API Specification

SDK Development Manual(Android)

Ver:POS2.8.7.180522_R

2018-05-22

Version

EDITION	DATE	REMARK
V1.0	2017/02/25	**First draft**
V2.0	2018/01/03	**1.Added contact common memory card& contactless UL, DesFire interface etc.** **2.Fix Several interfaces**
V2.7.0	2018/01/29	**Optimize the print function**
V2.8.0	2018/02/01	**merge smart pos SDK**
V2.8.2	2018/02/07	**Add api for fingerprint**
V2.8.3	2018/02/07	**Automatic recognition POS Model.**
V2.8.4	2018/03/13	**Support 90POS fingerprint**
V2.8.5	2018/04/27	**1. Remove some unuseful api of fingerprint** **2. Deprecated some api** **3. Add some print api** **4. Fix some api bug**
V2.8.6	2018/05/04	**1. support 91 pos fingerprint** **2. add api for nfc**
V2.8.7	2018/05/04	**1. To optimize the SDK**

Contents

Over View

Thank you for using our products. The best service will be provided to you.

Any corrections are welcomed if you find the mistakes on technology in this manual. The contents of this manual will be regularly updated, without notice. The updated content will be added in the new version of this manual. Products or procedures described in this manual will be improved and updated at any time.

This manual is an API programming guide mainly to describe our POS interface library. This library includes(IC/NFC/Magnetic Stripe cards 、printer) read and write function, fits for Smart POS and Mpos.

MposHandler:Control the connection

2.1: Init constructor

2.1.1: MposHandler

Prototype	**Public MposHandler (Context context)**	
Function	Default init construct function.	
Parameter	**parameter**	**Illustration**
	context	Context object
Return	**MposHandler**	
Note		

2.1.2: getInstance

Prototype	**public static MposHandler getInstance(Context _context)**	
Function	The singleton function	
Parameter	**parameter**	**Illustration**
	context	Context object
Return	**MposHandler**	
Note	Call singleton function, so that you can keeping on the connect in the whole life cycle of app	

2.1.3: addSwipeListener

Prototype	**public void addSwipeListener(SwipeListener listener)**	
Function	add SwipeListener to handler	
Parameter	**parameter**	**Illustration**
	SwipeListener	Callback function, Refer to appendix 2.
Return	**None**	
Note		

2.1.4: addEMVListener

Prototype	**public void addEMVListener(EMVListener listener)**	
Function	add EMVListener to handler	
Parameter	**parameter**	**Illustration**
	EMVListener	Callback function, Refer to appendix 3.
Return	**None**	
Note	Used to read IC Chip card.	

2.1.5: setShowLog

Prototype	**public void setShowLog(boolean log)**	
Function	Used to debug	
Parameter	**parameter**	**Illustration**
	log	True:output log to logcat
Return	**None**	
Note		

2.1.6: setShowAPDU

Prototype	**public void setShowAPDU(boolean show)**	
Function	used to debug	
Parameter	**parameter**	**Illustration**
	show	true:output APDU to logcat
Return	**None**	
Note	Used to read IC Chip card.	

2.1.7: isConnected

Prototype	**public boolean isConnected()**	
Function	make sure whether the connect is normal	
Parameter	**parameter**	**Illustration**
	None	
Return	**boolean**	true:connect false:disconnect
Note		

2.1.8: connect

Prototype	**public synchronized boolean connect()**	
Function	connect to the device	
Parameter	**parameter**	**Illustration**
	None	
Return	**boolean**	true:connect false:disconnect
Note		

2.1.9: onDestroy

Prototype	**public void onDestroy()**	
Function	release the connect and some object	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note	call this function before exit your app	

Settings : Control the operation

3.1: base interface

3.1.1: Settings

Prototype	**Public Settings(SwipeHandler handler)**	
Function	Construct function	
Parameter	**parameter**	**Illustration**
	SwipeHandler	MposHandler
Return	**Settings**	
Note	MposHandler via TTL	

3.1.2: getInstance

Prototype	public static Settings getInstance(SwipeHandler handler)	
Function	The singleton function	
Parameter	parameter	Illustration
	SwipeHandler	MposHandler
Return	Settings	
Note	MposHandler via TTL	

3.1.3： setReadSN

Prototype	public String setReadSN ()	
Function	Read SN NO. information	
Parameter	parameter	Illustration
	None	
Return	String	SN NO.
Note		

3.1.3： writeSN

Prototype	Public Boolean writeSN(String sn)	
Function	write SN number information	
Parameter	parameter	Illustration
	SN	Needed SN no. Writing, maximum 16byte
Return	Boolean	true:success false:fail
Note	SN no. Once only, no revision after its writing	

3.1.4： readVersion

Prototype	Public String readVersion()	
Function	Read information of version no.	
Parameter	parameter	Illustration

	None	
Return	**String**	If Return to null, it means communication fails or disconnect
Note	read out the version no. of firmware	

3.1.5：getSDKversion

Prototype	**public String getSDKversion()**	
Function	Read version of SDK	
Parameter	**parameter**	**Illustration**
	None	
Return	**String**	the version no. of SDK
Note	Version format: XXXM.N.L.YYYYYY_Z, eg POS2.7.0.180129_R	

3.1.6：mPosPowerOn

Prototype	**public void mPosPowerOn()**	
Function	power on the device	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note	need to power on the device before use it	

3.1.7：mPosPowerOff

Prototype	**public void mPosPowerOff()**	
Function	power off the device	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note	need to power off the device before use it	

3.2 Magnetic card

3.2.1: magOpen

Prototype	**Public String magOpen()**	
Function	Turn on mag card reader	
Parameter	**parameter**	**Illustration**
	None	
Return	Status code	00 represents success, all else means fail
Note	Use interrupt mode during mag card data reading. Once turn on the mag card reader, even the function reading is not needed, if you swipe the card, the head can read mag card data as well. Therefore, when not use the mag card reader, please keep it off.	

3.2.2: magClose

Prototype	**Public String magClose()**	
Function	Close mag card reader	
Parameter	**parameter**	**Illustration**
	none	
Return	Status code	00 represents success, all else means fail
Note		

3.2.3: magReset

Prototype	**Public string magReset()**	
Function	Reset maghead, and clear the buffer data in magcard	
Parameter	**parameter**	**Illustration**
	none	
Return	Status code	00 represents success, all else means fail
Note	Ensure the data which mag head reads is the most fresh, please test	

	the function first before the circle detection of swiping card to clear the buffer data in mag card.

3.2.4： magSwipe

Prototype	**Public String magSwipe()**	
Function	Check whether the card has been swiped or not	
Parameter	**parameter**	**Illustration**
	none	
Return	Status code	00 represents card swiped
Note	No matter the card is swipe or not, the function Returns immediately	

3.2.5： magRead

Prototype	**Public String magRead()**	
Function	Read buffer data of the first, second and third track in magcard	
Parameter	**parameter**	**Illustration**
	none	
Return	Mag card data	Combine status code and magcard data
Note	Status 1byte, 00means card reading fails, high address in front, when bite() is 1, it means the track data is read correctly, when bit1 is 1, it means the second track data reading is correct. When bit2 is 1, the third track correct. Mag card data is combine with data length(1byte) and data.	

3.3: Contact IC card

3.3.1 T=1/T=1 CPUcard

3.3.1.1：icReset

Prototype	**Public String ic Reset()**	
Function	Initialize contact IC card	
Parameter	**parameter**	**Illustration**
	none	
Return	ATR	Contact IC card(large CTR)reset info
Note	It's the default way to operate large CTR, when come to psam card, please change the reset method.	

3.3.1.2：icOff

Prototype	**Public String ic Reset()**	
Function	Initialize contact IC card	
Parameter	**parameter**	**Illustration**
	none	
Return	ATR	Contact IC card(large CTR)reset info
Note	It's the default way to operate large CTR, when come to psam card, please change the reset method.	

3.3.1.3：icDetect

Prototype	**Public String IC detect()**	
Function	Check IC card in deck or not	
Parameter	**parameter**	**Illustration**
	none	
Return	Status code	00 represents success, all else means fail
Note	The method is defaulted to operate large card deck, when come to psam card, please change the reset method. Since the library is	

	universal, whether the device supports this method, please contact us to confirm.

3.3.1.4: reset

Prototype	**Public String reset(int slot)**	
Function	Initialize the referred contact IC card card deck	
Parameter	**parameter**	**Illustration**
	slot	Contact IC card channel number(0~4)
Return	ATR	Contact IC card reset info
Note	The large deck channel no. Of our products is 0, PSAM card is 1~4 successively.	

3.3.1.5: off

Prototype	**Public String off(int slot)**	
Function	close the referred contact IC card card deck and power it off	
Parameter	**parameter**	**Illustration**
	slot	Contact IC card channel number(0~4)
Return	ATR	Contact IC card reset info
Note	The large deck channel no. Of our products is 0, PSAM card is 1~4 successively.	

3.3.1.6: detect

Prototype	**Public String detect(int slot)**	
Function	Check the IC card in referred deck or not	
Parameter	**parameter**	**Illustration**
	slot	Contact IC card channel number(0~4)
Return	status	00 means card exist, other else means not exist
Note	Since the library is universal, whether the device supports this method, please contact us to confirm.	

