ANDROID SDK SUNMI DOCUMENTATION
PT. CASHLEZ WORLDWIDE INDONESIA, Tbk

[Cashlez External]

Approved by: Product Owner
Version: 2.0.3.7.4
Classification: External Use
Date: 14 July, 2022

DOCUMENT INFORMATION

Document Name : Android SDK SUNMI Document v2.0.3.7.4

Document Status : Final

Detail Status

Release Date	May 11, 2022

Document Version History

FSD Version No.	Date	Content	Modified By
2.0.3.7.4	13 July 2022	• New Payment Card mock	Julian Natalino
	18 July 2022	• Finalization	Nathania Oey

Document Control

Role	Name	Division
Reviewed by	Nathania Oey	IT Compliance + TW Manager
Maintained by	Julian Natalino	IT Compliance + TW
Document Owner	Juansyah	Product Manager

Table of Contents

1. Introduction

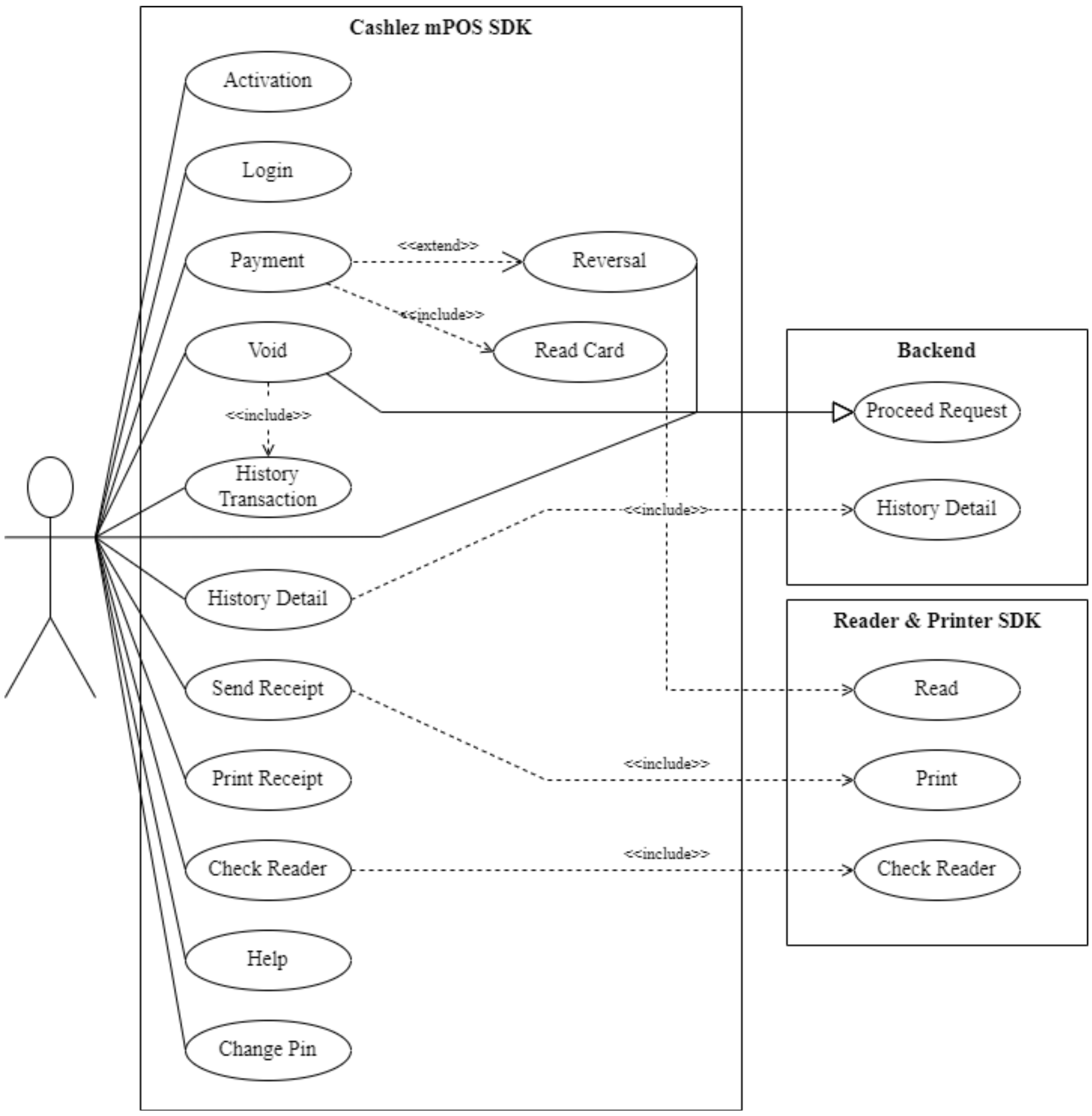


Figure 1.1 SDK Use Case Diagram

1.1. Summary

Sunmi - Cashlez SDK 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 Sunmi SDK.

The following document describes the SDK integration mechanism for third party apps to use Sunmi - Cashlez SDK library and accept payment and how to install Cashlez SDK for Sunmi in order to accept payments in your Sunmi device. 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 Sunmi – Cashlez SDK 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 SunmiAllInOne (C1)

1.4. Versions

Table 1-1 Documentation Versions

2.0.3.7.4	● New Payment Card Mock

2. Sample App/Code

2.1 Summary

This Sunmi – Cashlez SDK documentation includes an example app on how to use and the best practice of using the Sunmi 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, Android SDK and Sunmi 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.

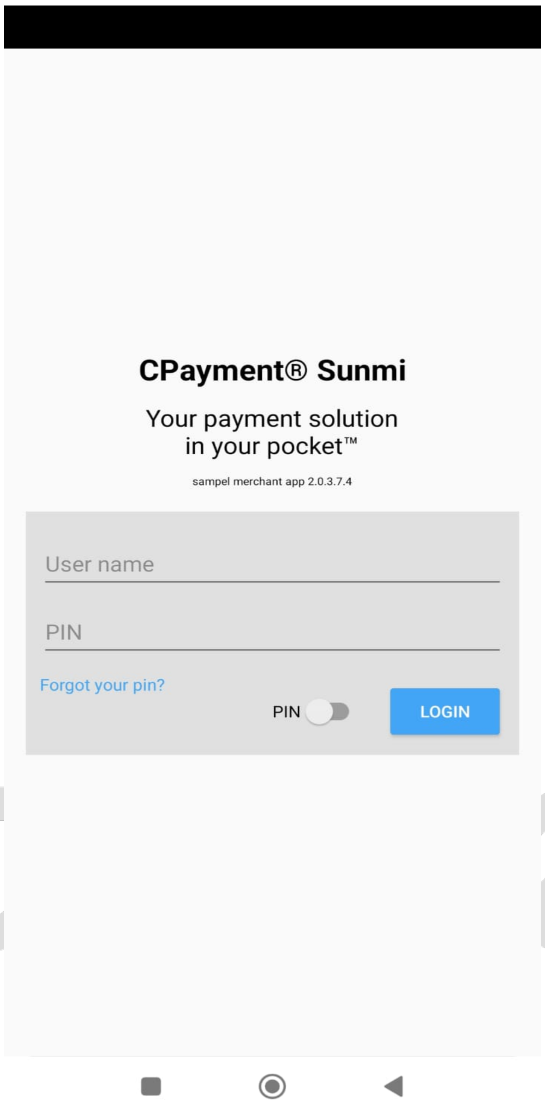

CPayment® Sunmi
Your payment solution in your pocket™
sampel merchant app 2.0.3.7.4
User name
PIN
Forgot your pin?
PIN
LOGIN

Figure 2.4 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 that has given from Cashlez Product Team.
2. Using with Libs name or random name same like **src** folder.
3. Paste Cashlez Lib that has been copied inside Libs or random name.
4. Open your Gradle project, then implement that to the Cashlez Lib SDK inside Gradle Project.

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.

