayment is success
onGoPayQRError(CLErrorResponse errorResponse)	callback when generate qrpayment is fail/Error
onCheckGoPayStatusSuccess(CLPaymentResponse qrResponse)	callback when status transaction is Success
onCheckGoPayStatusError(CLErrorResponse errorResponse)	callback when status transaction is error
onPrinterSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success
onPrinterError(CLErrorResponse error)	callback printing receipt is error/fail

3.4.1.11 ICLShopeePayQrHandler

ICLShopeePayQrHandler is a class for handling payment transaction **ShopeePay** reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. Then make sure the reader companion is connected for payment transactions.

Table 3-28 ICLShopeePayQrHandler

ICLShopeePayQrHandler	
Methods	**Description**
doStartHandlerShopeepay()	this function is used to start with QRISPayment
doResumeHandlerShopeepay()	this function is used to resume activity QRISPayment
doStopHandlerShopeepay()	this function is used to stop activity QRISPayment
doProceedShopeePayQr(CLPayment payment, LocationUpdater locationUpdate, LocationModel locationmodel)	this function is used to process transaction payment QRISPayment (ShopeePay) with location as parameter to remove the need of invoking doStartGoPayHandler beforehand
doInquiryShopeePayQr(CLPaymentResponse paymentresponse)	this function is used to check status transaction payment QRISPayment (ShopeePay)
doProceedShopeePayQr(CLPayment payment)	this function is used to process transaction payment QRISPayment (Gopay)
doPrintQRContent(Bitmap qrCode)	this function to process print qrcode
doPrintShopeePayReceipt(CLPaymentResponse paymentresponse)	this function to process print receipt after status transaction Approved (100)
doVoidShopeePayQr(String username, String password, CLPaymentResponse paymentResponse)	this function is used to process void payment

3.4.1.12 ICLShopeePayQrService

ICLShopeePayQrService is a protocol provided by **ICLShopeePayQrHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLShopeePayQrHandler** before sending the data. The **ICLShopeePayQrService** interface has methods/callbacks.

Table 3-29 ICLShopeePayQrService

ICLShopeePayQrService	
Methods	**Description**
onShopeePayQrSuccess(CLPaymentResponse paymentResponse)	Callback when generate qrpayment is success
onShopeePayQrError(CLErrorResponse errorResponse)	callback when generate qrpayment is fail/Error
onShopeePayQrCheckStatusSuccess(CLPayment Response paymentResponse)	callback when status transaction is Success
onShopeePayQrCheckStatusError(CLErrorRespo nse errorResponse)	callback when status transaction is error
onPrinterSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success
onPrinterError(CLErrorResponse error)	callback printing receipt is error/fail
onShopeePayQrVoidSuccess(CLVoidResponse paymentResponse)	callback when status void transaction is success
onShopeePayQrVoidError(CLErrorResponse errorResponse)	callback when status void transaction is error

3.4.1.13 ICLTcashQRHandler

ICLTcashQRHandler is a class for handling payment transaction **Link AJA** reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. Then make sure the reader companion is connected for payment transactions.

Table 3-30 ICLTcashQRHandler

ICLTcashQRHandler	
Methods	**Description**
doStartTCashHandler()	this function is used to start with QRISPayment
doResumeTCashHandler()	this function is used to resume activity QRISPayment
doStopTCashHandler()	this function is used to stop activity QRISPayment

doProceedTCashQRPayment(CLPayment payment, LocationUpdater locationUpdate, LocationModel locationmodel)	this function is used to process transaction payment QRISPayment (Link Aja) with location as parameter to remove the need of invoking doStartGoPayHandler beforehand
doCheckTCashQRStatus(CLTCashQRResponse qrResponse)	this function is used to check status transaction payment QRISPayment (Link Aja)
doProceedTCashQRPayment(CLPayment payment)	this function is used to process transaction payment QRISPayment (Link Aja)
doPrintQRContent(Bitmap qrCode)	this function to process print qrcode
doPrintTcashQR(CLTCashQRResponse responseReceipt)	this function to process print receipt after status transaction Approved (100)
doVoidTcashQRPayment(String username, String password, CLPaymentResponse paymentResponse)	this function is used to process void payment

3.4.1.14 ICLTCashQRService

ICLTCashQRService is a protocol provided by **ICLTCashQRHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **CLTCashQRHandler** before sending the data. The **ICLTCashQRService** interface has methods/callbacks.

Table 3-31 ICLTCashQRService

ICLShopeePayQrService	
Methods	**Description**
onTCashQRSuccess(CLTCashQRResponse qrResponse)	Callback when generate qrpayment is success
onTCashQRError(CLErrorResponse errorResponse)	callback when generate qrpayment is fail/Error
onCheckTCashQRStatusSuccess(CLTCashQRResponse paymentResponse)	callback when status transaction is Success
onCheckTCashQRStatusError(CLErrorResponse errorResponse)	callback when status transaction is error
onPrinterSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success

onPrinterError(CLErrorResponse error)	callback printing receipt is error/fail
onVoidTcashQRSuccess(CLVoidResponse paymentResponse)	callback when status void transaction is success
onVoidTcashQRError(CLErrorResponse errorResponse)	callback when status void transaction is error

3.4.1.15 ICLVospayHandler

ICLVospayHandler is a class for handling payment transaction **Vospay**, reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. then make sure the reader companion is connected for payment transactions.

Table 3-32 ICLVospayHandler

ICLVospayHandler	
Methods	**Description**
doStartVospayHandler()	this function is used to start with Vospay
doResumeVospayHandler()	this function is used to resume activity Vospay
doStopVospayHandler()	this function is used to stop activity Vospay
doProceedVospayPayment()	this function is used to process transaction payment Vospay with location as parameter to remove the need of invoking doStartVospayHandler beforehand
doInquiryVospayPayment	this function is used to check status transaction payment Vospay
doVoidedVospayPayment()	this function is invoked to void payment Vospay
doPrintReceiptVospay()	this function to process print receipt after status transaction Approved (100)

3.4.1.16 ICLVospayService

ICLVospayService is protocol provided from **ICLVospayHandler**. it will return response through delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLVospayHandler** before send the data. the **ICLVospayService** interfaces has methods/callbacks.