3.3.1.7：getDataWithAPDU

Prototype	**Public String getDataWithAPDU(Apdu_Send apdu)**	
Function	IC card operation function, it support IC universal interface protocol(T=0 and T=1)	
Parameter	**parameter**	**Illustration**
	Apdu_Send	Apdu command structure(com.imagpay)
Return	Response data	Null means operation fails or the device disconnect
Note	This method defaulted to operate IC large card deck.	

3.3.1.8：getDataWithAPDU

Prototype	**Public String getDataWithAPDU(Apdu_Send apdu)**	
Function	IC card operation function, it support IC universal interface protocol(T=0 and T=1)	
Parameter	**parameter**	**Illustration**
	slot	Contact ICcard channel no.(0~4)
	Apdu_Send	Apdu command structure(com.imagpay)
Return	Response data	Null means operation fails or the device disconnect
Note		

3.3.1.9：getDataWithAPDU

Prototype	**Public String getDataWithAPDU(int type, int slot, Apdu_Send apdu)**	
Function	Operate IC card and card deck to the referred method	
Parameter	**parameter**	**Illustration**
	type	Communication mode
	slot	Contact IC channel no.(0~4)
	Apdu_Send	Apdu command structure(com.imagpay)
Return	Response data	Null means operation fails or the device disconnect
Note	Reserved method, no need to adjust	

3.3.2.0：getDataWithAPDUForStr

Prototype	**Public String getDataWithAPDUForStr (String apdu)**	
Function	Operate IC card and card deck to the referred method	
Parameter	**parameter**	**Illustration**
	String	Apdu command string
Return	Response data	Null means operation fails or the device disconnect
Note		

3.3.2.1：getDataWithAPDUForStr

Prototype	**Public String getDataWithAPDUForStr (int slot, String apdu)**	
Function	Operate IC card and card deck to the referred method	
Parameter	**parameter**	**Illustration**
	slot	Contact IC channel no.(0~4)
	String	Apdu command string
Return	Response data	Null means operation fails or the device disconnect
Note		

3.3.2.2：getDataWithAPDUForStr

Prototype	**Public String getDataWithAPDUForStr (int type, int slot, String apdu)**	
Function	Operate IC card and card deck to the referred method	
Parameter	**parameter**	**Illustration**
	type	Communication mode
	slot	Contact IC channel no.(0~4)
	String	Apdu command string
Return	Response data	Null means operation fails or the device disconnect
Note		

3.3.2 SLE4442/SLE4428 card

3.3.2.1：sle4442Init

Prototype	Public Boolean sle4442()	
Function	Initialization to detect card type is correct or not	
Parameter	**parameter**	**Illustration**
	none	
Return	result	False means operation fails or the device disconnect, True means success.
Note	Only take SLE4442card as an example here, SLE4428 interface is similar, just change 4442 into 4428.	

3.3.2.2：sle4442SRD

Prototype	Public String sle4442SRD(int offset, int length)	
Function	Read data from referred address	
Parameter	**parameter**	**Illustration**
	offset	Offset address, range 0~225
	length	String length range 1~256
Return	Response data	Store and release the read data
Note		

3.3.2.3：sle4442SWR

Prototype	Public String sle4442SWR(int offset, int length, String data)	
Function	Write data from referred address	
Parameter	**parameter**	**Illustration**
	offset	Offset address, range 0~225
	length	String length range 1~256
	data	Store and release the written data
Return	result	False means operation fails or the device disconnect,

		True means success.
Note	Check card code before data writing	

3.3.2.4：sle4442CSC

Prototype	**Public String sle4442CSC(String key)**	
Function	Check card code	
Parameter	**parameter**	**Illustration**
	key	Store required code string to compare
Return	result	False means fail or device disconnect, True means success.
Note	SLE4442code is 3 bytes defaulted as "FFFFFF". SLE4428 code byte is 2, defaulted as "FFFF"	

3.3.2.5：sle4442RSC

Prototype	**Public String sle4442CSC(String key)**	
Function	Read card code	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Store read data code
Note	Only after it card code checking, the interface can read the current code successfully.	

3.3.2.6：sle4442WSC

Prototype	**Public String sle4442WSC(String key)**	
Function	Revise card code	
Parameter	**parameter**	**Illustration**
	key	Store the revise card code data
Return	result	False means fail or device disconnect, True means

		success.
Note		

3.3.2.7： sle4442RSTC

Prototype	**Public String sle4442RSTC()**	
Function	Code error read counter	
Parameter	**parameter**	**Illustration**
	none	
Return	response	Return code error counter value
Note	Two bytes means low position front, bit1~bit3 of low position means counter sign	

3.3.2.8： sle4442PRD

Prototype	**Public String sle4442PRD()**	
Function	Read protect mark	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Return to 4 byte(32bit)protect mark
Note	Read 4 bytes protection mark, (32bit) From offset address0, zone bit is 0 means protection set already, 1 means no protection set yet. what differs from 4428 is the protection mark can be read when reading data, it follows every byte.	

3.3.2.9： sle4442PWR

Prototype	**Public boolean sle4442PWR(int offset, int length, String data)**	
Function	Protect data of referred address	
Parameter	**parameter**	**Illustration**
	offset	Offset address, range 0~31

	length	String length range 1~32
	data	Store the protected data, should consist with the data in card
Return	result	False: fail True: success
Note		

3.3.3 AT24card

3.3.3.1：at24Reset

Prototype	**Public Boolean at 24Reset()**		
Function	Initialization, detect card type is correct or not		
Parameter	**parameter**	**Illustration**	
	none		
Return	result	False: fail or device disconnect, True: success	
Note	**Type**	**offSet**	**length**
	24C01("30")	0~127	1~128
	24C02("31")	0~255	1~256
	24C04("32")	0~511	1~512
	24C08("33")	0~1023	1~1024
	24C16("34")	0~2047	1~2048
	24C32("35")	0~5095	1~5096
	24C64("36")	0~8191	1~8192

3.3.3.2：at24Read

Prototype	**Public String at24Read(int offset, int length, String type)**	
Function	Read data from referred address	
Parameter	**parameter**	**Illustration**
	offSet	Offset address
	length	String length
	type	Card type
Return	Response data	Store read data
Note	Refer to the Note of chapter 2.3.3.1	

3.3.3.3： at24Write

Prototype	**Public String at24Write(int offset, int length, String type, String data)**	
Function	Write data to referred address	
Parameter	**parameter**	**Illustration**
	offSet	Offset address
	length	String length
	type	Card type
	data	Store read data
Return	resul	False means fail or the device disconnect, True means success.
Note	Refer to the Note of chapter 2.3.3.1	

3.4: NFC card

3.4.1 Mifare classic card

3.4.1.1： m1Request

Prototype	**Public String m1Request()**	
Function	Find card and Return to its UID no.	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Null means operation fails or the device disconnect
Note	This method is for M1card, defaulted for Type A card	

3.4.1.2：m1Request

Prototype	Public String m1Request(String type)	
Function	Find referred card and Return to its UID no.	
Parameter	parameter	Illustration
	type	Settings. M1_TYPE_A、 Setting.M1_TYPE_B、 Setting.M1_TYPE_C
Return	Response data	Null means operation fails or the device disconnect
Note	This method is for operating the referred M1card	

3.4.1.3：m1Select

Prototype	Public boolean m1Select(String sn)	
Function	Select proximity card via serial no.	
Parameter	parameter	Illustration
	sn	Returned Sn no. of m1Request function
Return	Result	Null means fail or the device disconnect
Note		

3.4.1.4：m1Auth

Prototype	Public boolean m1Auth(String sector, String pass)	
Function	Code Validation to specified sector of proximity card	
Parameter	parameter	Illustration
	sector	Serial no. Of the sector
	pass	Code 6bytes, defaulted as "FFFFFFFFFFFF"
Return	Result	Null means fail or the device disconnect
Note	This method is for M1card, defaulted to Type Acard	

3.4.1.5：m1Auth

Prototype	**Public boolean m1Auth(String type, String sector, String pass)**	
Function	Code Validation to specified sector of proximity card	
Parameter	**parameter**	**Illustration**
	type	Settings.M1_TYPE_A 、 Settings.M1_TYPE_B 、 Settings.M1_TYPE_C
	sector	Serial no. 00~0F
	pass	Code 6bytes, defaulted as “FFFFFFFFFFFF”
Return	Result	Null means fail or the device disconnect
Note		

3.4.1.6：m1ReadBlock

Prototype	**Public String m1ReadBlock(String block)**	
Function	Read block data	
Parameter	**parameter**	**Illustration**
	block	Block address: 00~03
Return	Response data	16 bytes block data
Note	This method read one block once, method of m1ReadSec can read the whole sector data	

3.4.1.7：m1WriteBlock

Prototype	**Public String m1WriteBlock(String block, String data)**	
Function	Code Validation to specified sector of standard proximity card	
Parameter	**parameter**	**Illustration**
	block	Block address: 00~03
	data	16bytes block data
Return	Status code	00 means card exist, other else means no card
Note	The 00 sector 00 block is vendor curing info storage block, no writing. The 03block of each sector is code control area, no writing.	