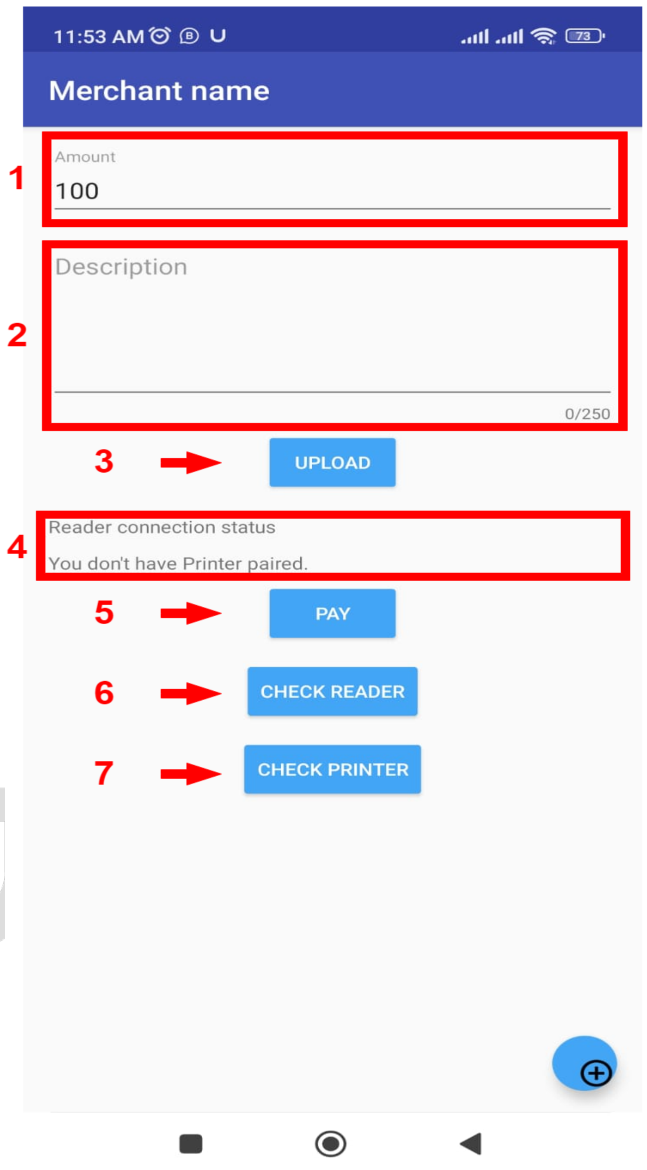


Figure 2-5 Home Page

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

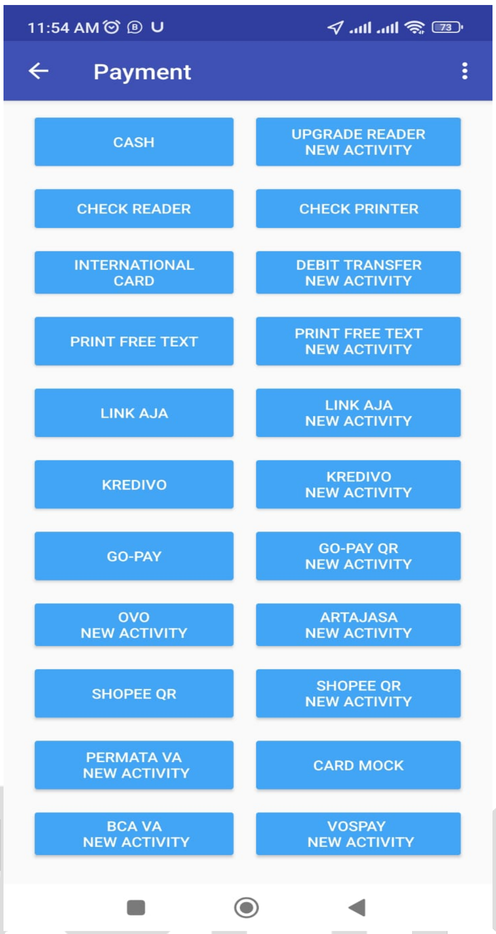


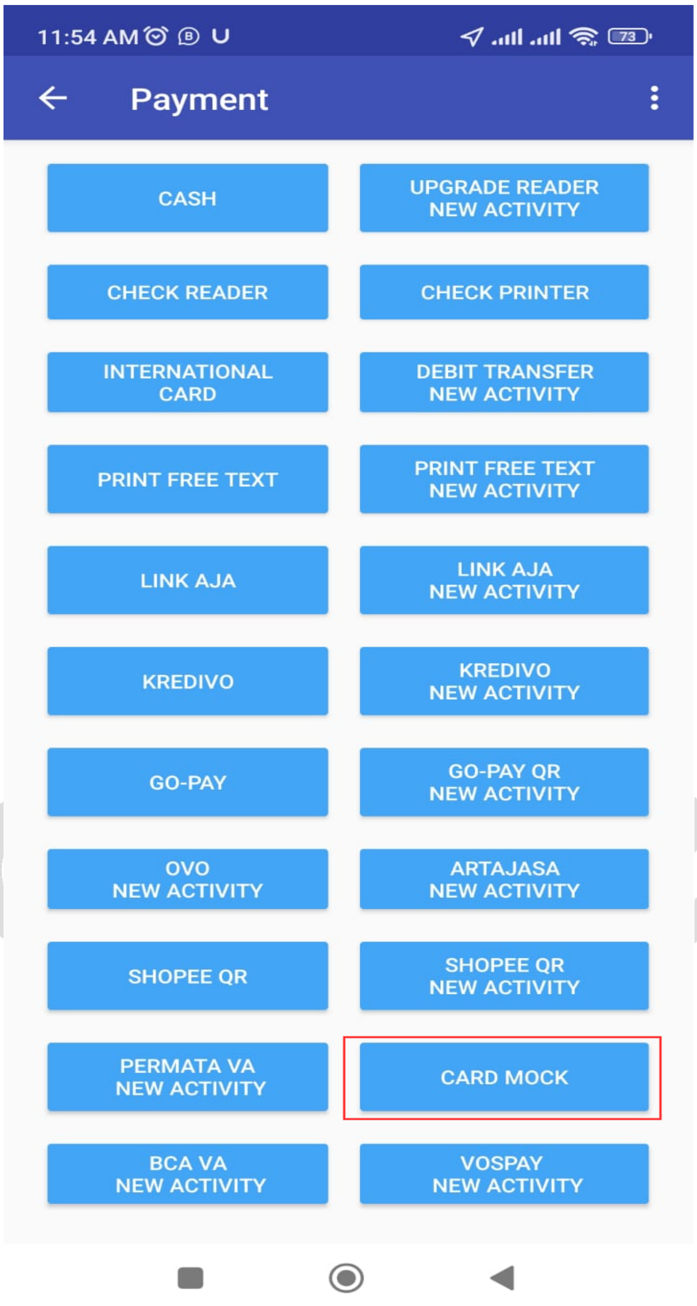
Figure 2-6 Payment Page

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)



On mock card features, user will have capability to test the card reader using any card with chip or magnetic stripe. There are some default amounts for using the card mock:

Table 2-3 Amounts for card mock

Amounts for card mock	
Amounts	**Description**

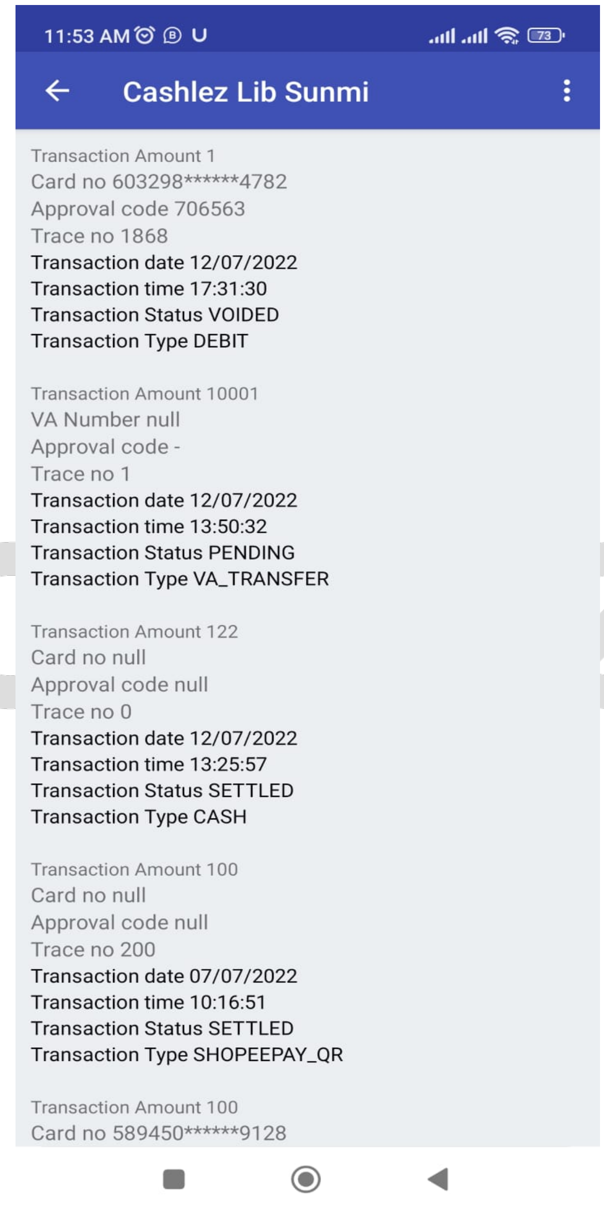
100	Success
50	Decline or rejected
105	PIN error
Other value than above	Batch not ready.