Table 3-33 ICLVospayService

ICLVospayService	
Methods	**Description**
onVospayPaymentSuccess(CLPaymentResponse response)	Callback when push vospay payment is success
onVospayPaymentError(CLErrorResponse error)	callback when push vospay is fail/Error
onVospayInquirySuccess(CLPaymentResponse response)	callback when status transaction is Success
onVospayInquiryError(CLErrorResponse error)	callback when status transaction is error
onVospayVoidedPaymentSuccess(CLVoidResponse response)	callback when status void transaction is success
onVospayVoidedPaymentError(CLErrorResponse error)	callback when status void transaction is error/fail
onPrintingSuccess(CLPrinterCompanion printercompanion)	callback printing receipt is success
onPrintingError(CLErrorResponse error)	callback printing receipt is error/fail

3.4.1.17 ICLOvoHandler

ICLOvoHandler is a class for handling payment transaction **OVO**, reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. then make sure the reader companion is connected for payment transactions.

Table 3-34 ICLOvoHandler

ICLOvoHandler	
Methods	**Description**
doStartOvoHandler()	this function is used to start with Ovo
doResumeOvoHandler()	this function is used to resume activity OVO
doStopOvoHandler()	this function is used to stop activity OVO
doOvoPayment(CLPayment payment, LocationUpdater locationUpdater, LocationModel locationModel)	this function is used to process transaction payment OVO with location as parameter to remove the need of invoking doStartPushToPayHandler beforehand

doOvoPayment(CLPayment payment)	this function is used to process transaction payment OVO
doOvoInquiry	this function is used to check status transaction payment OVO
doOvoVoidPayment	this function is invoked to void payment OVO
doPrintOvo	this function to process print receipt after status transaction Approved (100)

3.4.1.18 ICLOvoService

ICLOvoService is a protocol provided by **ICLOvoHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set a delegate from **CLOvoHandler** before sending the data. The **ICLOvoService** interface has methods/callbacks**.**

Table 3-35 ICLOvoService

ICLOvoService	
Methods	**Description**
onOvoPaymentSuccess(CLPaymentResponse response)	Callback when pustopay OVO is success
onOvoPaymentError(CLErrorResponse error)	callback when pustopay OVO is fail/Error
onOvoInquirySuccess(CLPaymentResponse response)	callback when status transaction is Success
onOvoInquiryError(CLErrorResponse error)	callback when status transaction is error
onOvoVoidPaymentSuccess(CLVoidResponse response)	callback when status void transaction is success
onOvoVoidPaymentError(CLErrorResponse error)	callback when status void transaction is error/fail
onPrintingSuccess(CLPrinterCompanion printercompanion)	callback printing receipt is success
onPrintingError(CLErrorResponse error)	callback printing receipt is error/fail

3.4.1.19 ICLCashlezLinkService

Table 3-36 ICLCashlezLinkService

ICLCashlezLinkService	
Methods	**Description**
onCzLinkGenerateUrlSuccess(CLPaymentResponse paymentResponse)	Callback when the payment link successfully generated
onCzLinkGenerateUrlError(CLErrorResponse errorResponse)	Callback when the payment link failed to generate
onPrintingSuccess(CLPrinterCompanion printerCompanion)	callback printing receipt is success
onPrintingError(CLErrorResponse errorResponse)	callback printing receipt is error/fail

3.4.1.20 ICLKredivoHandler

ICLKredivoHandler is a class for handling payment transaction **Kredivo** reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. Then make sure the reader companion is connected for payment transactions.

Table 3-37 ICLKredivoHandler

ICLKredivoHandler	
Methods	**Description**
doStartKredivoHandler()	this function is used to start with Kredivo
doResumeKredivoHandler()	this function is used to resume activity Kredivo
doStopKredivoHandler()	this function is used to stop activity Kredivo
doProceedKredivoPayment(CL Payment payment, LocationUpdater locationUpdater, LocationModel locationModel)	this function is used to process transaction payment Kredivo with location as parameter to remove the need of invoking doStartVospayHandler beforehand
doCheckKredivoStatus	this function is used to check status transaction payment Kredivo
doProceedKredivoPayment()	this function is used to process transaction payment Kredivo
doPrintKredivo()	this function to process print receipt after status transaction Approved (100)

doPrintKredivoQR	this function to print QRCode

3.4.1.21 ICLKredivoService

ICLKredivoService is a protocol provided by **ICLKredivoHandler.** it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set a delegate from **ICLKredivoHandler** before sending the data. The **ICLKredivoService** interface has methods/callbacks**.**

Table 3-38 ICLKredivoService

ICLKredivoService	
Methods	**Description**
onKredivoSuccess(CLPaymentResponse response)	Callback when pustopay Kredivo is success
onKredivoError(CLErrorResponse error)	callback when pustopay Kredivo is fail/Error
onCheckKredivoStatusSuccess(CLPaymentResponse response)	callback when status transaction is Success
onCheckKredivoStatusError(CLErrorResponse error)	callback when status transaction is error
onPrintingSuccess(CLPrinterCompanion printercompanion)	callback printing receipt is success
onPrintingError(CLErrorResponse errorResponse)	callback printing receipt is error/fail

3.4.1.22 ICLNobuQRHandler

ICLNobuQRHandler is a class for handling payment transaction **Nobu** reader connection and GPS location. Before doing payment, make sure it updates the location because location data is needed for payment transactions. Then make sure the reader companion is connected for payment transactions.

ICLNobuQRHandler	
Methods	**Description**
doStartHandlerNobuQris()	this function is used to start with QRISPayment
doResumeHandlerNobuQris()	this function is used to resume activity QRISPayment
doStopHandlerNobuQris()	this function is used to stop activity QRISPayment

doPayNobuQris(CLPayment payment);	this function is used to process transaction payment QRISPayment (Nobu)
doPayNobuQris(CLPayment payment, LocationUpdater locationUpdater, LocationModel locationModel);	this function is used to process transaction payment QRISPayment (Nobu) with location as parameter to remove the need of invoking doStartGoPayHandler beforehand
doCheckStatusNobuQris(CLPaymentResponse paymentResponse);	this function is used to check status transaction payment QRISPayment (Nobu)
doPrintQRContent(Bitmap qrCode)	this function to process print qrcode
doPrintReceiptNobuQris(CLPaymentResponse paymentResponse);	this function to process print receipt after status transaction Approved (100)
doVoidNobuQris(String userName, String password, CLPaymentResponse paymentResponse);	this function is used to process void payment

3.4.1.23 ICLNobuQRService

ICLNobuQRService is a protocol provided by I**CLNobuPayQRHandler**. it will return a response through the delegate method whenever it's success or error. make sure that protocol is placed in class and set delegate from **ICLNobuQRHandler** before sending the data. The **ICLNobuQRService** interface has methods/callbacks.