3.4.1.8：m1ReadSec

Prototype	**Public String m1ReadSec(String pass, String sector)**	
Function	Read the whole sector data	
Parameter	**parameter**	**Illustration**
	pass	Code 6bytes, defaulted as "FFFFFFFFFFFF"
	sector	Serial no. 00~0F
Return	Response data	48 bytes data
Note	The function is defaulted to use A code to operate the M1card of TypeA	

3.4.1.9：m1ReadSec

Prototype	**Public String m1ReadSec(String cardType, String KeyType, String pass, String sector)**	
Function	Read the whole sector data	
Parameter	**parameter**	**Illustration**
	cardType	Card type:M1_TYPE_A 、 M1_TYPE_B 、 M1_TYPE_C
	keyType	Key type:M1_KEY_A、M1_KEY_B
	pass	Code 6bytes, defaulted as "FFFFFFFFFFFF"
	sector	Serial no. 00~0F
Return	Response data	48 bytes data
Note		

3.4.1.10：m1WriteSec

Prototype	**Public String m1WriteSec(String pass, String sector, String data)**	
Function	Write the whole sector data	
Parameter	**parameter**	**Illustration**
	pass	Code 6bytes, defaulted as "FFFFFFFFFFFF"
	sector	Serial no. 00~0F
	data	48 bytes data needed to encode
Return	Status c	00means success, other else means fail.

Note	This function is defaulted to use A pass to operate M1card

3.4.1.11：m1WriteSec

Prototype	**Public String m1WriteSec(String cardType, String KeyType, String pass, String sector, String data)**	
Function	Write data of the whole sector	
Parameter	**parameter**	**Illustration**
	cardType	Card type:M1_TYPE_A 、 M1_TYPE_B 、 M1_TYPE_C
	keyType	Key type:M1_KEY_A、M1_KEY_B
	pass	Code 6bytes, defaulted as "FFFFFFFFFFFF"
	sector	Serial no. 00~0F
	data	48 bytes data needed to write
Return	Status code	00means success, other else means fail
Note		

3.4.2 Mifare Ultralight card

3.4.2.1：ulRequest

Prototype	**Public String ulRequest()**	
Function	To find ULTra lightcard	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Return to data of 7bytes card no.
Note		

3.4.2.2：ulSelect

Prototype	**Public boolean ulSelect(String sn)**

Function	Pitch up ULTra lightcard	
Parameter	**parameter**	**Illustration**
	sn	Find card interface Return to card no. data
Return	Result	False means fail or device disconnect, True means success.
Note		

3.4.2.3：ulReadPage

Prototype	**Public String ulReadPage(String page)**	
Function	Read ULTra lightcard data of specified page	
Parameter	**parameter**	**Illustration**
	page	Page number range 0~15
Return	Response data	4 bytes data
Note	Hex system string mode page no.(00~0F) is needed here	

3.4.2.4：ulWritePage

Prototype	**Public boolean ulWritePage(String page, Sting data)**	
Function	Write data to specified page of ULTra lightcard	
Parameter	**parameter**	**Illustration**
	page	Page number range 0~15
	data	Store 4 bytes data which is needed writing
Return	result	False means fail or device disconnect, True means success.
Note	PAGE here is the same meaning as BLOCK of M1 card	

3.4.3 Mifare DESFire card

3.4.3.1：dfSelect

Prototype	**Public String dfSelect()**	
Function	Select card	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Return to data of 7bytes card no.
Note		

3.4.3.2：dfReset

Prototype	**Public Boolean dfReset()**	
Function	Card reset	
Parameter	**parameter**	**Illustration**
	none	
Return	result	False: fail True: success
Note	Select, reset card by sequence first, then followed operations can be procceeded.	

3.4.3.3：dfPSE

Prototype	**Public Boolean dfPSE(String aid)**	
Function	Application selection	
Parameter	**parameter**	**Illustration**
	aid	Application no., 3bytes high position front
Return	result	False: fail True: success
Note	AID is defaulted as “000000” when card reset	

3.4.4 ISO14443 A&B card

3.4.4.1：typeBRequest

Prototype	**Public String typeBRequest()**	
Function	Find card	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Return to card info
Note	Acard uses contact IC CPUcard interface, only channel no. is needed, here only Bcard interface is described.	

3.4.4.2：typeBATTRIB

Prototype	**Public Boolean typeBATTRIB()**	
Function	Bcard parameter selection	
Parameter	**parameter**	**Illustration**
	none	
Return	result	False means fail and True means success
Note		

3.4.4.3：typeBApdu

Prototype	**Public String typeBApdu(Apdu_Send apdu)**	
Function	Bcard send APUD command	
Parameter	**parameter**	**Illustration**
	apdu	Send required APDU command object
Return	Response data	Apduresponse data
Note	APDU sending refers to contact CPU card	

3.4.4.4：typeBHalt

Prototype	**Public Boolean typeBHalt()**	
Function	Dormant Bcard	
Parameter	**parameter**	**Illustration**
	none	
Return	Result	False means fail or device disconnected, True means success
Note		

3.4.5 SONY Felica card

3.4.5.1：felicaRequest

Prototype	**Public String felicaRequest()**	
Function	Find card	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Return to Felicacard no. etc. info
Note		

3.4.5.2：felicaSendCmds

Prototype	**Public String felicaSendCmds(String cmd)**	
Function	Send specified command to felica and get Return data	
Parameter	**parameter**	**Illustration**
	cmd	Store command data which needs sending
Return	Response data	Return to Felicacard data
Note		

3.4.6 NFC Base API

3.4.6.1： nfcRequest

Prototype	**public void nfcRequest()**	
Function	auto detect nfc card in a thread , and callback the card type	
Parameter	**parameter**	**Illustration**
	None	
Return	None	
Note	If detect nfc card, will callback onCardDetected function	

3.4.6.2： nfcStop

Prototype	**public void nfcStop()**	
Function	exit the nfcRequest function	
Parameter	**parameter**	**Illustration**
	None	
Return	None	
Note		

3.5: ID card

3.5.1： writeIDReset

Prototype	**Public String writeIDReset()**	
Function	Reset ID card	
Parameter	**parameter**	**Illustration**
	none	
Return	Status code	00means card exist, other else means no card
Note		

3.5.2： getIDData

Prototype	Public String getIDData()	
Function	obtain ID card data	
Parameter	**parameter**	**Illustration**
	none	
Return	Response data	Please refer to the data analysis method of Demo
Note	ID card operation needs picture decode lib. Put document wltlibinto sdcard root catalog.	

3.5.3： writeIDOff

Prototype	Public String writeIDOff()	
Function	ID card module power off	
Parameter	**parameter**	**Illustration**
	none	
Return	Status code	00means card exist, other else means no card
Note		

3.6: printer

3.6.1： mPosEnterPrint

Prototype	public boolean mPosEnterPrint()	
Function	Printing initialization	
Parameter	**parameter**	**Illustration**
	none	
Return	Implement result	True:success False:fail
Note	Recover printer defaulted setting when initialization implement.	

3.6.2： mPosPrintAlign

Prototype	**public void mPosPrintAlign(String align)**	
Function	Set the print alignment format	
Parameter	**parameter**	**Illustration**
	align	Align style
Return	Implement result	
Note	After the printer is valid settings will remain valid until the next set Align left: Settings.MPOS_PRINT_ALIGN_LEFT, Align center: Settings. MPOS_PRINT_ALIGN_CENTER, Align right: Settings. MPOS_PRINT_ALIGN_RIGHT	

3.6.3： mPosPrintTextSize

Prototype	**public void mPosPrintTextSize(String size)**	
Function	Set the print font size	
Parameter	**parameter**	**Illustration**
	size	Font style
Return	Implement result	
Note	After the printer is valid settings will remain valid until the next set Normal font printing Settings.MPOS_PRINT_TEXT_NORMAL, Fonts print twice as high Settings.MPOS_PRINT_TEXT_DOUBLE_HEIGHT, Fonts print twice as width Settings.MPOS_PRINT_TEXT_DOUBLE_WIDTH, Fonts print twice as width and high Settings.MPOS_PRINT_TEXT_DOUBLE_SIZE	

3.6.4： mPosPrnStr

Prototype	**public void mPosPrnStr(String str)**	
Function	Print the string	
Parameter	**parameter**	**Illustration**
	str	String to print

Return	Implement result	
Note		

3.6.5：mPosPrnImg

Prototype	**public void mPosPrnImg(Bitmap bit)**	
Function	Print the picture	
Parameter	**parameter**	**Illustration**
	bit	Bitmap object
Return	none	
Note		

3.6.6：prnImage

Prototype	**Public void prnStr(Bitmap bitmap)**	
Function	Picture print	
Parameter	**parameter**	**Illustration**
	bitmap	Bitmap file
Return	none	Implement printing, no Return value
Note		