2.5.1 Mock Up Card Transaction

The service is used to create mockup transactions for card payment.

Mock Up Card Transaction		
No.	**Function**	**Description**
1	**CLPaymentHandler**	This function creates mockup transactions. **doProceedCardMock**
2	**CLPaymentService**	The CLPaymentService interfaces has methods/callbacks: 1. This callback is called when a transaction for card payment is successful. **onPaymentSuccess** 2. This callback is called when a transaction for card payment fails. **onPaymentError**
3	**CLPayment Response**	Callback for this function is hardcoded. 1. Success payment from class CLPaymentResponse. Response = Amount 100 2. Failed payment from class CLErrorResponse Response = Amount 50 (Incorrect PIN), Amount 105 (Payment Error). There are 2 options for payments, such as self-service and selected payment method. The differences between self-service and mainstream/selected payment methods are on printing the receipt itself. The receipt can be printed by the user when the payment is

		finished, and of course on this occasion, the user can void the transaction directly without going to payment history details.


11:53 AM

Cashlez Lib Sunmi

Transaction Amount 1
Card no 603298******4782
Approval code 706563
Trace no 1868
Transaction date 12/07/2022
Transaction time 17:31:30
Transaction Status VOIDED
Transaction Type DEBIT

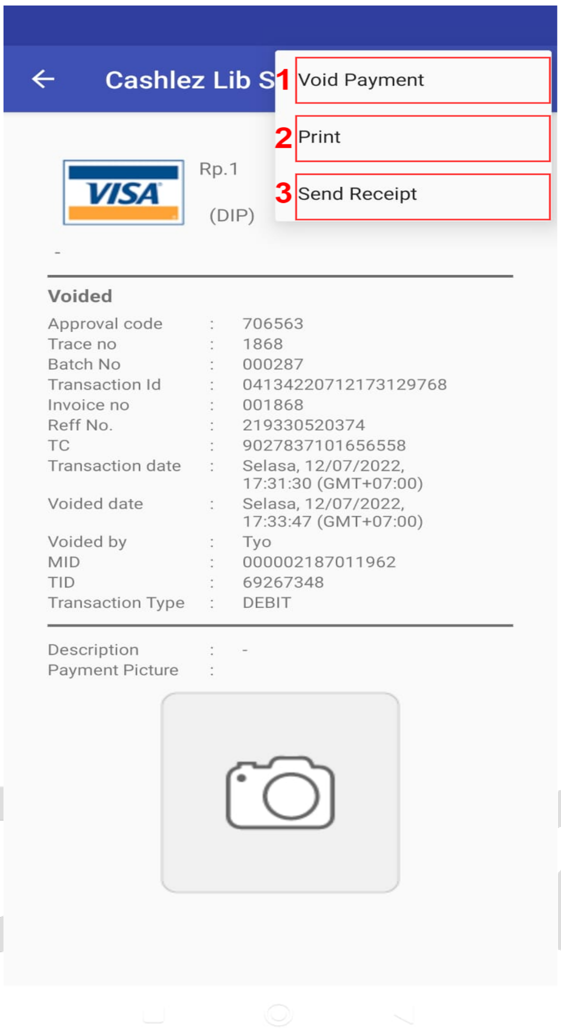
Transaction Amount 10001
VA Number null
Approval code -
Trace no 1
Transaction date 12/07/2022
Transaction time 13:50:32
Transaction Status PENDING
Transaction Type VA_TRANSFER

Transaction Amount 122
Card no null
Approval code null
Trace no 0
Transaction date 12/07/2022
Transaction time 13:25:57
Transaction Status SETTLED
Transaction Type CASH

Transaction Amount 100
Card no null
Approval code null
Trace no 200
Transaction date 07/07/2022
Transaction time 10:16:51
Transaction Status SETTLED
Transaction Type SHOPEEPAY_QR

Transaction Amount 100
Card no 589450******9128

Figure 2-11 Payment History



Rp.1

(DIP)

-

Voided

Approval code	:	706563
Trace no	:	1868
Batch No	:	000287
Transaction Id	:	04134220712173129768
Invoice no	:	001868
Reff No.	:	219330520374
TC	:	9027837101656558
Transaction date	:	Selasa, 12/07/2022, 17:31:30 (GMT+07:00)
Voided date	:	Selasa, 12/07/2022, 17:33:47 (GMT+07:00)
Voided by	:	Tyo
MID	:	000002187011962
TID	:	69267348
Transaction Type	:	DEBIT
Description	:	-
Payment Picture	:	

Figure 2-12 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 SUNMI Device.
2. Turn on Location Service.
3. Bluetooth between SUNMI reader and/or printer. The SDK will automatically find and use one reader and printer available in the Bluetooth paired list.
4. Create a libs folder in your application package, paste the SDK library (AAR) provided / updated version into the libs folder. in Figure 3.1 (FOLLOWING)
5. Implement the SDK library (AAR) into your app's gradle build. Examples like this:

```
implementation(name: 'cashlez-productionallinoneRelease-2.0.3.7.4', ext: 'aar')
```

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. CLLoginResponse

Table 3-1 CLLoginResponse

CLLoginResponse		
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	GO-PAY
KREDIVO_QR	KREDIVO_QR
SHOPEEPAY_QR	Payment Shopeepay (QrPayment)
VA_TRANSFER	Payment Virtual Account
VOSPAY	Payment Vospay
CARDMOCK	

3.2.1.3. CLPayment

Table 3-3 CLPayment

CLPayment		
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. CLPaymentResponse

Table 3-4 CLPaymentResponse

CLPaymentResponse		
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. CLErrorResponse

Table 3-9 CLErrorResponse

CLErrorResponse	
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.

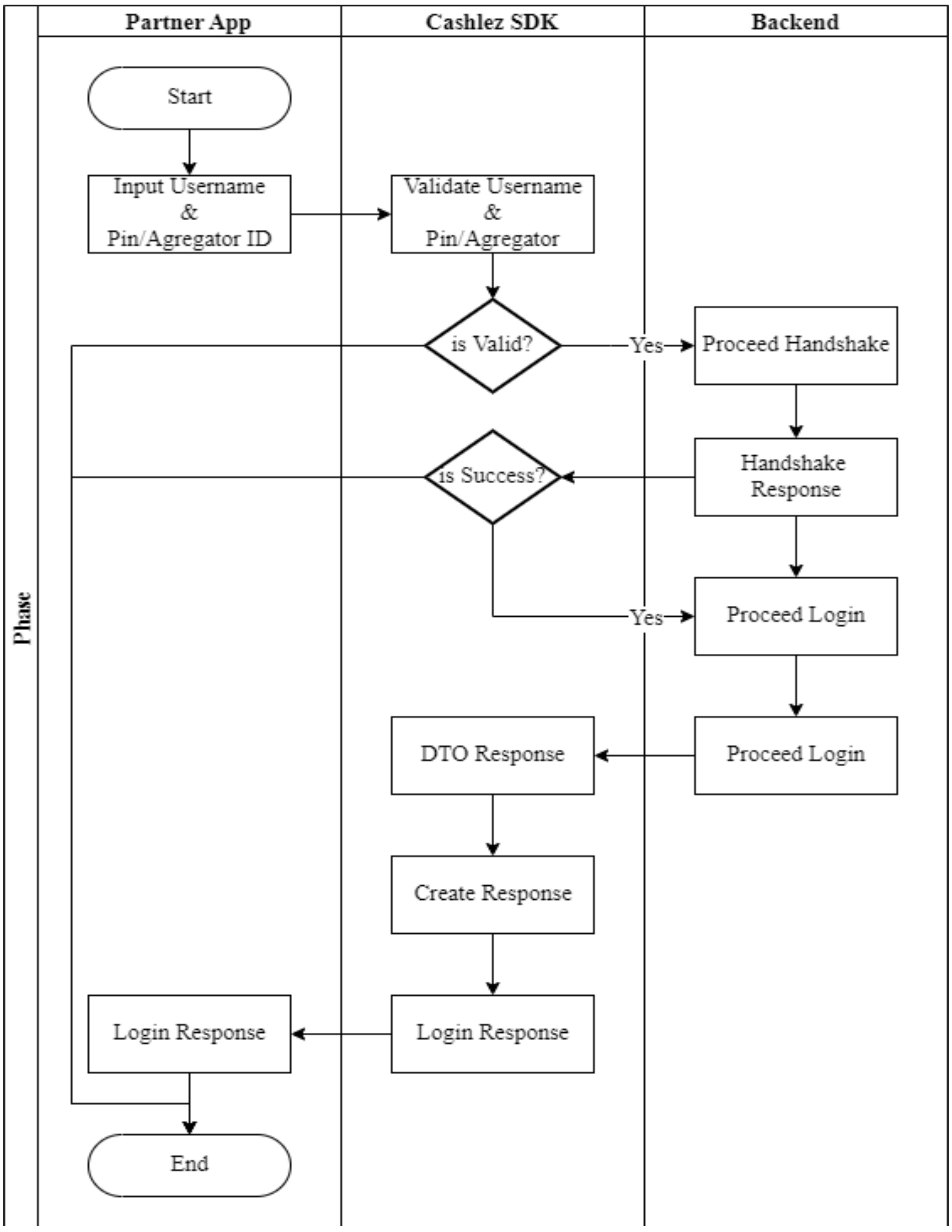


Figure 3.1 Login Flow

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 CLLoginHandler

The **CLLoginHandler** 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 3.1 ICLLoginHandler Methods

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

Table 3-10 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

CLLoginService is a protocol provided by CLLoginHandler. 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 3-11 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).