ICLNobuQRService	
Methods	**Description**
onNobuQRPaymentSuccess(CLPaymentResponse paymentResponse);	Callback when generate qrpayment is success
onNobuQRPaymentError(CLErrorResponse errorResponse);	callback when generate qrpayment is fail/Error
onNobuQRCheckStatusPaymentSuccess(CLPaymentResponse paymentResponse);	callback when status transaction is Success
onNobuQRCheckStatusPaymentError(CLErrorResponse errorResponse);	callback when status transaction is error
onPrintingSuccess(CLPrinterCompanion printerCompanion);	callback printing receipt is success

onPrintingError(CLErrorResponse errorResponse);	callback printing receipt is error/fail
onNobuQRVoidPaymentSuccess(CLVoidResponse voidResponse);	callback when status void transaction is success
onNobuQRVoidPaymentError(CLErrorResponse errorResponse);	callback when status void transaction is error

3.4.2 Voided Payment

The void service is used to void the mPos debit and credit sale transaction. Voiding basically cancels transactions. It does not delete it but clears the amount. Cashlez transactions can be voided only if they are not settled yet. Below is Void flow (Figure 3.5).

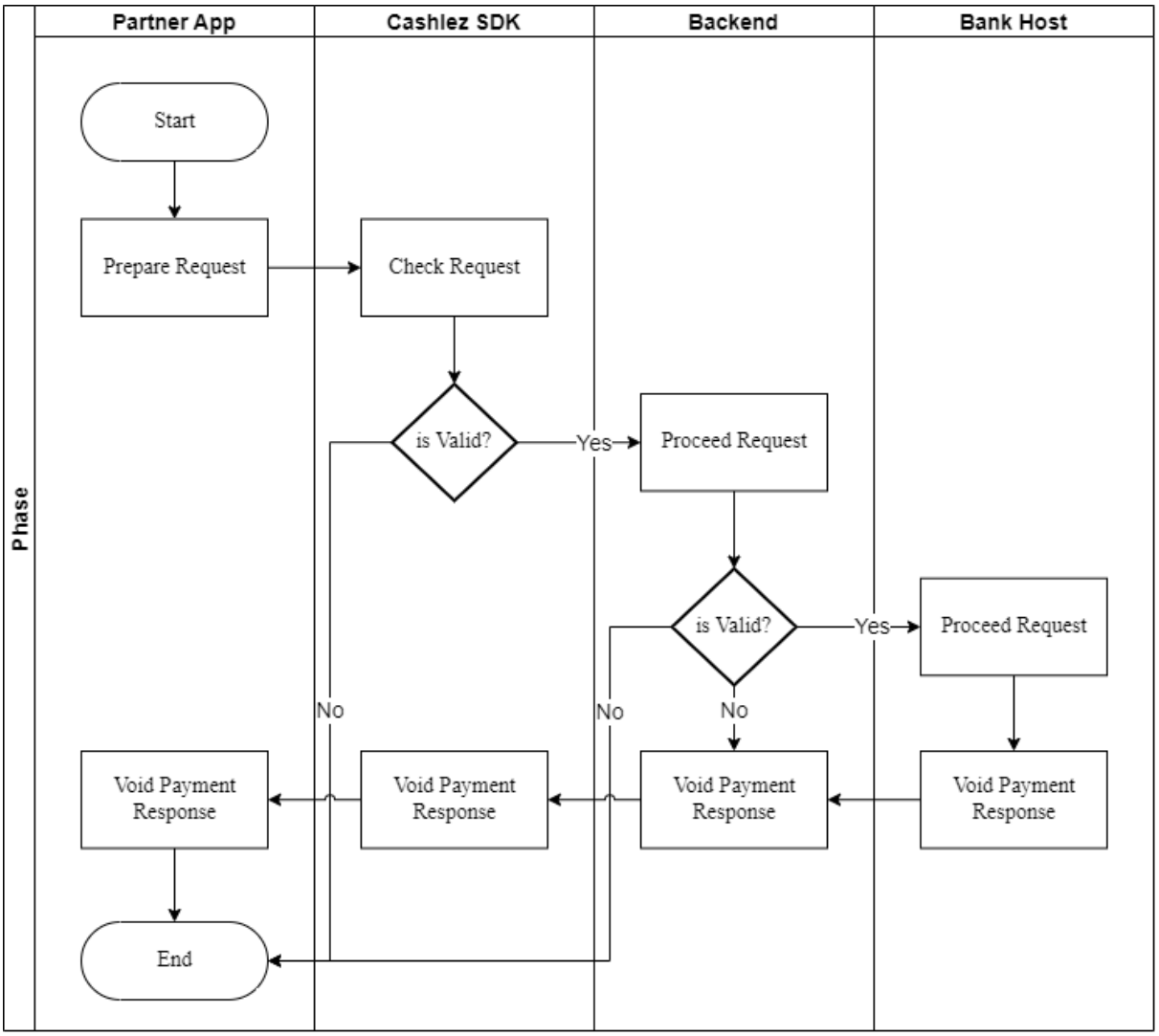

Figure 3.5 Void Payment Flow

3.4.2.1 ICLVoidPaymentHandler

The **ICLVoidPaymentHandler** is a class for canceling approved payment, it provides **doVoidPayment**. method using **ICLVoidPaymentHandler** as a parameter object.

Table 3-39 ICLVoidPaymentHandler

ICLVoidPaymentHandler	
Methods	**Description**
doVoidePayment(String userName, String Password, CLPaymentResponse paymentresponse)	this function is used to process void payment

This function void transaction details using the administrative username and password. The detail of the transaction to be voided is placed in the **CLVoidResponse** response object like voided by, voided date, voided time.

3.4.2.2 ICLVoidService

The **ICLVoidService** is a protocol provided by **ICLVoidPaymentHandler**. It is used to return the result of a void process. (**onVoidPaymentSuccess** and **onVoidPaymentError**)

This callback is called when void transaction succeeded

onVoidPaymentSuccess

This callback is called when void transaction failed or there is an error

onVoidPaymentError

Table 3-40 ICLVoidService

ICLVoidService	
Methods	**Description**
onVoidPaymentSuccess(CLVoidResponse response	callback when void payment success

onVoidPaymentError(CLErrorResponse error)	callback when void payment fail/error

3.5 Payment History and Detail

The following section shows how to check the latest payments and get details of every transaction. the services can return a valid response only if only the authentication with the login service is successful and not expired.

3.5.1 Payment History

The payment history service is used to get historical data of the transaction. it is strongly advised to use this service to get the valid transaction status when time out occurs during payment.