3.6.7：mPosPrnConvertBmp

Prototype	**public List<byte[]> mPosPrnConvertBmp(Bitmap bitmap)**	
Function	Convert BMP Picture to print data	
Parameter	**parameter**	**Illustration**
	bitmap	Bitmap file
Return	List<byte[]>	Print data
Note		

3.6.8: mPosPrnImg

Prototype	**public void mPosPrnImg(List<byte[]> bit)**	
Function	Picture print	
Parameter	**parameter**	**Illustration**
	List<byte[]>	print data
Return	none	Implement printing, no Return value
Note		

3.6.9: mPosExitPrint

Prototype	**public boolean mPosExitPrint()**	
Function	Exit Print	
Parameter	**parameter**	**Illustration**
	none	
Return	Implement result	True:success False:fail
Note		

3.7.0: prnStr

Prototype	**public void prnStr(String str)**	
Function	Prepare print string, add data to buffer	
Parameter	**parameter**	**Illustration**
	String	print text
Return	None	
Note		

3.7.1: prnStr

Prototype	**public void prnStr(String str, PrnStrFormat format)**	
Function	Prepare print string with custome font&style , add data to buffer	
Parameter	**parameter**	**Illustration**
	String	print text
	PrnStrFormat	refer to appendix 6
Return	None	
Note		

3.7.2: prnBitmap

Prototype	**public void prnBitmap(Bitmap bitmap)**	
Function	Prepare print string, add data to buffer	
Parameter	**parameter**	**Illustration**
	Bitmap	print bitmap
Return	None	
Note		

3.7.3: prnStart

Prototype	**public boolean prnStart( )**	
Function	start to print buffer data	
Parameter	**parameter**	**Illustration**
	None	
Return	boolean	True: print successful Flase: print failure
Note		

3.7.4: prnStrClear

Prototype	public void prnStrClear ( )	
Function	clear print buffer data	
Parameter	parameter	Illustration
	None	
Return	None	
Note		

3.7.5: prnStep

Prototype	public void prnStep()	
Function	print a newline	
Parameter	parameter	Illustration
	None	
Return	None	
Note		

EMV4.3 Kernel Guide

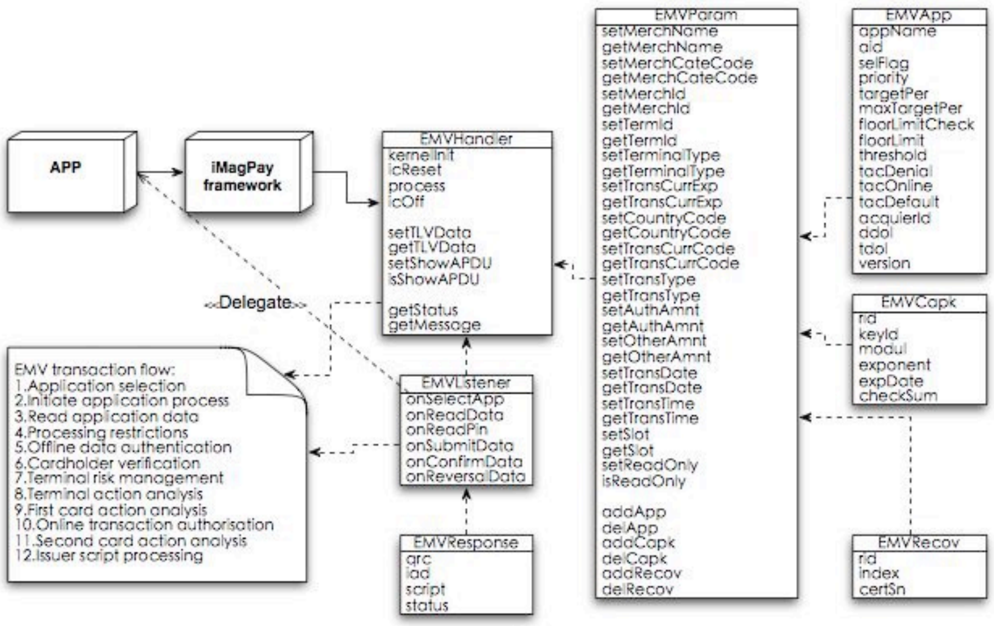
4.1: Introduce

This document describes the interfaces and requirements for using the EMV kernel. EMV kernel is developed with EMV level 2.

4.2: Architecture

The diagram below shows the components that are required to use EMV kernel.

- App is the payment application on iPhone and iPad.
- ZCSCombo.jar and JNIEMV.so are the EMV kernel, which contains the EMV processing functionality
- EMV Handler is a set of functions that contain EMV transaction flow



4.3： How to use

4.3.1 Initialize EMV Handler

After user selects some goods or services and will be ready to pay, App should show an
activity or view and initialize EMV Handler. EMV Handler is used for EMV transaction. When
the activity is started, app should drive EMV reader. This is an example:

```
//When app create an activity, Android will fire onCreate function.
public void onCreate(Bundle savedInstanceState) {
        ……
        _handler = new EMVHandler(this);
        // if call setShowAPDU(true), SDK will show EMV kernel log
        _handler.setShowAPDU(true);
        _settings = new Settings(_handler);
        _handler.addSwipeListener(new SwipeListener() {
        ……
        });
        _handler.addEMVListener(new EMVListener() {
        ……
        });
}
```

4.3.2 Implement EMV Listener

EMV Listener is a set of events that should be implemented in the activity. When app runs an EMV transaction, some steps need to be confirmed with card holder. and some steps need to connect payment server. Event list is:

```
//if card support more one application, EMV Handler will fire the event and app should show a select view. Card holder should select one of applications.
public int onSelectApp(List apps) {
//apps[0] is a json NSString. Example:
{"appPreName":"MasterCard","appLabel":"MASTERCARD","AID":"a0000000041010","priority":1,"priorityExistFlg":1}
//if appPreName is not null, the select view should show appPreName;
//if appPreName is null and appLabel is not null, the select view should show appLabel;
//if appPreName is null and appLabel is null, the select view should show AID.
Return getSelectApp(apps);
}
//after read card data, EMV Handler will fire the event. If app only need to read card data and do not need to finish a EMV transaction, it only need to Return NO. EMV Handler will not go next step and finish.
public boolean onReadData() {
Return true;
}
//If EMV transaction needs a pin, EMV Handler will fire the event. App should show a input view and card holder should input pin or select bypass
//pin type: EMVConstants.ONLINE_PIN/OFFLINE_PLAIN_PIN/OFFLINE_ENCRYPT_PIN
public String onReadPin(final int type, final int ucPinTryCnt) {
//if card holder want to go bypass, this should Return null.
Return showPinView();
}
//If EMV transaction needs to connect server, EMV Handler will fire the event. App should connect server and submit some data with ISO8853, then app get the response. See 12.1.2 of EMV Book 4.
public EMVResponse onSubmitData() {
// submit data to payment server and get response, such as ARC/IAD/Issuer Script
//                  EMVResponse resp = getServerResponse();
//if online approve
//                  EMVResponse resp = new EMVResponse();
//                  resp.setARC("3030");
//                  resp.setIAD(...)
//                  resp.setScript(...)
//                  resp.setStatus(EMVResponse.ONLINE_APPROVE);

//if online reject
//                    EMVResponse resp = new EMVResponse();
//                    resp.setARC("3035");
//                    resp.setIAD(...)
//                    resp.setScript(...)
//                    resp.setStatus(EMVResponse.ONLINE_DENIAL);

//if online referrals
//need to show a dialog "Call your bank", let user confirm approve or reject

//if online failed
                    EMVResponse resp = null;
                    Return resp;
}
//After trans approves, EMV Handler will fire the event. App should submits some data to server
and confirms the transaction. See 12.1.4 of EMV book 4
public void onConfirmData() {
}
//After online approves but terminal reject or online rejects but terminal approves, EMV Handler
will fire the event. App should submits some data to server and reversals the transaction. See
12.1.8 of EMV book 4.
public void onReversalData() {
}
```