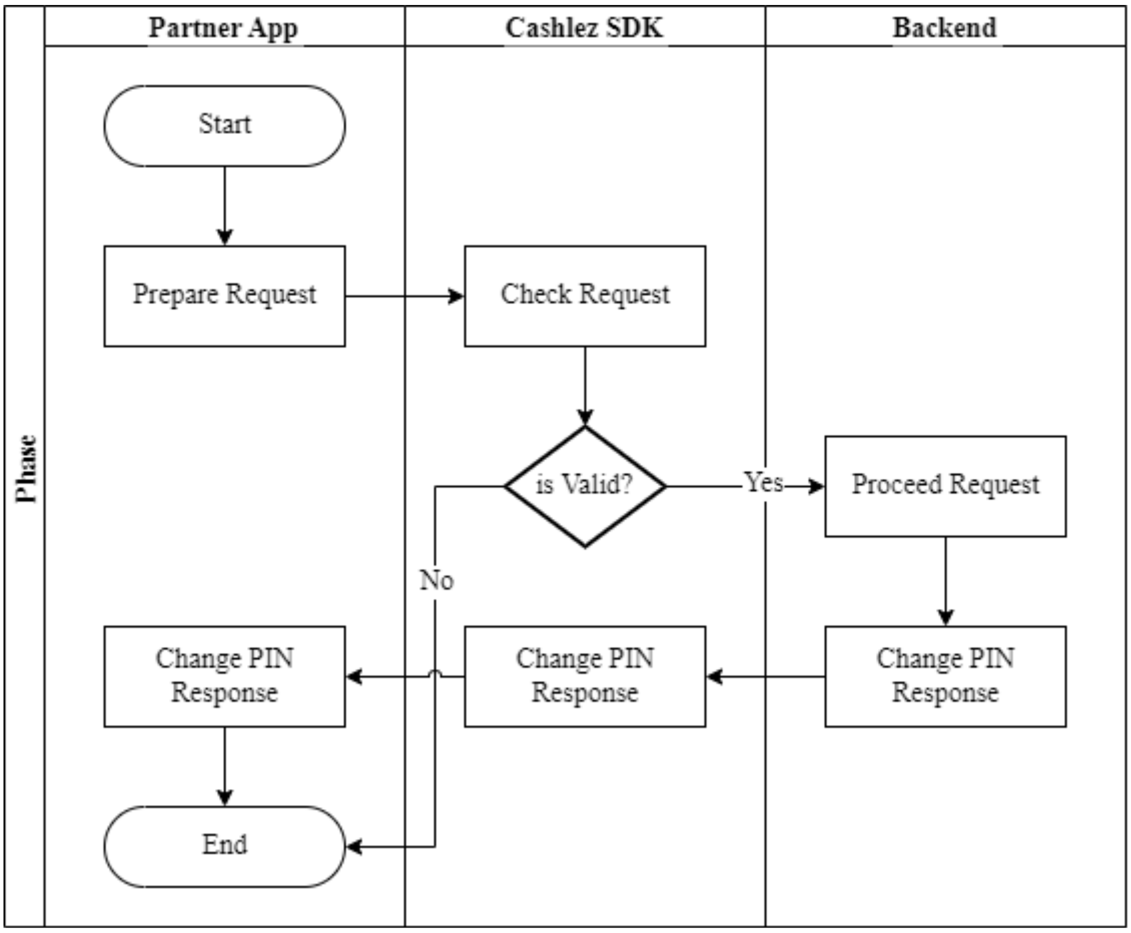


Figure 3.2 Forgot PIN Flow

3.3.2.1 CLManagePasswordHandler

CLManagePassword class main to do forgot pin function and this

doChangePassword this method as execution

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

Table 3-12 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 **CLManagePasswordHandler**. 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 **CLManagePasswordHandler** before forgot PIN.

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

- When forgot PIN is success

```
onManagePasswordSuccess
```

- When forgot PIN is failed

```
onManagePasswordError
```

Table 3-13 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**

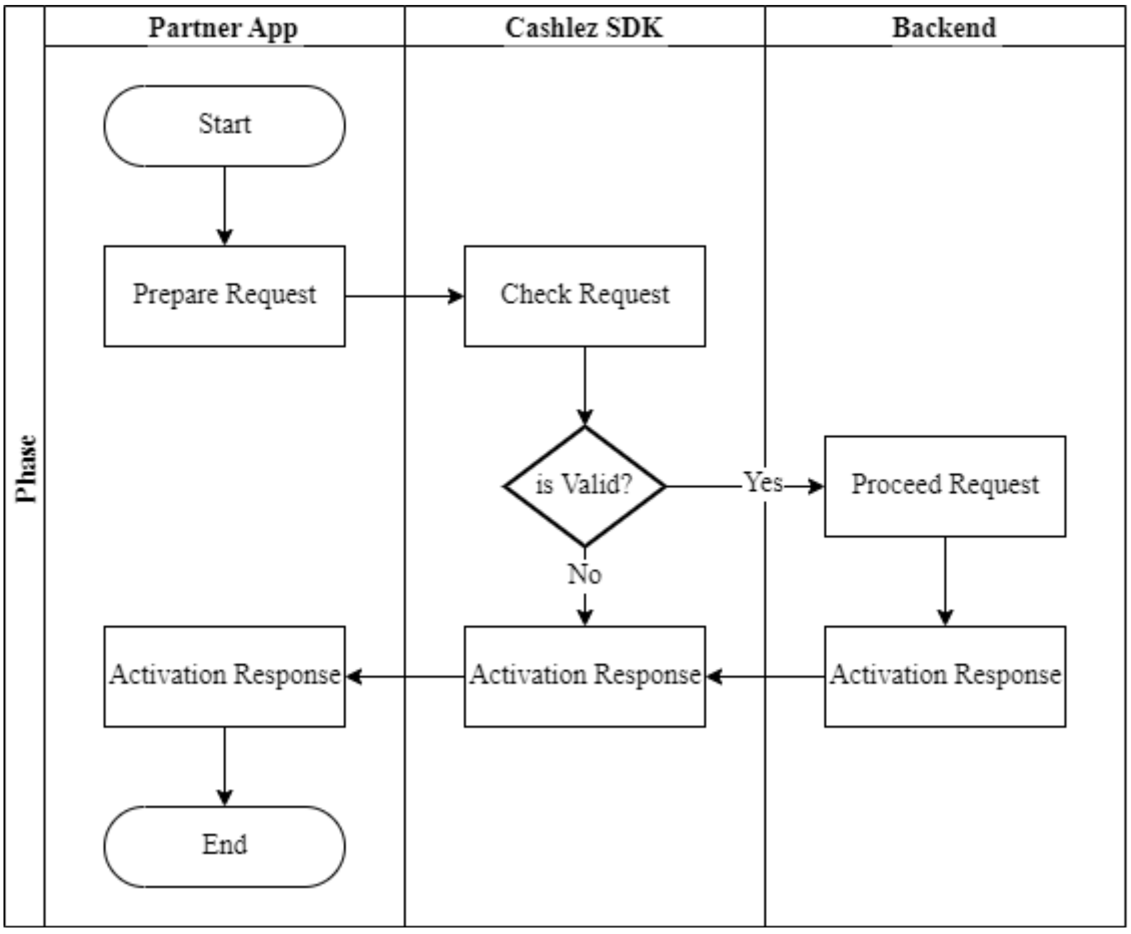


Figure 3.3 Activation Flow

3.3.3.1 CLActivationHandler

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

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

Table 3-14 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 3-15 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

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

C. QRIS

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

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

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

F. 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 I**CLPaymentService** 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.