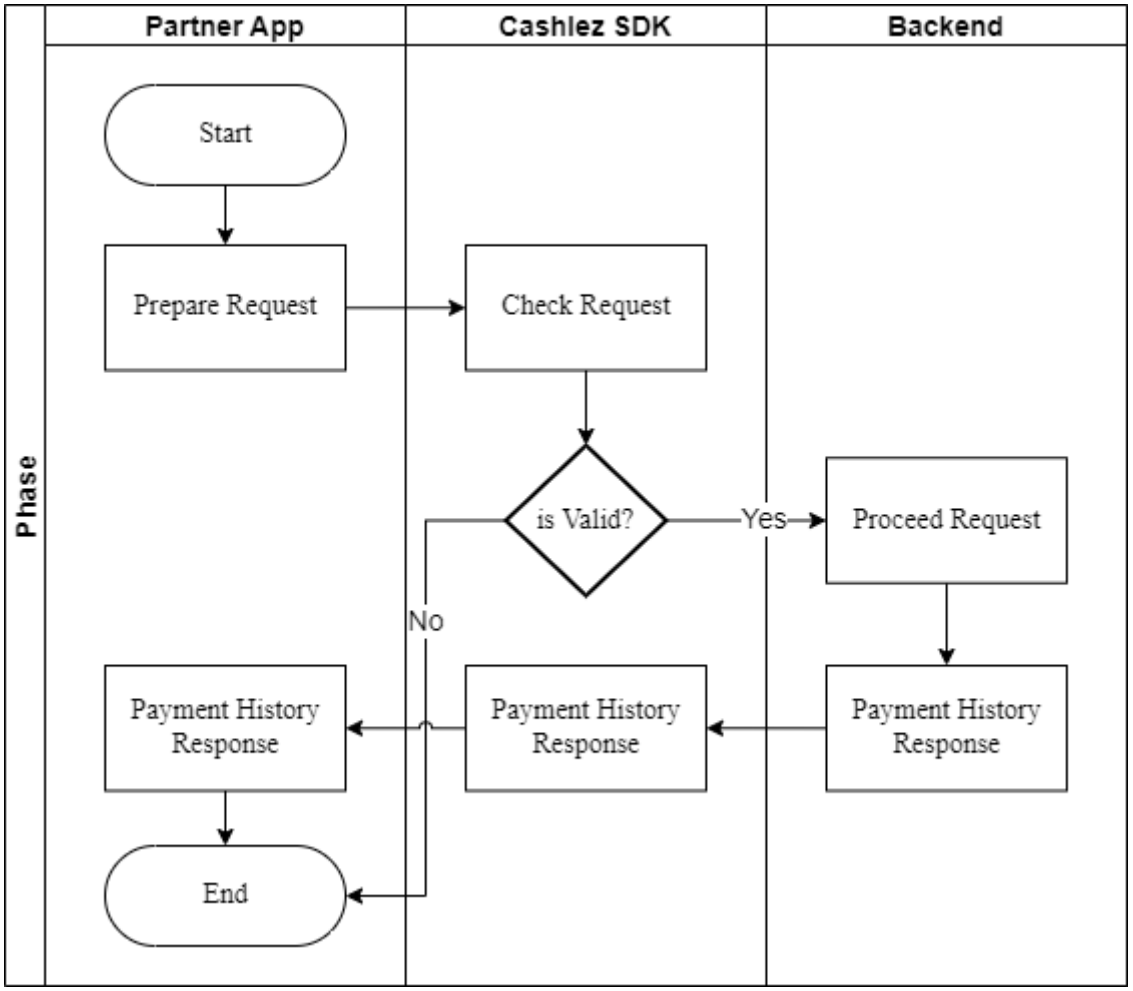

Figure 3.6 Payment History Flow

3.5.1.1 ICLPaymentHistoryHandler

The I**CLPaymentHistoryHandler** service class mainly used to get transaction history (Table 3.7).

Table 3-41 ICLPaymentHistoryHandler

ICLPaymentHistoryHandler	
Methods	**Description**
doGetSalesHistory(int page, String param1, String param2)	This function gets transaction history based on invoice number and approval code descending on transaction time. tine input page is the pagination indicator with fixed 5 transactions per-page.
doGetPaymentByTransactionId(int page, String transactionId)	this function gets transaction history based on TxId
doGetPaymentByInvoiceAndApprovalCode(int page, String invoiceNo, String approvalCode)	this function get transaction history based on invoice approval code
doGetPaymentByMerchantTransactionId(int page, String merchantTransactionId)	this function gets transaction history based on merchant transaction Id
doGetPaymentByDate(int page, String transactionDate)	this function gets transaction history based on date

3.5.1.2 ICLPaymentHistoryService

ICLPaymentHistoryService is a protocol provided by **ICLPaymentHistoryHandler**. It will return a response through the delegate method whenever it throws a success or an error. Make sure that protocol is placed in class and set a delegate from **ICLPaymentHistoryHandler** before sending the data.
The **ICLPaymentHistoryService** interfaces has methods/callbacks:

This callback is called when user can see transaction history

onSalesHistorySuccess

This callback is called when user can't see transaction history because there is

an error

onSalesHistoryError

ICLPaymentHistoryService is a protocol provided by **ICLPaymentHistoryHandler**. it will return a response through the delegate method whenever it throws a success or an error. make sure that protocol is placed in class and set a delegate from **ICLPaymentHistoryHandler** before sending the data. The **ICLPaymentHistoryService** interface has method/callbacks.

Table 3-42 ICLPaymentHistoryService

ICLPaymentHistoryService	
Methods	**Description**
onSalesHistorySuccess(CLPaymentHistoryResponse response)	This callback is called when user can see transaction history
onSalesHistoryError(CLErrorResponse error)	This callback is called when user can't see transaction history because there is an error

3.5.2 Payment History Detail

Payment history detail feature is to show detail of one payment transaction from list payment history. It contains a data card, amount, payment status, etc. Below is Payment History Detail flow (Figure 3.7).