4.3.3 Run EMV transaction

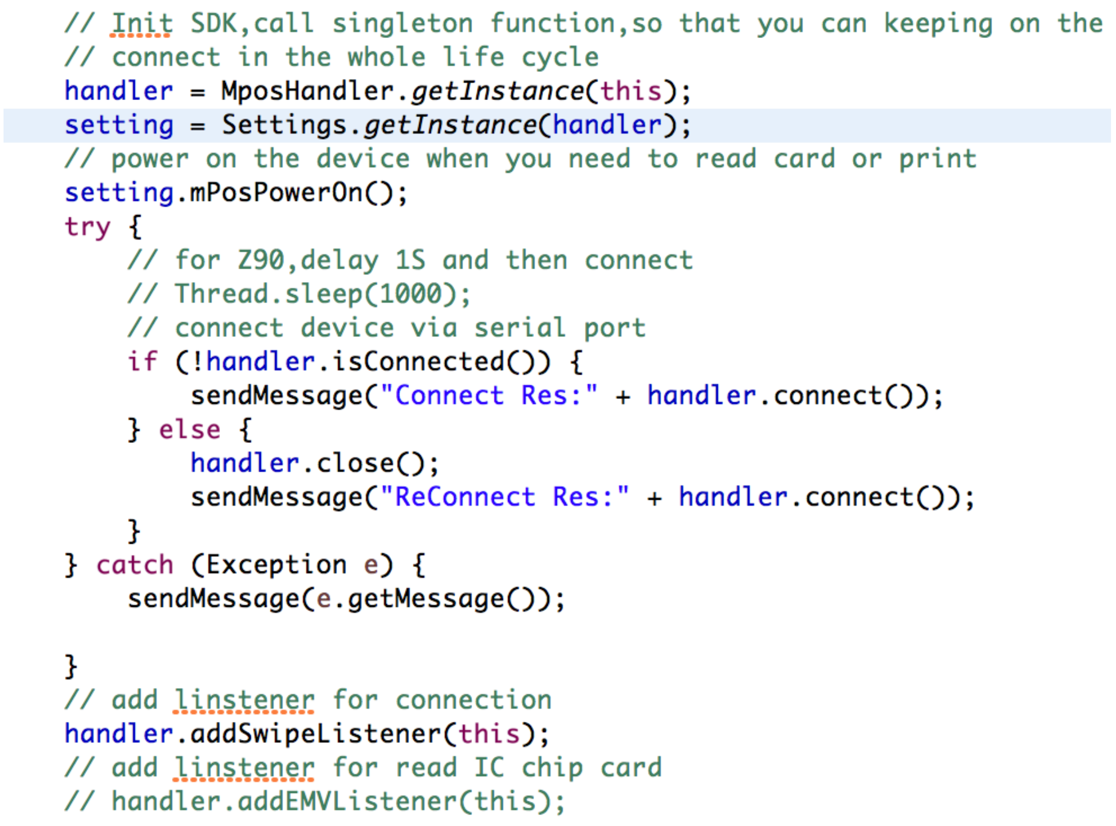
After user press payment. App need to create a EMVParam object and setup some parameters, then call icReset function and run process function.

```
private void startEMV() {
          showProgressDialog("Reading data......");
          EMVParam param = new EMVParam();
          param.setSlot((byte) 0x00);
// if call setReadOnly(true), SDK will only read card data
// if call setReadOnly(false), SDK will read card data and verify pin and submit data
          param.setMerchName("4368696E61");// hex string of china
          param.setMerchCateCode("0001");
          param.setMerchId("313233343536373839303132333435");
          param.setTermId("3132333435363738");
          param.setTerminalType((byte) 0x22);
          param.setCapability("E0F8C8");
          param.setExCapability("F00000A001");
          param.setTransCurrExp((byte) 0x02);
          param.setCountryCode("0840");
          param.setTransCurrCode("0840");
```

```
        param.setTransType((byte) 0x00);
        param.setTermIFDSn("3838383838383838");//SN is 88888888

        param.setAuthAmnt(8000000);// transaction amount
        param.setOtherAmnt(0);
        Date date = new Date();
        DateFormat sdf = new SimpleDateFormat("yyMMdd");
        param.setTransDate(sdf.format(date));
        sdf = new SimpleDateFormat("HHmmss");
        param.setTransTime(sdf.format(date));

        // FIME parameters(MasterCard Test Card), if other card type, need to change.
        loadMasterCardAIDs(param);
        loadMasterCardCapks(param);
        loadMasterCardRevocs(param);
        // Visa
        loadVisaAIDs(param);
        loadVisaCapks(param);
        loadVisaRevocs(param);

        _handler.kernelInit(param);
        if (_handler.icReset() != null) {
            _handler.process();
        }
        _handler.icOff();
}
```

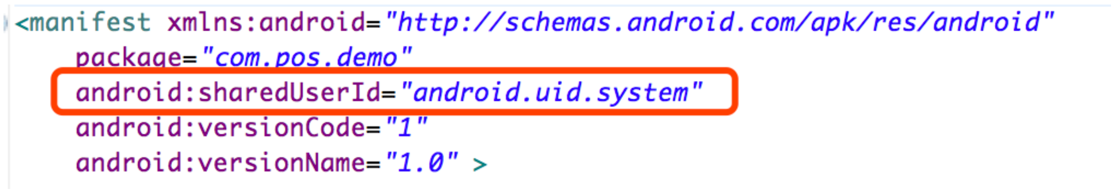
5. SDK Guide

5.1 Initialize SDK



```
// Init SDK,call singleton function,so that you can keeping on the
// connect in the whole life cycle
handler = MposHandler.getInstance(this);
setting = Settings.getInstance(handler);
// power on the device when you need to read card or print
setting.mPosPowerOn();
try {
    // for Z90,delay 1S and then connect
    // Thread.sleep(1000);
    // connect device via serial port
    if (!handler.isConnected()) {
        sendMessage("Connect Res:" + handler.connect());
    } else {
        handler.close();
        sendMessage("ReConnect Res:" + handler.connect());
    }
} catch (Exception e) {
    sendMessage(e.getMessage());

}
// add linstener for connection
handler.addSwipeListener(this);
// add linstener for read IC chip card
// handler.addEMVListener(this);
```