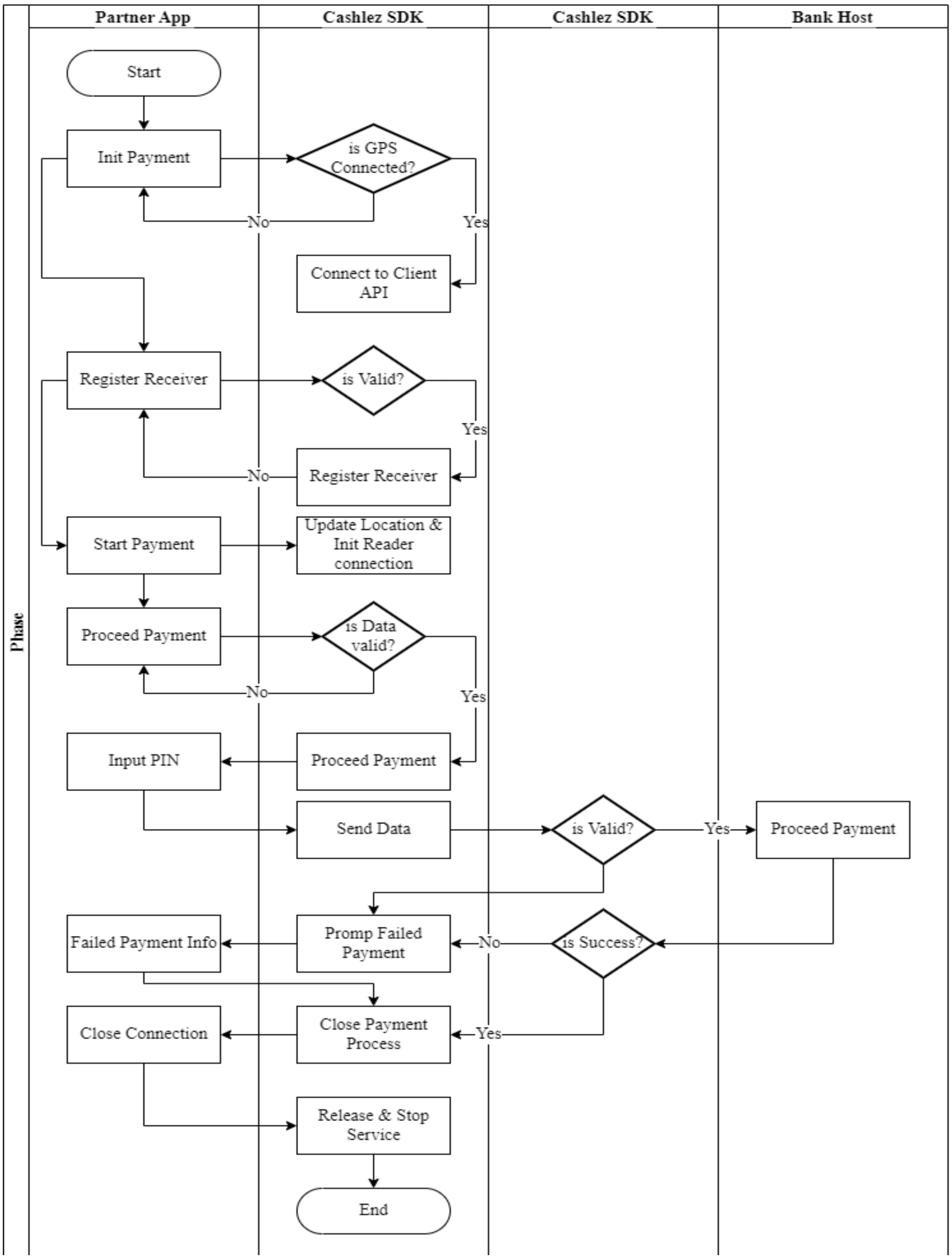


Figure 3.4 Payments Flow

3.4.1.1 CLPaymentHandler

CLPaymentHandler 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 3-16 CLPaymentHandler 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
doProccedCardMock(CLPayment payment);	This function to process Card payment mock mode.
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(CLPaymentpaymer);	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(CLShopeePayQrRe ponse paymentResponse);	To Print Receipt after Payment Success
doProceedShopeePayQr(CLPayment payment	To generate Transaction Shopee Pay QRIS
doInquiryShopeePayQr(CLShopeePayQrResp nse paymentResponse);	To Check Status
doPrintQRContent(Bitmap qrValue);	To Print QRIS Shopee Pay

3.4.1.2 ICLPaymentService

ICLPaymentService is a protocol provided by **CLPaymentHandler**. 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 **CLPaymentHandler** before sending the data. the ICLPaymentService interface has methods/callbacks**.**

Table 3-17 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

3.4.1.3 CLArtajasaVAHandler

CLArtajasaVAHandler 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 3-18 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 **CLArtajasaVAHandler**. 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 **CLArtajasaVAHandler** before sending the data. The ICLArtajasaService interface has methods/callbacks.

Table 3-19 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 CLBcaVaHandler

CLBcaVaHandler 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 3-20 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 **CLBcaVaHandler**. 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 **CLBcaVaHandler** before sending the data. The **ICLBcaVaService** interface has methods/callbacks.

Table 3-21 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 CLPermataVAHandler

CLPermataVAHandler 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 3-22 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 **CLPermataVAHandler**. 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 **CLPermataVAHandler** before sending the data. The **ICLPermataVAService** interface has methods/callbacks.

Table 3-23 ICLPermataVAService

ICLVaService	
Methods	**Description**
onPermataGenerateVASuccess(CLPaymentRespons e 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 CLGoPayQRHandler

CLGoPayQRHandler 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 3-20 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 **CLGoPayQRHandler**. 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 **CLGoPayQRHandler** before sending the data. The **ICLGoPayQRService** interface has methods/callbacks.

Table 3-21 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 CLShopeePayQrHandler

CLShopeePayQrHandler 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 3-20 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(CLPaymentRespons e 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 **CLShopeePayQrHandler**. 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 **CLShopeePayQrHandler** before sending the data. The **ICLShopeePayQrService** interface has methods/callbacks.

Table 3-20 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 CLTcashQRHandler

CLTcashQRHandler 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 3-20 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(CLTCashQRRespons e 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 CLTCashQRHandler. 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 3-20 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 CLVospayHandler

CLVospayHandler 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 3-22 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 CLVospayHandler. 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 **CLVospayHandler** before send the data. the ICLVospayService interfaces has methods/callbacks.

Table 3-23 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 CLOvoHandler

CLOvoHandler 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 3-24 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 CLOvoHandler. 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 3-25 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

This service is used specially for our payment called Cashlez Link. It will generate a link directly to payment. For each callback it will return responses.

Table 3-25 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 CLKredivoHandler

CLKredivoHandler 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 3-25 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 **CLKredivoHandler**. 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 **CLKredivoHandler** before sending the data. The **ICLKredivoService** interface has methods/callbacks**.**

Table 3-25 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

3.4.2 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).