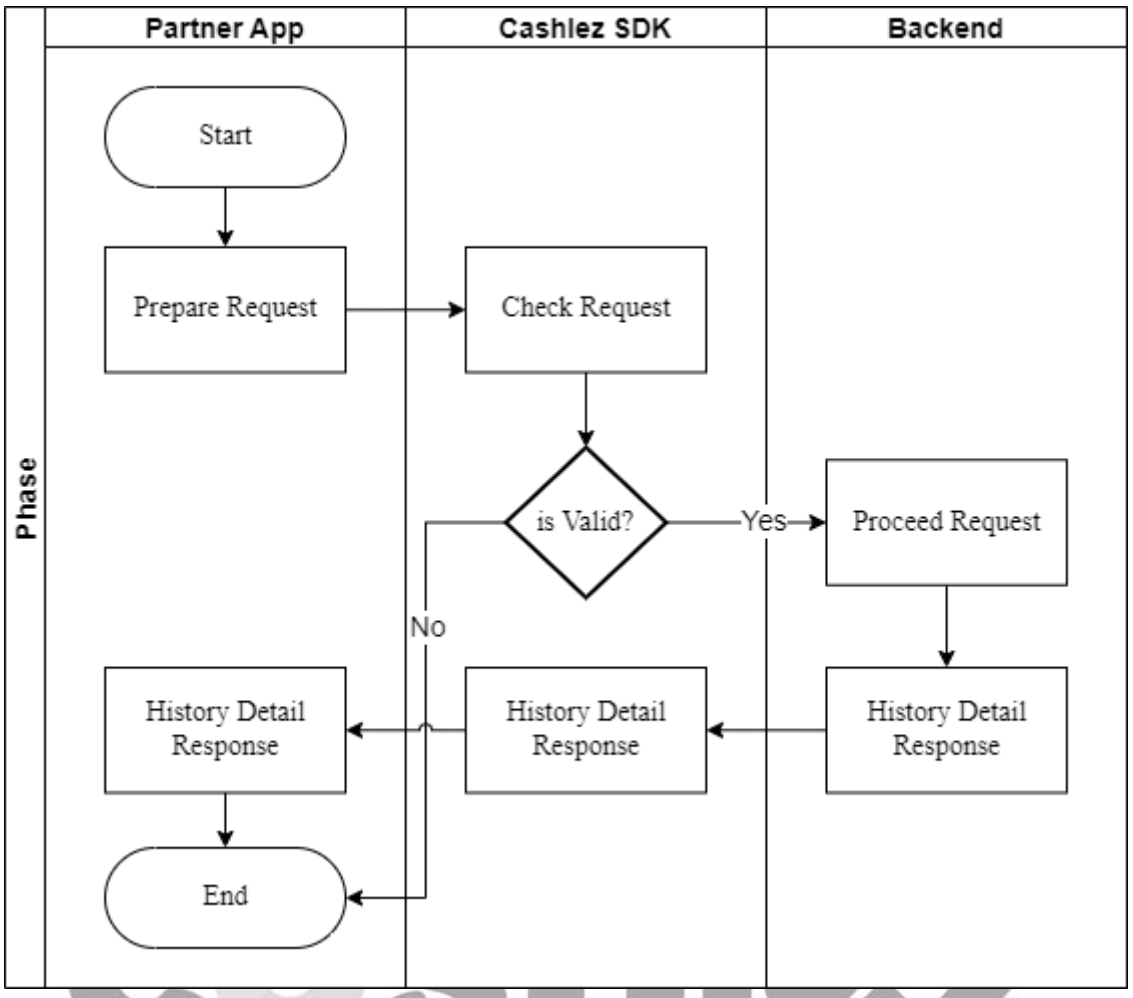

Figure 3.7 Payment History Detail Flow

Payment history detail feature is to show detail of one payment transaction from list payment history. it contains a data card, amount, payment status, etc.

3.5.2.1 ICLPaymentHistoryDetailHandler

ICLPaymentHistoryDetailhandler is a class for handling payment history detail requests.

This function gets transaction detail based on transaction identifier

doGetSalesHistoryDetail

ICLPaymenetHistoryDetailHandler is a class for handling payment detail requests.

Table 3-43 ICLPaymentHistoryDetailHandler

ICLPaymentHistoryDetailHandler	
Methods	**Description**
doGetSalesHistoryDetail(String transactionId)	this function gets transaction detail based on transaction identifier

3.5.2.2 ICLPaymentHistoryDetailService

ICLPaymentHistoryDetailService is protocol provided from **ICLPaymentHistoryDetailHandler**. It will return a response through the delegate method whenever it throws a success or an error. Make sure that protocol is placed in class and set a delegate from **ICLPaymentHistoryDetailHandler** before sending the data (Table 3.8).

Table 3.44 ICLPaymentHistoruDetailService Methods

ICLPaymentHistoryDetailService	
Methods	**Description**
onSalesHistoryDetailSuccess	This callback is called to get the transaction details.
onSalesHistoryDetailError	This callback is called when user can't see transaction detail history because there is error
onSalesHistoryImageSuccess	This callback is called when success showing image
onSalesHistoryImageError	This callback is called when fail showing image

3.6 Other Features

Besides the basic services there are also additional services provided by the SDK.

3.6.1 Product Image

The services are used to upload and download images. The image is mainly product image, but not restricted to provide invoice images or others.

3.6.1.1 ICLUploadHandler

The **ICLUploadHandler** class mainly used to get transaction history

This function uploads images from the local android file to the cloud.

doUpload

The **ICLUploadHandler** class mainly used to get transaction history.

Table 3-45 ICLUploadHandler

ICLUploadHandler	
Methods	**Description**
doUpload(String photoPath)	This function uploads images from the local android file to the cloud.

3.6.1.2 ICLUploadService

The **ICLUploadService** interfaces has methods/callbacks:

This callback is called when the upload is finished.

onUploadImageSuccess

This callback is called when images can’t be uploaded

onUploadImageError

The **ICLUploadService** interfaces has methods/callback.

Table 3-46 ICLUploadService

ICLUploadService	
Methods	**Description**
onUploadSuccess(CLUploadResponse response)	this callback is called when the upload image success
onUploadError(CLErrorResponse error)	this callback is called when the upload image fail/error

3.6.1.3 ICLDownloadHandler

The I**CLDownloadHandler** service class mainly used to get transaction history

This function downloads images in the URL with authentication.

doDownload

The **CLDownloadHandler** service class mainly used to get transaction history

Table 3-47 ICL DownloadHandler

ICLDownloadHandler	
Methods	**Description**
doDownload(String imageUrl)	this function download image in the URL with authentication

3.6.1.4 ICLDownloadService

The I**CLDownloadService** interfaces has methods/callbacks:

This callback is called to get the image when download is finished.

onDownloadImageSuccess

This callback is called when image can't be download

onDownloadImageError

The **ICLDownloadService** interfaces has method/callback;

Table 3-48 ICLDownloadService

ICLDownloadService	
Methods	**Description**
onDownloadImageSuccess(CLDownloadImageResponse response)	this callback to get the image when download is finished
onDownloadImageError(CLErrorResponse error)	this callback is called when image can't be download

3.6.2 Send Receipt

The service is used to send receipt payment transactions. The receipt is sent by cashlez's e-mail or SMS. Below is Send Receipt flow (Figure 3.11).