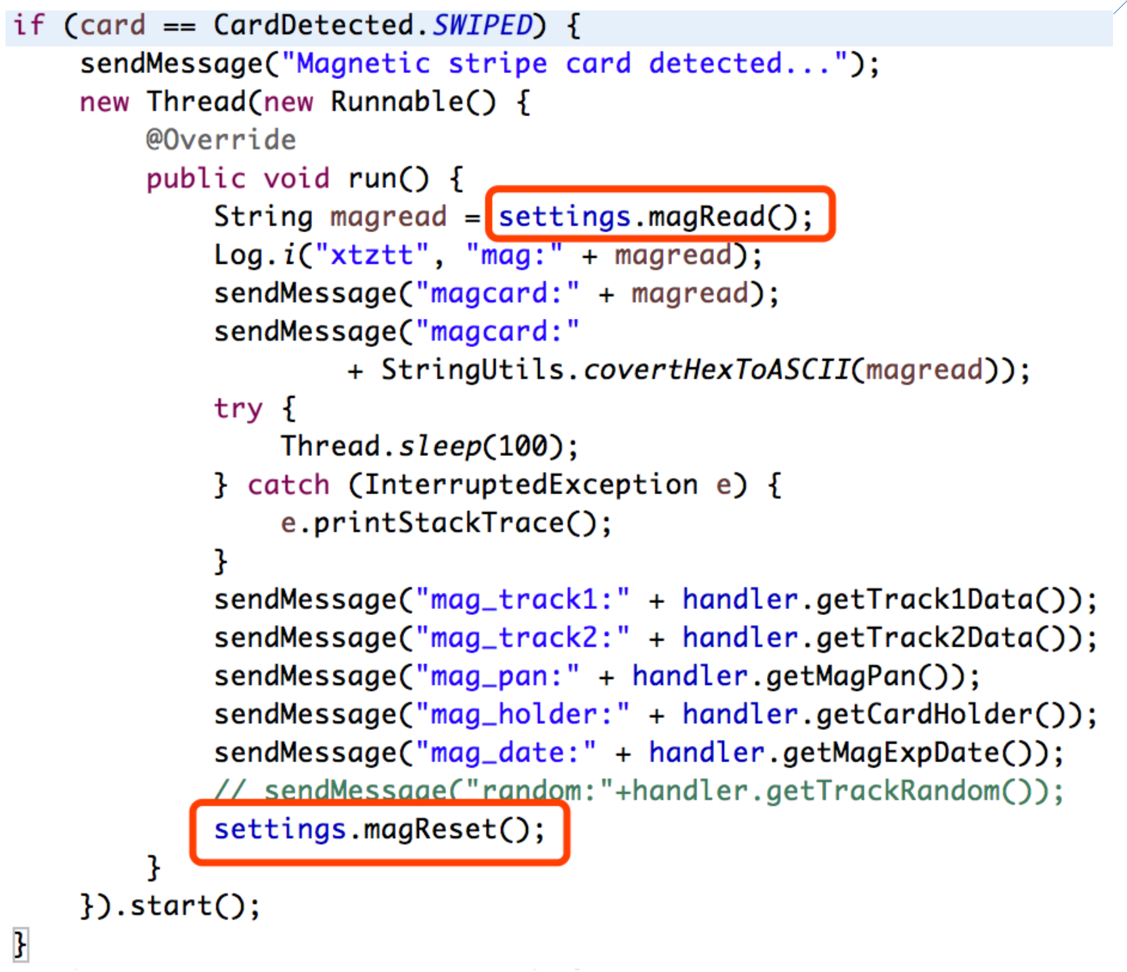
For 90POS,need to config shareUserId



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pos.demo"
    android:sharedUserId="android.uid.system"
    android:versionCode="1"
    android:versionName="1.0" >
```

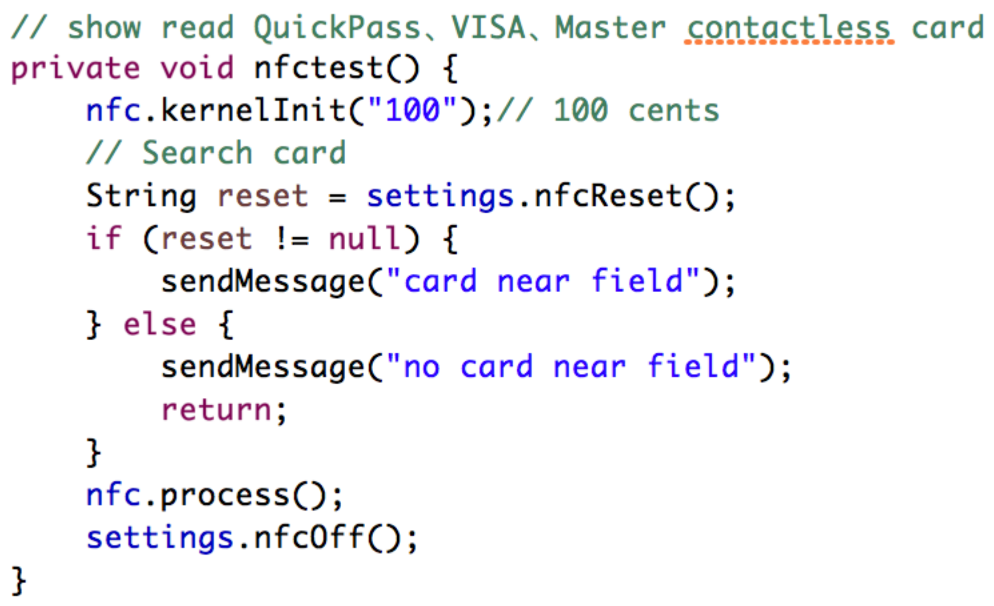
5.2 read Magnetic Stripe Card

1.detect CardDected.SWIPED in onCardDetect callback function.
2.call magRead function to read out card data
3.call magReset function to clear data buffer



```
if (card == CardDetected.SWIPED) {
    sendMessage("Magnetic stripe card detected...");
    new Thread(new Runnable() {
        @Override
        public void run() {
            String magread = settings.magRead();
            Log.i("xtztt", "mag:" + magread);
            sendMessage("magcard:" + magread);
            sendMessage("magcard:"
                    + StringUtils.covertHexToASCII(magread));
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            sendMessage("mag_track1:" + handler.getTrack1Data());
            sendMessage("mag_track2:" + handler.getTrack2Data());
            sendMessage("mag_pan:" + handler.getMagPan());
            sendMessage("mag_holder:" + handler.getCardHolder());
            sendMessage("mag_date:" + handler.getMagExpDate());
            // sendMessage("random:"+handler.getTrackRandom());
            settings.magReset();
        }
    }).start();
}
```

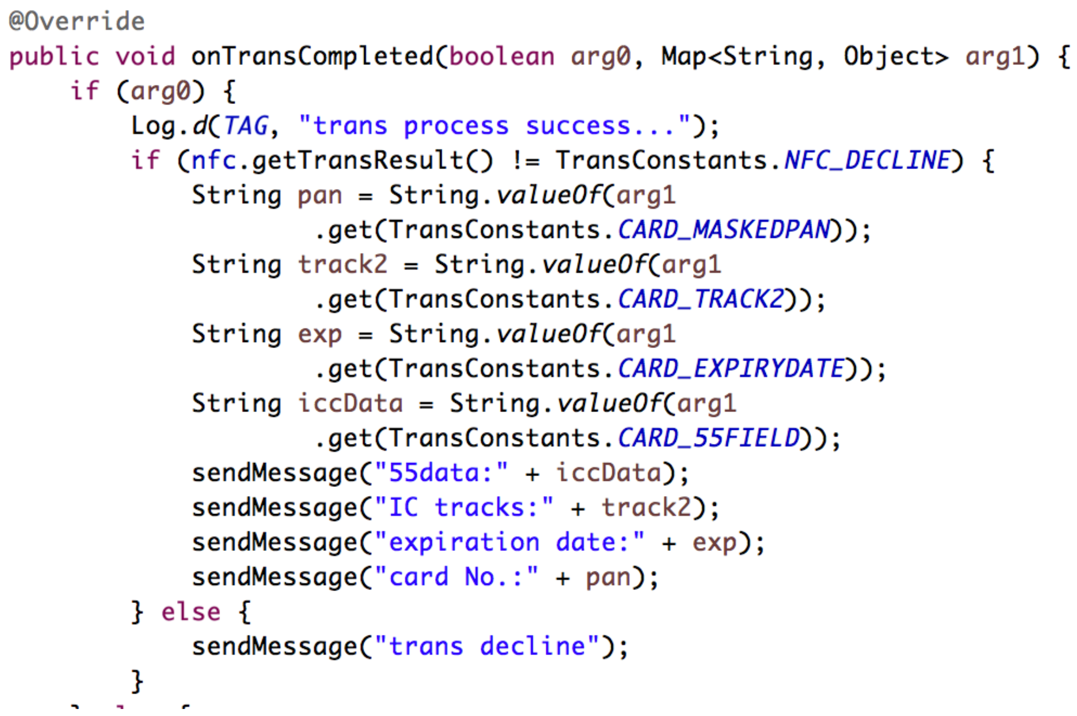
5.3 how to use NFCEmvHandler



```
// show read QuickPass、VISA、Master contactless card
private void nfctest() {
    nfc.kernelInit("100");// 100 cents
    // Search card
    String reset = settings.nfcReset();
    if (reset != null) {
        sendMessage("card near field");
    } else {
        sendMessage("no card near field");
        return;
    }
    nfc.process();
    settings.nfcOff();
}
```

After completed transaction, sdk will callback

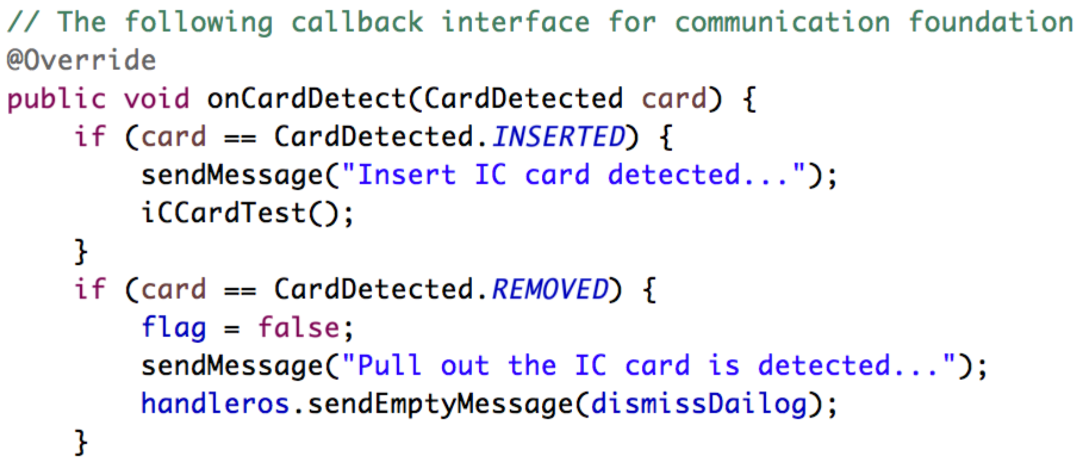
onTransCompleted function, you can get card data here.



```
@Override
public void onTransCompleted(boolean arg0, Map<String, Object> arg1) {
    if (arg0) {
        Log.d(TAG, "trans process success...");
        if (nfc.getTransResult() != TransConstants.NFC_DECLINE) {
            String pan = String.valueOf(arg1
                    .get(TransConstants.CARD_MASKEDPAN));
            String track2 = String.valueOf(arg1
                    .get(TransConstants.CARD_TRACK2));
            String exp = String.valueOf(arg1
                    .get(TransConstants.CARD_EXPIRYDATE));
            String iccData = String.valueOf(arg1
                    .get(TransConstants.CARD_55FIELD));
            sendMessage("55data:" + iccData);
            sendMessage("IC tracks:" + track2);
            sendMessage("expiration date:" + exp);
            sendMessage("card No.:" + pan);
        } else {
            sendMessage("trans decline");
        }
```

5.4 read IC chip Card

1.detect CardDected.INSERTED in onCardDetect callback function.
2.run emv process to read ic chip card in a thread. Refer to section 4.