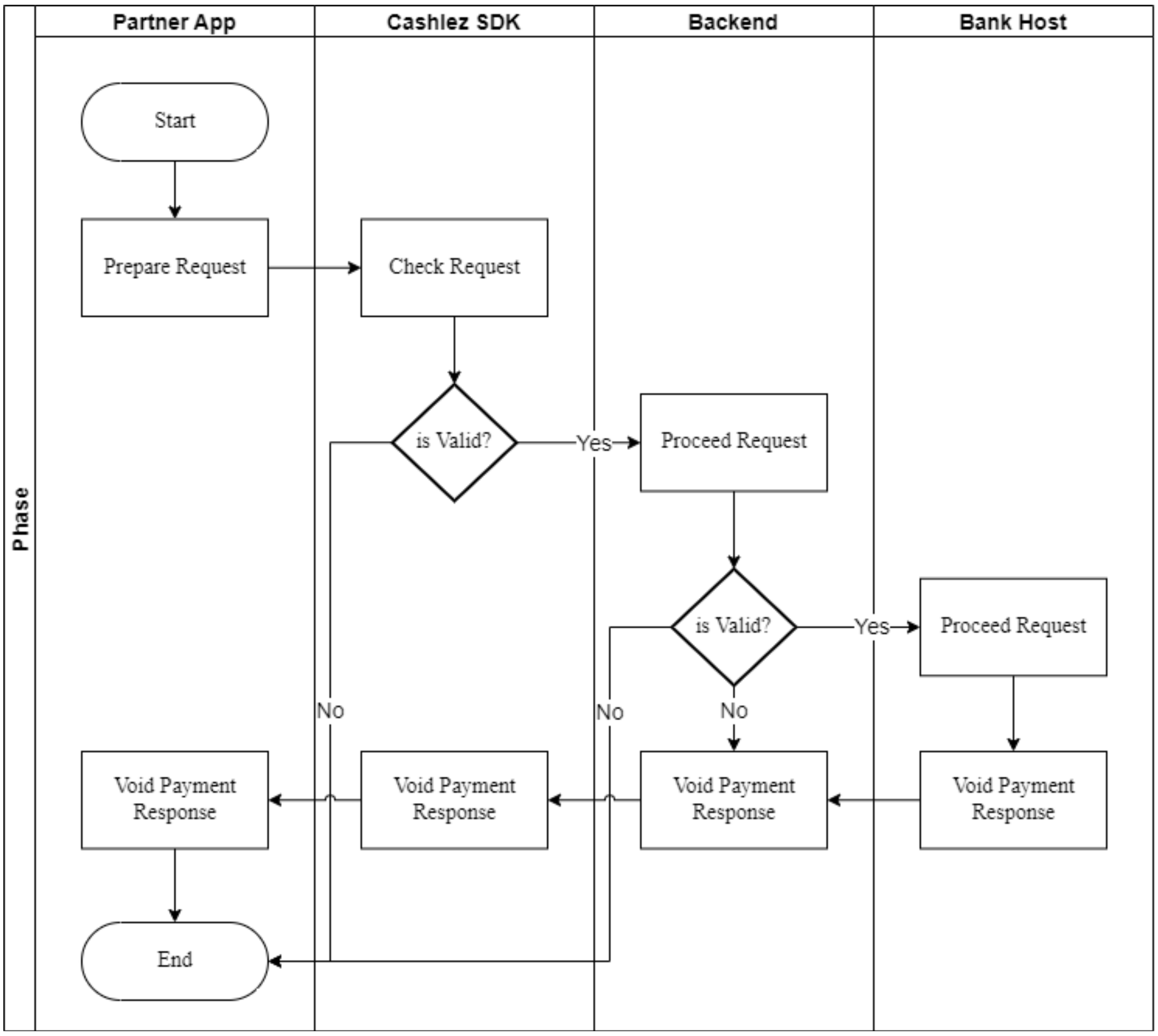


Figure 3.5 Void Payment Flow

3.4.2.1 CLVoidPaymentHandler

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

Table 3-26 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 CLVoidService is a protocol provided by CLVoidPaymentHandler. 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 3-27 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.

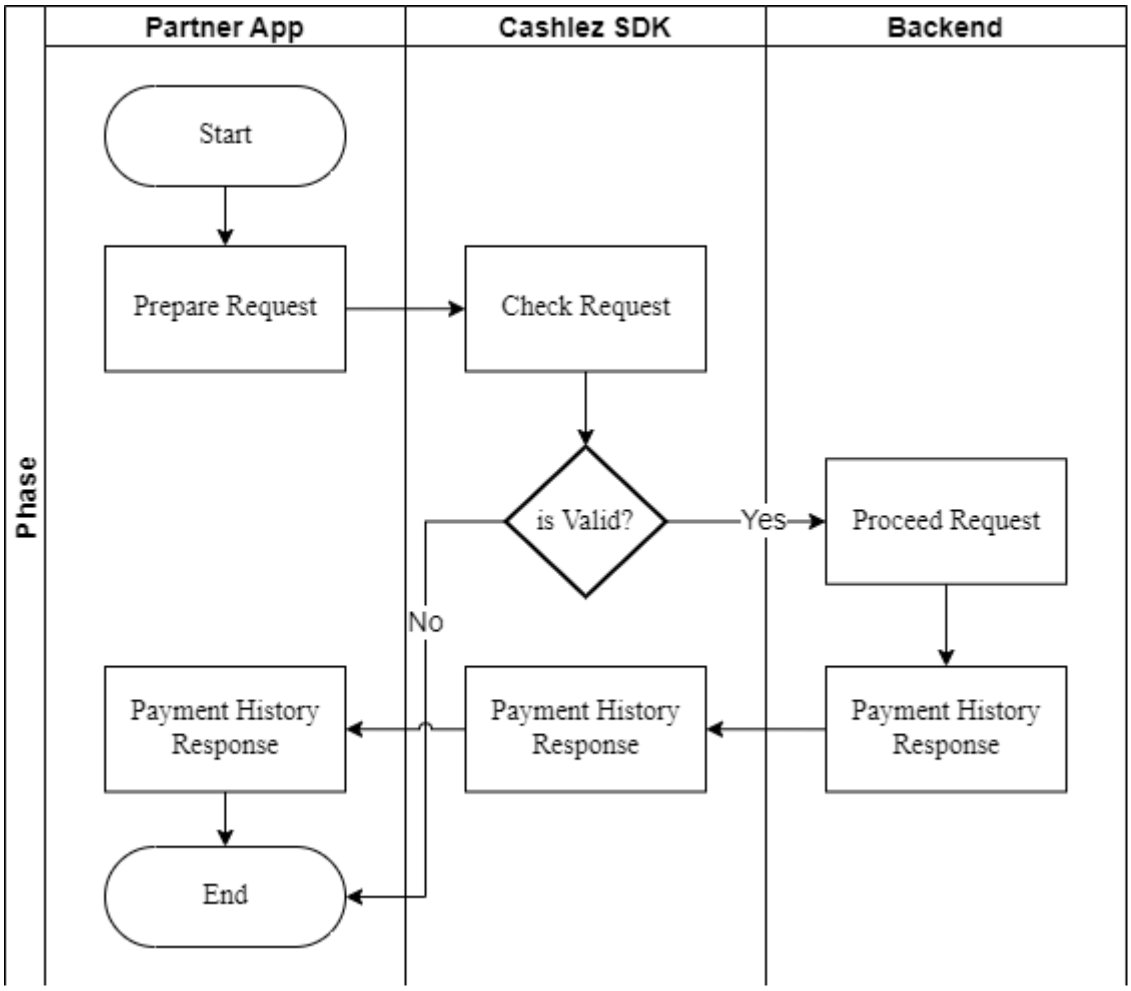


Figure 3.6 Payment History Flow

3.5.1.1 CLPaymentHistoryHandler

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

Table 3-28 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

CLPaymentHistoryService is a protocol provided by CLPaymentHistoryHandler. 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 CLPaymentHistoryHandler before sending the data.

The CLPaymentHistoryService 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 CLPaymentHistoryHandler. 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 CLPaymentHistoryHandler before sending the data. The ICLPaymentHistoryService interface has method/callbacks.

Table 3-29 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).