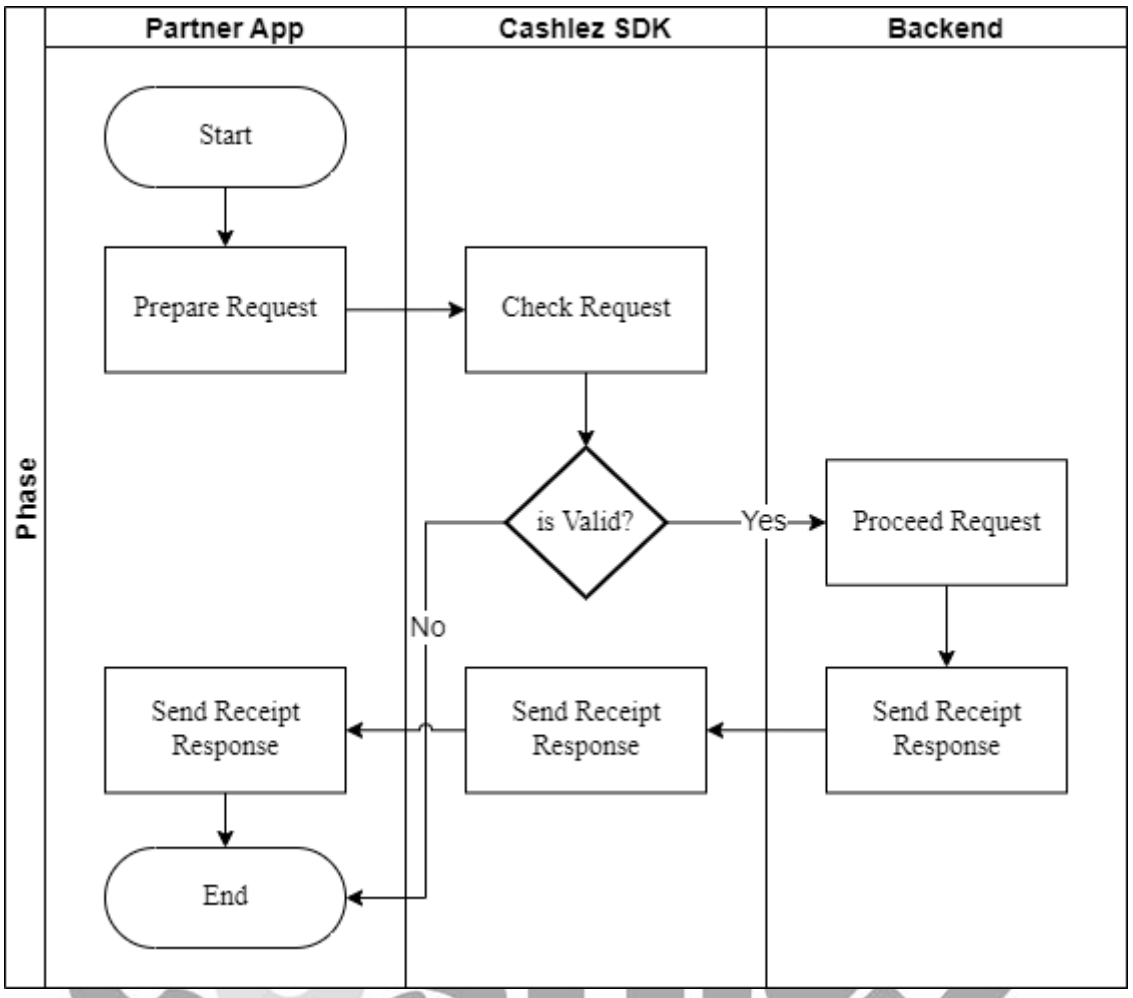

Figure 3.11 Send Receipt Flow

The service is used to send receipt payment transactions. the receipt sent by cashlez's e-mail or SMS.

3.6.2.1 ICLSendReceiptHandler

The **ICLSendReceiptHandlerservice** class to send receipt.

This function to send receipt.

doSendReceipt

The **ICLSendReceiptHandler** service class to send send receipt.

Table 3-49 ICLSendReceiptHandler

ICLSendReceiptHandler	
Methods	**Description**
doSendReceipt(CLPaymentResponse response)	this function used to send receipt

3.6.2.2 ICLSendReceiptService

The I**CLSendReceiptServiceinterfaces** has methods/callbacks:

This callback is called when send receipt success

onSendReceiptSuccess

This callback is called when send receipt failed

onSendReceiptError

The **ICLSendReceiptService** interfaces has methods/callbacks;

Table 3-50 ICLSendReceiptService

ICLSendReceiptService	
Methods	**Description**
onSendReceiptSuccess(CLSendReceiptResponse response)	this callback is called when send receipt success
onSendReceiptError(CLErrorResponse error)	this callback is called when send receipt fail/error

3.6.3 Help Message

The service is used when customers need some help and send messages to cashlez. Below is Help Message flow (Figure 3.12).