```
// The following callback interface for communication foundation
@Override
public void onCardDetect(CardDetected card) {
    if (card == CardDetected.INSERTED) {
        sendMessage("Insert IC card detected...");
        iCCardTest();
    }
    if (card == CardDetected.REMOVED) {
        flag = false;
        sendMessage("Pull out the IC card is detected...");
        handleros.sendEmptyMessage(dismissDailog);
    }
```

5.5 how to send APDU with IC/NFC

```
// For IC card
setting.icReset();
String apdu = "0084000008";
setting.getDataWithAPDUForStr(apdu);
setting.icOff();
// For PSAM card
setting.reset(Settings.SLOT_PSAM01);// or SLOT_PSAM02
apdu = "0084000008";
setting.getDataWithAPDUForStr(Settings.SLOT_PSAM01, apdu);
setting.off(Settings.SLOT_PSAM01);
// For NFC CPU
setting.nfcReset();
setting.getDataWithAPDUForStr(Settings.SLOT_NFC, apdu);
setting.nfcOff();
```

5.6 print

1.call prnStr , add print text to buffer

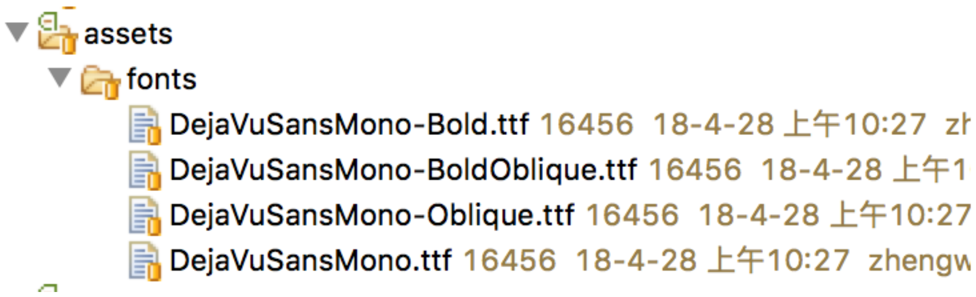
2.you can print with specified style、font、size and so on

3.call prnStart to start to print

4. print status will callback onPrintStatus

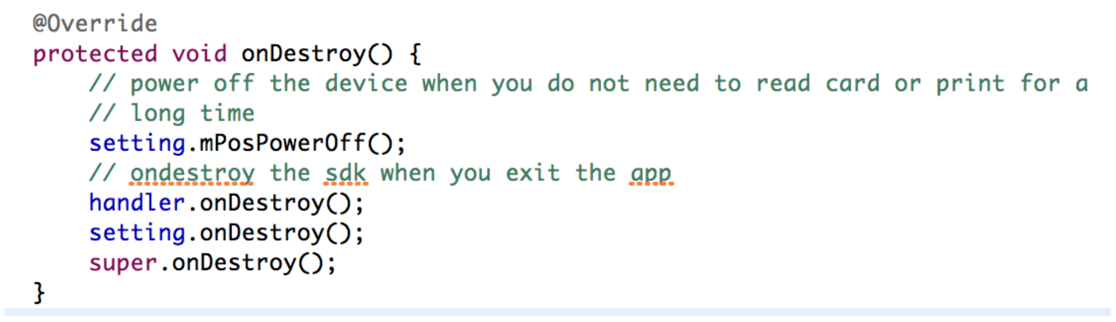
```
if (setting.isPrinting()) {// check print status
    Message msg = new Message();
    msg.what = 101;
    mHandler.sendMessage(msg);
    Log.d(TAG, "setting.isPrinting():" + setting.isPrinting());
    return;
}
StringBuffer receipts = new StringBuffer();
receipts.append("The cardholder stub   \nPlease properly keep\n");
// 1. print text with default style and font
setting.prnStr(receipts.toString());
receipts.setLength(0);
receipts.append("Merchant Name:ABC\n");
receipts.append("Merchant No.:846584000103052\n");
receipts.append("Terminal No.:12345678\n");
PrnStrFormat psf = new PrnStrFormat();
psf.setFont(PrnTextFont.MONOSPACE);
psf.setStyle(PrnTextStyle.BOLD);
// 2. print text with specified style and font
setting.prnStr(receipts.toString(), psf);
receipts.setLength(0);
receipts.append("Trade Type:consumption\n");
receipts.append("Serial No.:000024  \nAuthenticode:096706\n");
receipts.append("Date/Time:2016/09/01 11:27:12\n");
receipts.append("Ref.No.:123456789012345\n");
receipts.append("Amount:$ 100.00\n");
// 3. print text with custom font
psf.setFont(PrnTextFont.CUSTOM);
psf.setAm(getAssets());
psf.setPath("fonts/DejaVuSansMono.ttf");
setting.prnStr(receipts.toString(), psf);
// 4. start to print
setting.prnStart();
```

For custom font, need to add .ttf to project like this:



```
▼ assets
  ▼ fonts
      DejaVuSansMono-Bold.ttf 16456  18-4-28 上午10:27  zh
      DejaVuSansMono-BoldOblique.ttf 16456  18-4-28 上午1
      DejaVuSansMono-Oblique.ttf 16456  18-4-28 上午10:27
      DejaVuSansMono.ttf 16456  18-4-28 上午10:27  zhengw
```

5.7 exit app

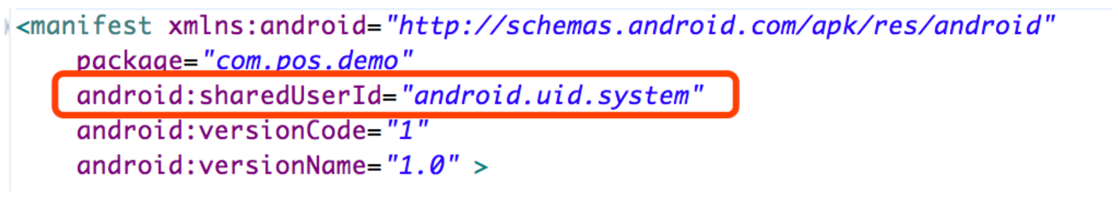


```
@Override
protected void onDestroy() {
    // power off the device when you do not need to read card or print for a
    // long time
    setting.mPosPowerOff();
    // ondestroy the sdk when you exit the app
    handler.onDestroy();
    setting.onDestroy();
    super.onDestroy();
}
```

5.8 signature apk file

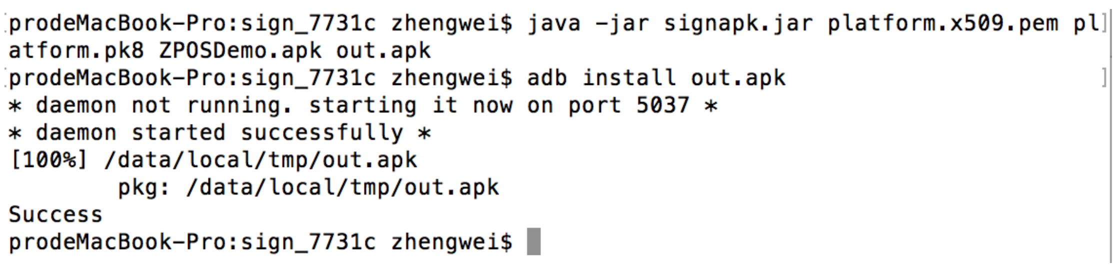
For 90Pos,before install apk to pos device , you need to signature your apk file at first.

1. .Please add "android:sharedUserId="android.uid.system"" in Androidmanifest.



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pos.demo"
    android:sharedUserId="android.uid.system"
    android:versionCode="1"
    android:versionName="1.0" >
```

2. use signature tools to sign apk file and then “adb install” it



```
prodeMacBook-Pro:sign_7731c zhengwei$ java -jar signapk.jar platform.x509.pem pl
atform.pk8 ZPOSDemo.apk out.apk
prodeMacBook-Pro:sign_7731c zhengwei$ adb install out.apk
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
[100%] /data/local/tmp/out.apk
        pkg: /data/local/tmp/out.apk
Success
prodeMacBook-Pro:sign_7731c zhengwei$ 
```

6. Fingerprint Guide

6.1: Init constructor

6.1.1: FingerprintHandler

Prototype	**public FingerprintHandler (Context context)**	
Function	Construct function	
Parameter	**parameter**	**Illustration**
	context	Context object
Return	**FingerprintHandler**	
Note		

6.1.2: getInstance

Prototype	**public static FingerprintHandler getInstance(Context _context)**	
Function	The singleton function	
Parameter	**parameter**	**Illustration**
	context	Context object
Return	**FingerprintHandler**	
Note		

6.1.3: addFignerprintListener

Prototype	**public void addFignerprintListener (FignerprintListener listener)**	
Function	add FignerprintListener to handler	
Parameter	**parameter**	**Illustration**
	FignerprintListener	Callback function, Refer to appendix 4.
Return	**None**	
Note		

6.1.4: onDestroy

Prototype	**public void onDestroy()**	
Function	release object	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note	call this function before exit your app	

6.2: Fingerprint operate

6.2.1: enrollment

Prototype	**public void enrollment ()**	
Function	Open fignerprint ,start enrollment	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note	progress will callback to FignerprintListener	

6.2.2: cancelEnrollment

Prototype	**public void cancelEnrollment ()**	
Function	Cancel enrollment	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note		

6.2.3: authenticate

Prototype	**public void authenticate ()**
Function	authenticate fingerprint

Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note		

6.2.4: cancelAuthentication

Prototype	**public void cancelAuthentication ()**	
Function	Cancel authenticate fingerprint	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note		

6.2.5: remove

Prototype	**public void remove(int fingerid)**	
Function	remove fingerprint	
Parameter	**parameter**	**Illustration**
	fingerid	fingerprint ID
Return	**None**	
Note		