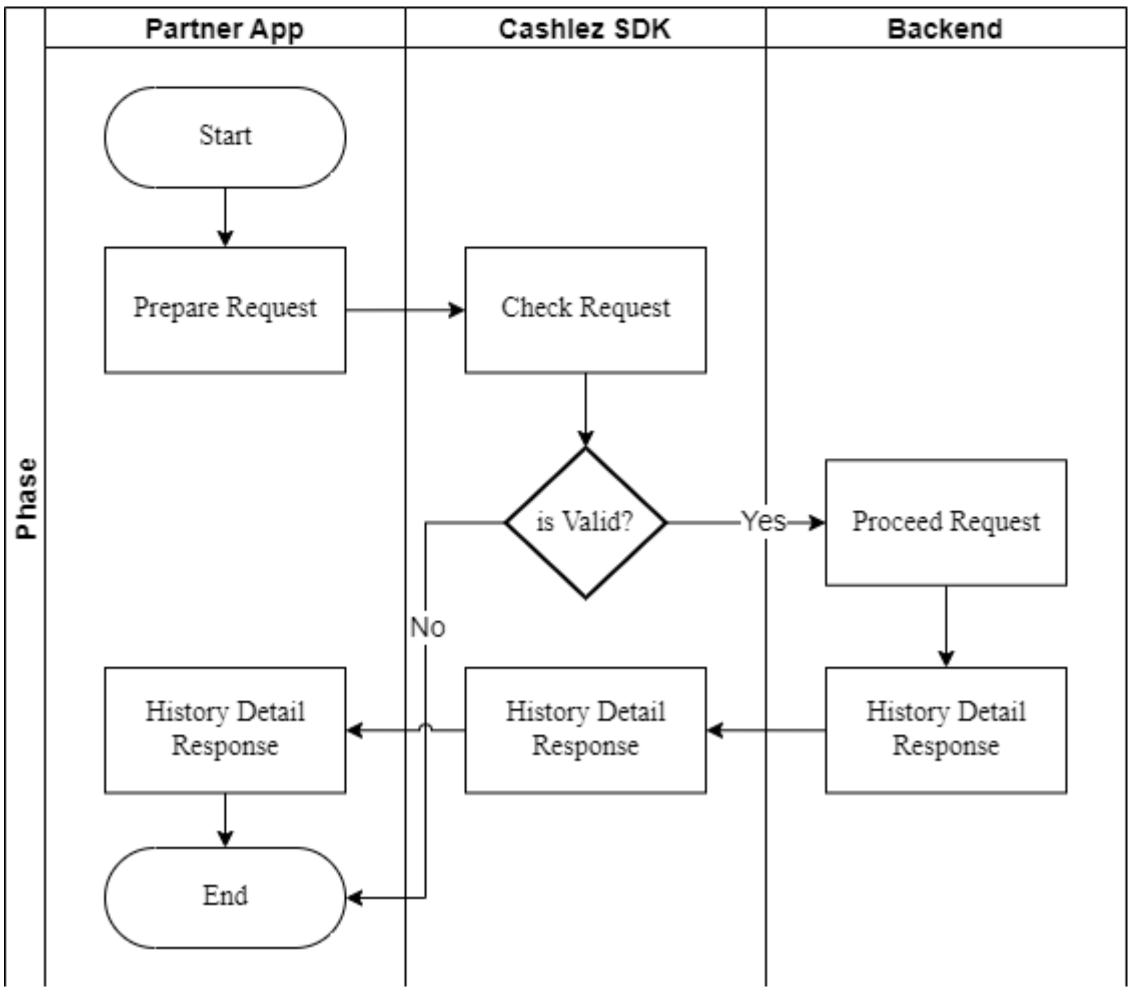


Figure 3.7 Payment History Detail Flow

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 CLPaymentHistoryDetailHandler

CLPaymentHistoryDetailhandler is a class for handling payment history detail requests.

This function gets transaction detail based on transaction identifier

doGetSalesHistoryDetail

CLPaymenetHistoryDetailHandler is a class for handling payment detail requests.

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

3.5.2.2 ICLPaymentHistoryDetailService

CLPaymentHistoryDetailService is protocol provided from CLPaymentHistoryDetailHandler. 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 CLPaymentHistoryDetailHandler before sending the data (Table 3.8).

Table 3.15 ICLPaymentHistoruDetailService Methods

CLPaymentHistoryDetailService	
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 CLUploadHandler

The CLUploadHandler class mainly used to get transaction history

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

doUpload

The CLUploadHandler class mainly used to get transaction history.

Table 3-31 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 CLUploadService 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 3-32 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 CLDownloadHandler

The 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

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

3.6.1.4 ICLDownloadService

The 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 3-33 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).

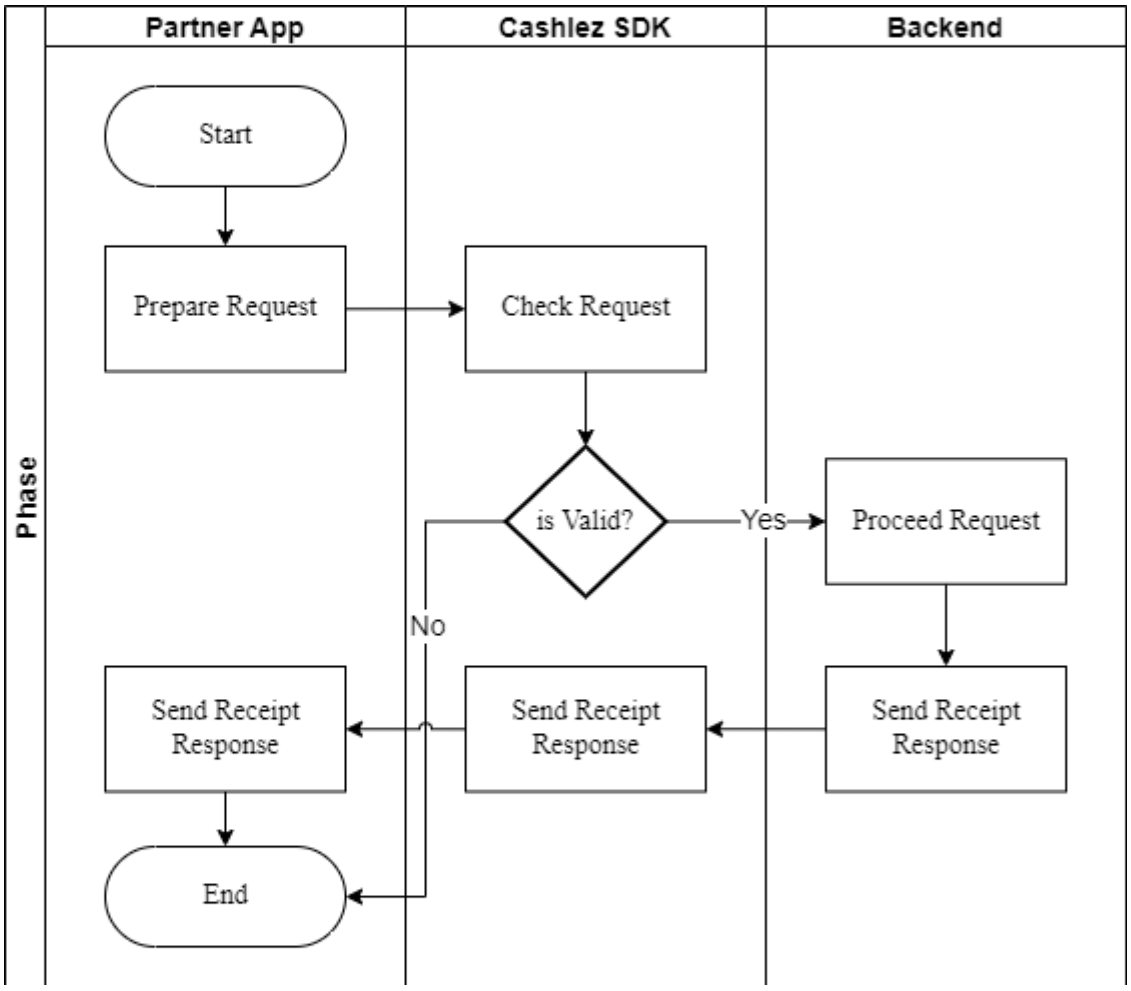


Figure 3.11 Send Receipt Flow

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

3.6.2.1 CLSendReceiptHandler

The CLSendReceiptHandlerservice class to send receipt.

This function to send receipt.

doSendReceipt

The CLSendReceiptHandler service class to send send receipt.

Table 3-34 ICLSendReceiptHandler

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

3.6.2.2 CLSendReceiptService

The 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 3-35 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).