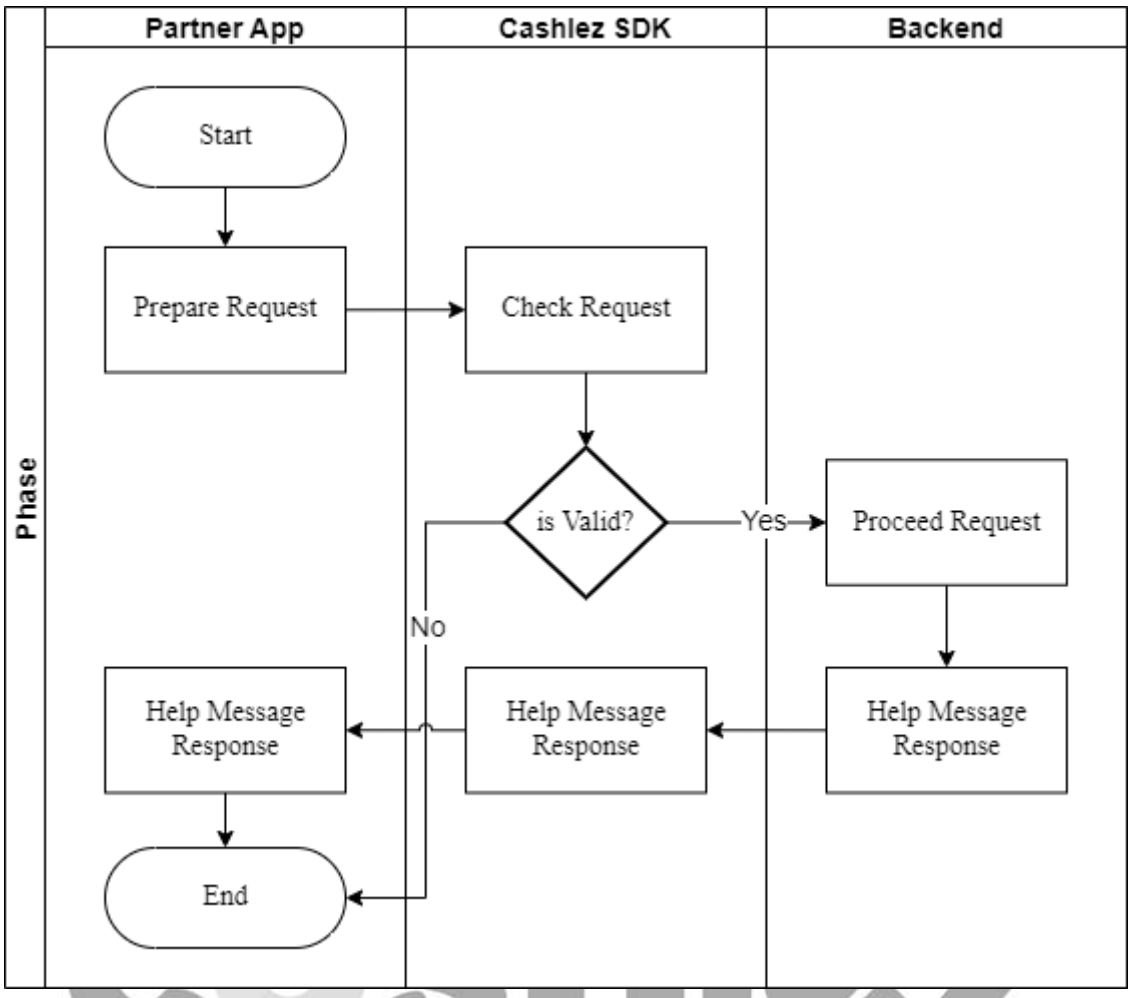

Figure 3.12 Help Message Flow

The service is used when customers need some help and send messages to cashlez.

3.6.3.1 ICLHelpHandler

The I**CLHelpHandlerservice** class mainly used to check the reader.

This function to send help message to cashlez.

doSendMessage

The **ICLHelpHandler** class mainly used to check the reader.

Table 3-51 ICLHelpHandler

ICLHelpHandler	
Methods	**Description**
doSendMessage	this function to send help messages to Cashlez.

3.6.3.2 ICLHelpMessageService

The **ICLHelpMessageServiceinterfaces** has methods/callbacks:

This callback is called when the result of the help message is available.

onSendHelpSuccess

This callback is called when help message failed

onSendHelpError

The **ICLHelpMessageService** interfaces has methods/callbacks;

Table 3-52 ICLHelpMessageService

ICLHelpMessageService	
Methods	**Description**
onSendHelpSuccess	this callback is called when send help message success
onSendHelpError	this callback is called when send help message fail/error

3.7 Response Code

Below are the response codes from our SDK (Table 3.11).

Table 3.53 Response Code

No.	Response Code	Message
1.	1001	Please fill Username and PIN
2.	1002	Please fill Username
3.	1003	Please fill PIN
4.	1004	Username must be more than 3 characters in length
5.	1005	PIN must be 6 characters in length
6.	1006	Username and Pin too short
7.	1007	Aggregator login data is needed
8.	1008	Server public key is needed
9.	1009	Aggregator id is needed
10.	1010	Please fill activation code
11.	1011	Fail handshake, please try again
12.	1012	Fail to decrypt process
13.	1013	Please provide valid reader companion
14.	1014	Please fill message
15.	1015	Please provide valid image path
16.	1016	Upload image failed
17.	1017	Image already exist
18.	1018	Transaction Id required
19.	1019	Download image failed
20.	1020	Please provide valid payment data
21.	1021	Location Service is not available
22.	1022	Please update Location Service to continue the process
23.	1023	Please provide valid signature

24.	1024	Amount is not valid
25.	1025	Please enable GPS
26.	1026	Please wait, updating location
27.	1027	Please provide transaction type
28.	1028	No reader compainon paired
29.	1029	You don't have Printer paired
30.	1030	Bluetooth off
31.	1031	Connect to printer failed
32.	1032	Printer off
33.	1033	Printer overheat
34.	1034	Paper empty
35.	1035	Please try again
36.	1036	Printer battery low
37.	1037	Please provide verification mode
38.	1038	You're not connecting with your Reader companion, only CASH Transaction can proceed
39.	1039	Waiting for reader
40.	1040	Failed get companion serial number, check your companion
41.	1042	Reader not connected
42.	1043	Reader connection fail to start
43.	1044	Reader waiting time out
44.	1045	Transaction cancelled
45.	1046	Error while processing
46.	1047	Card expired
47.	1048	Card data not valid
48.	1049	Transaction declined

49.	1050	Reader not activated
50.	1051	Transaction failed
51.	1052	Password is mandatory
52.	1053	User data is mandatory
53.	1062	Please fill old PIN and new PIN
54.	1063	Please fill old PIN
55.	1064	Please fill new PIN
56.	1065	Old PIN must be 6 characters in length
57.	1066	New PIN must be 6 characters in length
58.	1067	You can't do settlement
59.	1068	Merchant Transaction Id required
60.	1069	Mobile number required
61.	1070	Please provide valid printer companion
62.	1071	Client private key is needed
63.	1072	Payment Status Not Valid
64.	1073	Printer Disconnect
65.	1074	Printer Error
66.	1075	Please Provide valid QR
67.	1076	Reader Error
68.	1077	Reader Companion Disconnect
69.	1078	Reversal Before Void Error
70.	1079	Login Process Failed
71.	1080	Only send email receipt
72.	1081	Payment Only send receipt, but email not valid
73.	1082	Payment only send receipt, but email empty
74.	1083	Email format not valid

75.	1084	Email and Phone number not valid
76.	1085	Phone number not valid
77.	1086	Email or Phone number is required
78.	1087	Reader process interrupter
79.	1088	GPN is not Enabled
80.	1089	Please provide card processing mode
81.	1090	Please check your transaction history
82.	1091	Phone number is required
83.	1101	Auto transfer mode must not have beneficiary / destination account and bank code filled
84.	1102	Manual transfer mode must not have beneficiary / destination account and bank code filled
85.	1103	Invalid or Empty Bank Code
86.	1104	Invalid or Empty Bank Account Number
87.	1054	Email, username and image path required
88.	1105	Transfer Detail is missing
89.	1106	Transfer Detail is tampered
90.	1107	Debit transfer cancelled
91.	1110	Product installment required
92.	1164	Mandiri Pay Response Required
93.	1165	Gopay QR Response Required
94.	1166	Kredivo Response Required
95.	1167	Dana QR Response Required
96.	1168	Shopee QR Response Required
97.	1169	Transaction can't be voided, Paid using other wallet
98.	1170	Virtual Account Response Required
99.	1171	To use the payment method minimum amount 10 rbu and max 25jt