6.2.6: getEnrolledFingerprints

Prototype	**public List<Integer> getEnrolledFingerprints()**	
Function	list fingerprints ID	
Parameter	**parameter**	**Illustration**
	None	
Return	**List<Integer>**	list fingerprints ID
Note		

6.2.7: hasEnrolledFingerprints

Prototype	**public boolean hasEnrolledFingerprints(int fingerId)**

Function	check fingerprint with ID	
Parameter	**parameter**	**Illustration**
	None	
Return	**boolean**	true : exist
Note		

6.2.8: getImage

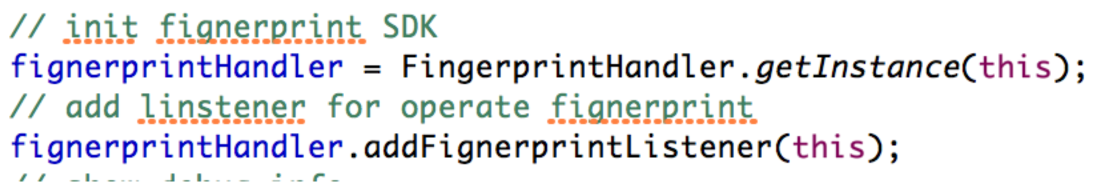
Prototype	**public void getImage()**	
Function	record fingerprint image data	
Parameter	**parameter**	**Illustration**
	None	
Return	**None**	
Note	run in a thread , and then wait for some seconds, SDK will call back “onGetImageComplete” linstener	

6.2.9: generateBmp

Prototype	**public Bitmap generateBmp(byte[] buffer, String path)**	
Function	convert fingerprint image data to bmp	
Parameter	**parameter**	**Illustration**
	buffer	image source data
	path	generate image file path
Return	**Bitmap**	bitmap of BMP
Note		

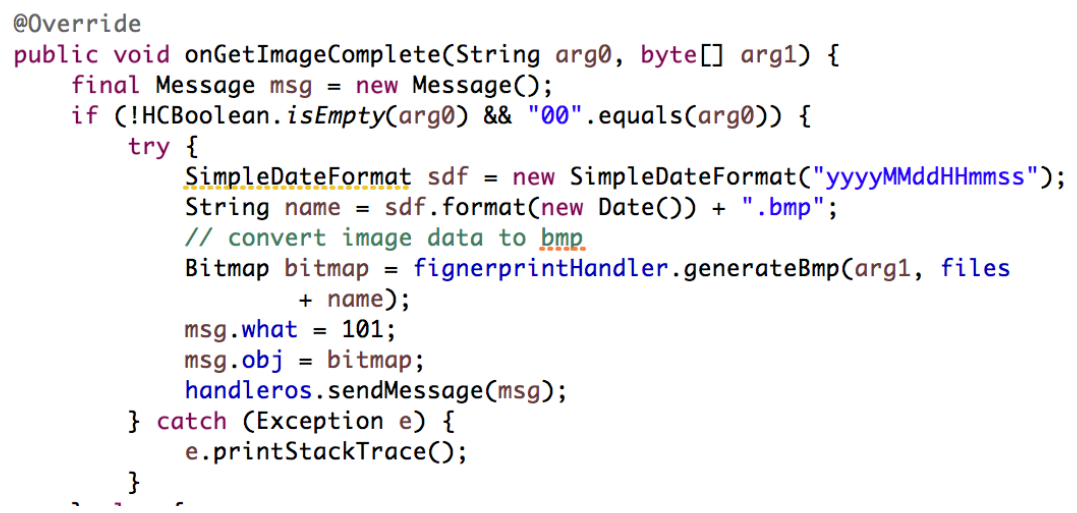
6.3: How to use FingerprintHandler

1.init SDK , and add listener



```
// init fignerprint SDK
fignerprintHandler = FingerprintHandler.getInstance(this);
// add linstener for operate fignerprint
fignerprintHandler.addFignerprintListener(this);
```

2. call getImage get fingerprint source data
3. covert data to bmp



```
@Override
public void onGetImageComplete(String arg0, byte[] arg1) {
    final Message msg = new Message();
    if (!HCBoolean.isEmpty(arg0) && "00".equals(arg0)) {
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
            String name = sdf.format(new Date()) + ".bmp";
            // convert image data to bmp
            Bitmap bitmap = fignerprintHandler.generateBmp(arg1, files
                    + name);
            msg.what = 101;
            msg.obj = bitmap;
            handleros.sendMessage(msg);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Appendix

1. PosModel

the model of Smart POS

90
91

2. SwipeListener

```
/**
     * fire after ttl disconnected
     *
     * @param event
     */
    public void onDisconnected(SwipeEvent event);

    /**
     * fire after ttl connected
     *
     * @param event
     */
    public void onConnected(SwipeEvent event);

    /**
     * fire after parsed the read data.
     *
     * @param event
     */
    public void onParseData(SwipeEvent event);

    /**
     * fire after detect insert icc or swipe card
     *
     * @param type
     * @since v2.5.5
     */
    public void onCardDetect(CardDetected type);


    /**
     * fire after enter print mode
     *
     * @param status
     * @since v2.5.8
     */
    public void onPrintStatus(PrintStatus status);

    /** fire after emv process err
     * @param err
     */
    public void onEmvStatus(EmvStatus err);
```

3. EMVListener

```
/**
     * fire after select app
     *
     * @param apps
     */
    public int onSelectApp(List<String> apps);

    /**
     * fire after read data
     */
    public boolean onReadData();

    /**
     * fire after need to input pin
     *
     * @param type
     * @param ucPinTryCnt
     */
    public String onReadPin(int type, int ucPinTryCnt);

    /**
     * fire after need to submit data
     *
     */
    public EMVResponse onSubmitData();

    /**
     * fire after offline approve or online approve
     *
     */
    public void onConfirmData();

    /**
     * fire after online approve but terminal reject
     *
     */
    public void onReversalData();
```

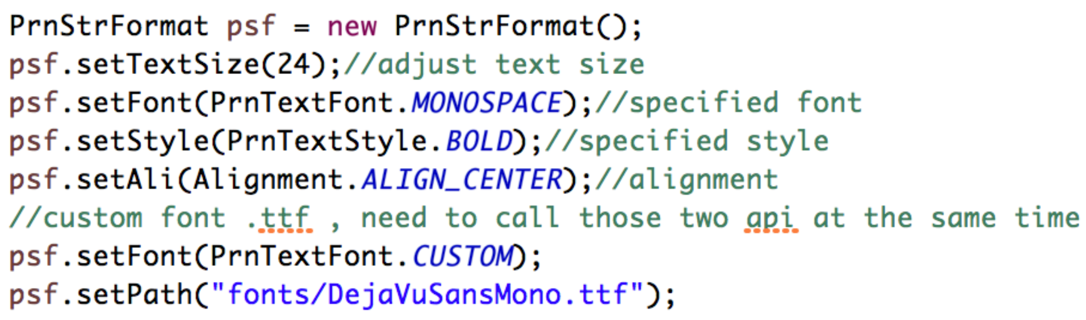
4. FingerprintListener

```
/**
     * progress of enrollment
     *
     * @param fingerId
     *            :finger ID
     * @param remaining
     *            : remaining times to finish enrollment(default 8)
     * @param reason
     *            :reason 0-successful
     *
     */
    public void onEnrollmentProgress(int fingerId, int remaining,
int reason);

    /**
     * Authenticate failed
     *
     * @param reason
     */
    public void onAuthenticationFailed(int reason);

    /**
     * Authenticate success
     *
     * @param result
     */
    public void onAuthenticationSucceeded(int fingerId, Object
obj);

/**
     * fingerprint image data
     */
    public void onGetImageComplete(String result, byte[] imgBuff);
```

5. TransListener

```
    /**
     * fire after finish nfc transaction
     */
    public void onTransCompleted(boolean isSuccessful,
           Map<String, Object> transData);
   TransData KEY List:
    public static String CARD_MASKEDPAN = "maskedPAN";
    public static String CARD_EXPIRYDATE = "expiryDate";
    public static String CARD_HOLDERNAME = "cardHolderName";
    public static String CARD_TRACK1LEN = "track1Length";
    public static String CARD_TRACK2LEN = "track2Length";
    public static String CARD_TRACK3LEN = "track3Length";
    public static String CARD_TRACK1 = "track1";
    public static String CARD_TRACK2 = "track2";
    public static String CARD_TRACK3 = "track3";
    public static String CARD_55FIELD = "iccData";
    public static String CARD_TYPE = "cardType";
    public static String CARD_BALANCE = "balance";
```

6. PrnStrFormat



```
PrnStrFormat psf = new PrnStrFormat();
psf.setTextSize(24);//adjust text size
psf.setFont(PrnTextFont.MONOSPACE);//specified font
psf.setStyle(PrnTextStyle.BOLD);//specified style
psf.setAli(Alignment.ALIGN_CENTER);//alignment
//custom font .ttf , need to call those two api at the same time
psf.setFont(PrnTextFont.CUSTOM);
psf.setPath("fonts/DejaVuSansMono.ttf");
```