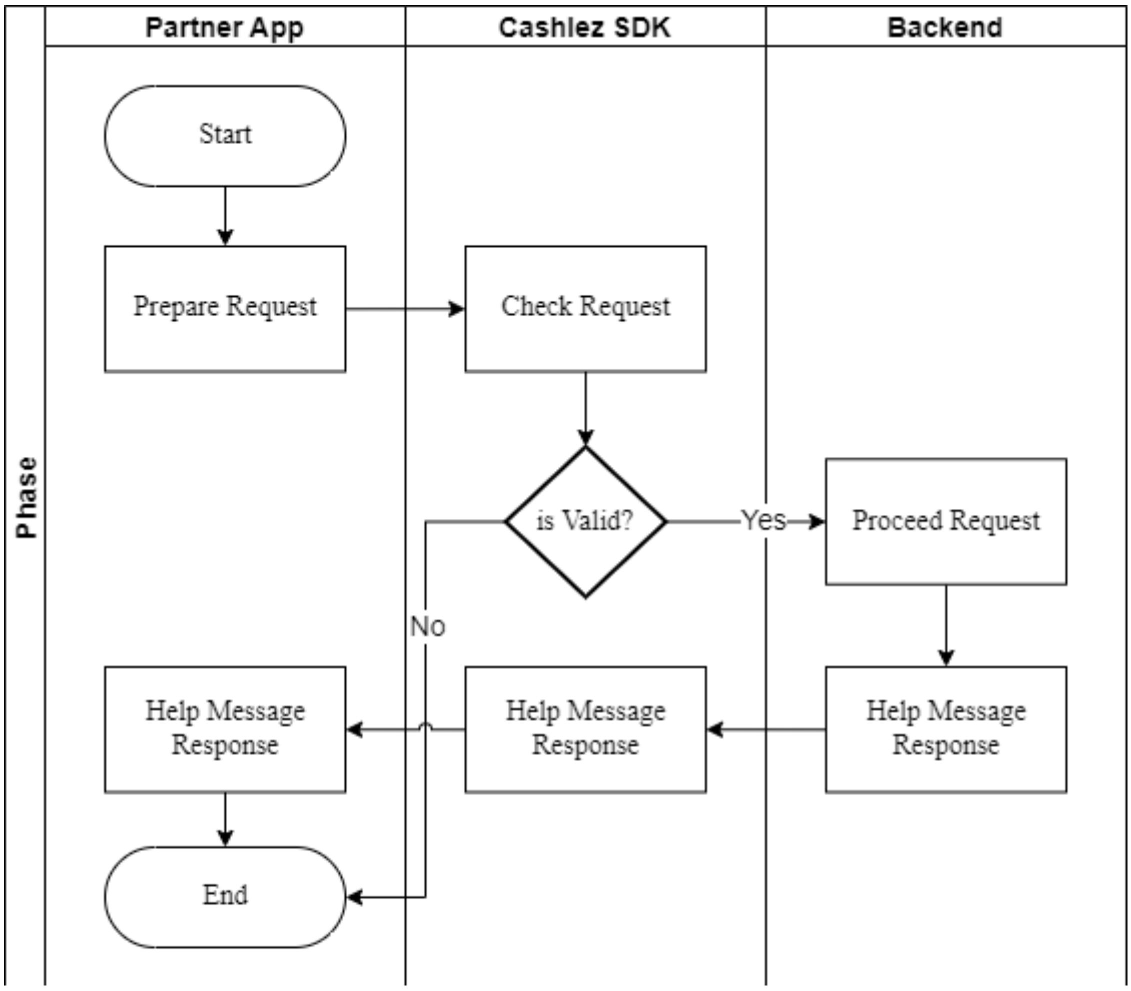


Figure 3.12 Help Message Flow

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

3.6.3.1 CLHelpHandler

The 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 3-36 ICLHelpHandler

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

3.6.3.2 ICLHelpMessageService

The CLHelpMessageServiceinterfaces 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 3-37 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 3.18 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.	10010	Please fill activation code
11.	10011	Fail handshake, please try again
12.	10012	Fail to decrypt process
13.	10013	Please provide valid reader companion
14.	10014	Please fill message
15.	10015	Please provide valid image path
16.	10016	Upload image failed
17.	10017	Image already exist
18.	10018	Transaction Id required
19.	10019	Download image failed
20.	10020	Please provide valid payment data
21.	10021	Location Service is not available
22.	10022	Please update Location Service to continue the process
23.	10023	Please provide valid signature

24.	10024	Amount is not valid
25.	10025	Please enable GPS
26.	10026	Please wait, updating location
27.	10027	Please provide transaction type
28.	10028	No reader compainon paired
29.	10029	You don't have Printer paired
30.	10030	Bluetooth off
31.	10031	Connect to printer failed
32.	10032	Printer off
33.	10033	Printer overheat
34.	10034	Paper empty
35.	10035	Please try again
36.	10036	Printer battery low
37.	10037	Please provide verification mode
38.	10038	You're not connecting with your Reader companion, only CASH Transaction can proceed
39.	10039	Waiting for reader
40.	10040	Failed get companion serial number, check your companion
41.	10042	Reader not connected
42.	10043	Reader connection fail to start
43.	10044	Reader waiting time out
44.	10045	Transaction cancelled
45.	10046	Error while processing
46.	10047	Card expired
47.	10048	Card data not valid
48.	10049	Transaction declined
49.	10050	Reader not activated

50.	10051	Transaction failed
51.	10052	Password is mandatory
52.	10053	User data is mandatory
53.	10062	Please fill old PIN and new PIN
54.	10063	Please fill old PIN
55.	10064	Please fill new PIN
56.	10065	Old PIN must be 6 characters in length
57.	10066	New PIN must be 6 characters in length
58.	10067	You can't do settlement
59.	10068	Merchant Transaction Id required
60.	10069	Mobile number required
61.	10070	Please provide valid printer companion
62.	10071	Client private key is needed
63.	1054	Email, username and image path required
64.	1055	Email and username required
65.	1056	Email and image path required
66.	1057	mail required
67	2001	Fail to response, please try again
68	2002	Session is expired
69	2003	TLE LTWK key download error
70	2004	TLE Logon download error
71	2012	Page number is invalid
72	3010	You have exceeded a maximum number of three (3) attempts. Please contact your Merchant System Administrator
73	3011	You have exceeded a maximum number of five (5) attempts. Please contact your Merchant System Administrator
74	3012	You are not authorized to void or settle transactions

75	3020	Please activate account using another phone /device
76	3021	Invalid Reader
77	3022	Please use the same Smart Reader
78	3023	Invalid phone ID. Please reset your Smart Reader
79	3030	Reader is not linked to the current merchant
80	3031	Reader is inactive or suspended. Please insert another reader
81	3032	Reader malfunction. Please contact our Merchant Hotline for replacement
82	3040	TID is suspended or not linked to Mobile User
83	3042	No TID is linked with this mobile user
84	3043	Application Expired, please update the application
85	3044	New version is available, please update the application
86	5010	Invalid login, please try again or contact your Merchant System Administrator
87	5011	User PIN must be 6 numeric characters
88	5012	Please do not reuse the last 5 passwords
89	5013	Invalid activation code. Please try again
90	5014	Please ensure User ID and User PIN are valid. This will be your last attempt before your account is suspended
91	5015	User is not active
92	5016	Activation failed
93	5017	Mobile user already exists with that name
94	5020	You are using an outdated application. Please update your version
95	5030	Unable to find resource you\'re looking for
96	5031	Password must have 6 numbers
97	5032	Old password must be different with new password
98	5033	New password already used before
99	5034	Wrong password when voiding

100	5035	You are not authorized to void transactions
101	5036	Void failed because this user is suspended
102	5037	Settlement failed because this user is suspended
103	5038	Invalid format user login. User login can contain alphanumeric, \'.\' (dot), \'-\' (dash), \'_\'(underscore)
104	5039	Wrong password when settlement
105	5040	You are not authorized to settle this batch
106	3041	Failed to do settlement, kindly contact our Merchant Hotline
107	3042	Batch is full, please settle
108	3043	Unable to find transaction you\'re looking for
109	5110	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
110	5111	You have exceeded your daily transaction limit. Please contact our Merchant Hotline
111	5112	You have exceeded your monthly transaction limit. Please contact our Merchant Hotline
112	5113	You have exceeded your transaction limit. Please contact our Merchant Hotline
113	5114	Please verify mobile number
114	5115	Please verify email
115	5116	Email or SMS service is currently unavailable. Please contact Merchant Hotline
116	5117	Your transaction is not allowed by risk management. Please contact our Merchant Hotline
117	5118	Unable to process payment. Host keys not properly configured
118	5119	Invalid template SMS
119	5120	Error while saving data to table
120	5121	Error while saving data to table
121	5122	You cannot perform transaction outside permitted location

122	5123	Your transaction is below than limit per transaction
123	5124	Your transaction currency is not supported
124	5125	Transaction amount mismatch between EMV amount and service amount
125	5126	Transaction is already reversed
126	5127	No TID supported for current transaction
127	5128	Merchant disallowed magstripe and signature verification. Please contact support
128	5129	No aggregator supported for current transaction
129	5130	Invalid request URL
130	5131	Card not supported for current transaction
131	5555	System is currently not available. Please try again later
132	5600	Transaction must use PIN
133	5601	Wrong choice of transaction type: please use credit transaction
134	5602	Wrong choice of transaction type: please use debit transaction
135	5603	Incorrect PIN
136	5604	Duplicate Transaction
137	8090	An error has occurred. Please contact our Merchant Hotline
138	8091	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
139	8092	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
140	8093	Batch Upload failed. Please call Help Desk
141	8094	Connection Error. Please try again, if the problem persists kindly contact our Merchant Hotline
142	9001	Invalid card
143	9010	Invalid service name/version
144	9011	Method invocation error
145	9012	No Application ID is selected

146	10001	Service is currently unavailable. Please try again, if the problem persists kindly contact our Merchant Hotline
147	11001	Reader ID in session and request don’t match
148	11002	Reader ID does not exist in the concurrent map
149	12001	Connection between client and host expired, due to cancellation or timeout
150	12002	Maximum thread limit reached
151	12003	Thread interrupted in long poller, probably triggered by a forced destroy
152	13001	Error during encryption/decryption
153	13002	Error, client disconnected
154	14001	Connection timed out
155	14002	Login token could not be created
156	14003	Login token could not be found or found to be mismatched
157	14004	Login token expired.
158	15001	Problem in receiving help message
159	16001	Requested data is unavailable, if the problem persists kindly contact our Merchant Hotline
160	16002	State of requested data is invalid, please contact our Merchant Hotline