100.	1172	To use the payment method minimum amount 10000 and maximum 25000000
101.	1056	Email and image path required
102.	1057	mail required
103.	2001	Fail to response, please try again
104.	2002	Session is expired
105.	2003	TLE LTWK key download error
106.	2004	TLE Logon download error
107.	2012	Page number is invalid
108.	2013	Transaction Terminated
109.	2014	Timeout
110.	2015	Transaction capk failed
111.	2016	Transaction not ICC
112.	2017	Transaction App Fail
113.	2018	Transaction Device Error
114.	2019	Transaction Application Blocked
115.	2020	Transaction ICC Card Removed
116.	2021	Transaction Card Blocked
117.	2022	Transaction Card not Supported
118.	2023	Transaction Condition not satisfied
119.	2024	Transaction Invalid ICC Data
120.	2025	Transaction missing mandatory data
121.	2026	Transaction no EMV Apps
122.	2027	Declined
123.	2028	Swipe not allowed insert card
124.	2029	Wrong length

125.	2030	PIN Timeout
126.	2031	PIN Canceled
127.	2032	No Card Detected
128.	2033	Card Inserted
129.	2034	Bad Swipe
130.	3010	You have exceeded a maximum number of three (3) attempts. Please contact your Merchant System Administrator
131.	3011	You have exceeded a maximum number of five (5) attempts. Please contact your Merchant System Administrator
132.	3012	You are not authorized to void or settle transactions
133.	3020	Please activate account using another phone /device
134.	3021	Invalid Reader
135.	3022	Please use the same Smart Reader
136.	3023	Invalid phone ID. Please reset your Smart Reader
137.	3030	Reader is not linked to the current merchant
138.	3031	Reader is inactive or suspended. Please insert another reader
139.	3032	Reader malfunction. Please contact our Merchant Hotline for replacement
140.	3040	TID is suspended or not linked to Mobile User
141.	3042	No TID is linked with this mobile user
142.	3043	Application Expired, please update the application
143.	3044	New version is available, please update the application
144.	5010	Invalid login, please try again or contact your Merchant System Administrator
145.	5011	User PIN must be 6 numeric characters
146.	5012	Please do not reuse the last 5 passwords
147.	5013	Invalid activation code. Please try again
148.	5014	Please ensure User ID and User PIN are valid. This will be your last

		attempt before your account is suspended
149.	5015	User is not active
150.	5016	Activation failed
151.	5017	Mobile user already exists with that name
152.	5020	You are using an outdated application. Please update your version
153.	5030	Unable to find resource you\'re looking for
154.	5031	Password must have 6 numbers
155.	5032	Old password must be different with new password
156.	5033	New password already used before
157.	5034	Wrong password when voiding
158.	5035	You are not authorized to void transactions
159.	5036	Void failed because this user is suspended
160.	5037	Settlement failed because this user is suspended
161.	5038	Invalid format user login. User login can contain alphanumeric, \'.\' (dot), \'-\' (dash), \'_\'(underscore)
162.	5039	Wrong password when settlement
163.	5040	You are not authorized to settle this batch
164.	5041	Failed to do settlement, Kindly contach our merchant hotline
165.	5042	Application Settlement Required
166.	5043	Application Transaction not found
167.	3041	Failed to do settlement, kindly contact our Merchant Hotline
168.	3042	Batch is full, please settle
169.	3043	Unable to find transaction you\'re looking for
170.	5110	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
171.	5111	You have exceeded your daily transaction limit. Please contact our Merchant Hotline

172.	5112	You have exceeded your monthly transaction limit. Please contact our Merchant Hotline
173.	5113	You have exceeded your transaction limit. Please contact our Merchant Hotline
174.	5114	Please verify mobile number
175.	5115	Please verify email
176.	5116	Email or SMS service is currently unavailable. Please contact Merchant Hotline
177.	5117	Your transaction is not allowed by risk management. Please contact our Merchant Hotline
178.	5118	Unable to process payment. Host keys not properly configured
179.	5119	Invalid template SMS
180.	5120	Error while saving data to table
181.	5121	Error while saving data to table
182.	5122	You cannot perform transaction outside permitted location
183.	5123	Your transaction is below than limit per transaction
184.	5124	Your transaction currency is not supported
185.	5125	Transaction amount mismatch between EMV amount and service amount
186.	5126	Transaction is already reversed
187.	5127	No TID supported for current transaction
188.	5128	Merchant disallowed magstripe and signature verification. Please contact support
189.	5129	No aggregator supported for current transaction
190.	5130	Invalid request URL
191.	5131	Card not supported for current transaction
192.	5555	System is currently not available. Please try again later
193.	5600	Transaction must use PIN
194.	5601	Wrong choice of transaction type: please use credit transaction

195.	5602	Wrong choice of transaction type: please use debit transaction
196.	5603	Incorrect PIN
197.	5604	Duplicate Transaction
198.	5605	Application Transaction already Approve
199.	5606	Application Invalid Card
200.	8090	An error has occurred. Please contact our Merchant Hotline
201.	8091	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
202.	8092	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
203.	8093	Batch Upload failed. Please call Help Desk
204.	8094	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
205.	9001	Invalid card
206.	9010	Invalid service name/version
207.	9011	Method invocation error
208.	9012	No Application ID is selected
209.	10001	Service is currently unavailable. Please try again, if the problem persists kindly contact our Merchant Hotline
210.	11001	Reader ID in session and request don't match
211.	11002	Reader ID does not exist in the concurrent map
212.	12001	Connection between client and host expired, due to cancellation or timeout
213.	12002	Maximum thread limit reached
214.	12003	Thread interrupted in long poller, probably triggered by a forced destroy
215.	13001	Error during encryption/decryption
216.	13002	Error, client disconnected
217.	14001	Connection timed out

218.	14002	Login token could not be created
219.	14003	Login token could not be found or found to be mismatched
220.	14004	Login token expired.
221.	15001	Problem in receiving help message
222.	16001	Requested data is unavailable, if the problem persists kindly contact our Merchant Hotline
223.	16002	State of requested data is invalid, please contact our Merchant Hotline