



powered by

m|swipe

Mswipe Technologies PVT. LTD.
UTap Universal SDK
18th October, 2023

TABLE OF CONTENTS

- 1. INTRODUCTION
- 2. SYSTEM REQUIREMENTS
- 3. OVERVIEW
- 4. ANDROID PERMISSIONS
- 5. MSWIPE DEVICE API INTEGRATION
- 6. COMMUNICATING WITH DEVICE
 - 6.1 MSDeviceController
- 7. MSWIPE DEVICE LISTENER METHODS
- 8. MSWIPE AAR INTEGRATION
- 9. PAY BY LINK SALE

1. INTRODUCTION

This document provides UTap SDK Integration Guide for android.

2. SYSTEM REQUIREMENTS

- Development Platform: JDK: Java 1.6 or above
- OS: Android 2.3.3 or above
- CPU: 600MHz

3. OVERVIEW

- Mswipe's Device is a payment card reading device that works with POS devices. It has a magnetic card reader to read magnetic stripe cards and also a smart card reader to read EMV chip cards. All the card reader components are integrated to the onboard circuits. Mswipe device also accepts PIN from the cardholder for the EMV and MAG pin cards transactions through a secure keypad mounted on the Mswipe device.
- Mswipe device communicates with the host POS device through Serial ports. All the communication components are hard wired to the POS on board circuits.
- The sequence of steps needed for establishing the communication link between the Mobile and Mswipe device and the procedures involved for intercepting the MAG and ICC card data once the communication link has been patched and then posting the details to online gateway are briefed in detail in the following sections.

4. ANDROID PERMISSIONS

The library needs permission to use. The following lines must be added to the AndroidManifest.xml file in the app project.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" /> <Uses-
permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

5. MSWIPE DEVICE API INTEGRATION

- The API distributed through the library file (UTapUniversalSDKVer1.0.1.aar), and the other files located in the lib folder) has to be included in the projects build path, please refer the sample example code application (UTapUniversalSDKDemoApp), has of current release the SDK includes several API, described below the details of the API and the way they can be invoked from Android application, the working example demonstrates the SDK integration process and all the information related to the available functions.
- The sample application demonstrated two methods of integration, one is through **Activity** based, and the other is pure **POJO based**, the former is the quickest way of integrating the SDK, just move all the required source files and then change the UI as per the original applications themes.
- The API's, which deals with the communications between the Mswipe Device and the application are part of the 'MSDeviceController' once the communication has been established the full set of API can be leveraged to initiate the MAG and ICC transactions.
- The API's, which deals with the communications between the Mswipe Device and the application are part of the 'MSDeviceController' once the communication has been established the full set of API can be leveraged to initiate the MAG and ICC transactions.
- The entire API, which facilitates the communicating medium between the Application and Gateway are part of the class 'MSController', and the data communication between the API and the gateway are securely encrypted as per the industry standards.

6. COMMUNICATING WITH DEVICE

6.1 MSDeviceController

The MSDeviceController class processes Debit or Credit MAG and ICC card, this class has to be initialized for accommodating the acceptance of payments into the application, this will primarily interact with the Mswipe Device and detect the cards issued by different Card Schemes. And at various stages during the course of connection, to detect the card used on the device, to accept input from the user like amount for the transaction, or the card pin MSDeviceController class delegates to the listening application through various asynchronous functions.

The application has to appropriately process and provide any information to the Mswipe device any data delegated through these methods, the MSDeviceController class requests to the application. The Application can use the various convenience methods to further instruct the Mswipe Device to process and initialize accepting of payment cards.

- Define service component in your application Manifest
`<service android:name = "com.mswipetech.sdk.device.MSDeviceController"/>`

- The MSDeviceController Binding

The MSDeviceController class first has to be bound to the application for it to receive the callback and once it binds its instance the service starts. The service based approach is preferred and recommended entirely because of in-consistent behavior of the Android devices from different manufacturers. So running in a context of the Android service will make sure all the disconnection and connections to the Mswipe Device works reliably and most of the crashes are eliminated and this will ensure a smooth user experience.

This class takes care of the connections to the Mswipe Device, and does run in a separate context and will delegates back to the application through various defined callback function, for the application to listen in to this message it should registered through the interface class MSDeviceControllerResponseListener.

The android service binding to the component is asynchronous in nature and will be delegated to the component through the ServiceConnection interface, refer the below code snippet.

```
PrivateServiceConnection mMSDeviceControllerService = new ServiceConnection()
{
    Public void onServiceConnected(ComponentName className, IBinder service)

    { try

        {
```

```

LocalBinderlocalBinder = (LocalBinder) service;

mMSDeviceController = localBinder.getService();

mIsMSConnectionServiceBound = true;

//start the connection to the device asynchronously, and call backs the listeners object

//with the status of the connection

if(isConnetCalled)
{
    mAutoConnect = getIntent().getBooleanExtra("autoconnect", true);

    mCallCheckCardAfterConnection=
    getIntent().getBooleanExtra("checkcardAfterConnection",true);
}

    mMSDeviceController.initDeviceController(mMSDeviceObserver,
        mAutoConnect,      false,  mCallCheckCardAfterConnection,
        mAllowFirmwareUpdatge, DeviceCheckCardMode.SWIPE_OR_INSERT);
}

catch (Exception e) {}
}

```

```

Public void onServiceDisconnected(ComponentNameclassName)

```

```

{

```

```

// this is called when the connection with the service has been //
unexpectedly disconnected - process crashed.

```

```

        mIsMSConnectionServiceBound = false; mMSDeviceController
        = null;

    }

};

```

Refer the file MSDeviceView.java in the sample project file, for more information related to the Mswipe device service integration.

The bound service has a number of general and utility methods that manage the class itself and a number of methods that communicate with the Mswipe device through the serial port channel. When commands are sent to the App from the Mswipe Device, the MSDeviceControllerResponseListener will receive them through the delegate functions and are triggered asynchronously.

The following are the steps for binding the MSDeviceController service:

- 1) Declare an instance of the MSdDeviceController service.
- 2) Bind service by calling the below function.

```

bindService(newIntent(this,MSDeviceController.class),
mMSDeviceControllerService, Context.BIND_AUTO_CREATE);

```

- 3) Implement Service Connection.

In your implementation of onServiceConnected(), you will receive an IBinder instance (called service). Cast the returned parameter to LocalBinder type and get mMSDeviceController instance with getService().

- 4) Call various methods available in mMSDeviceController.

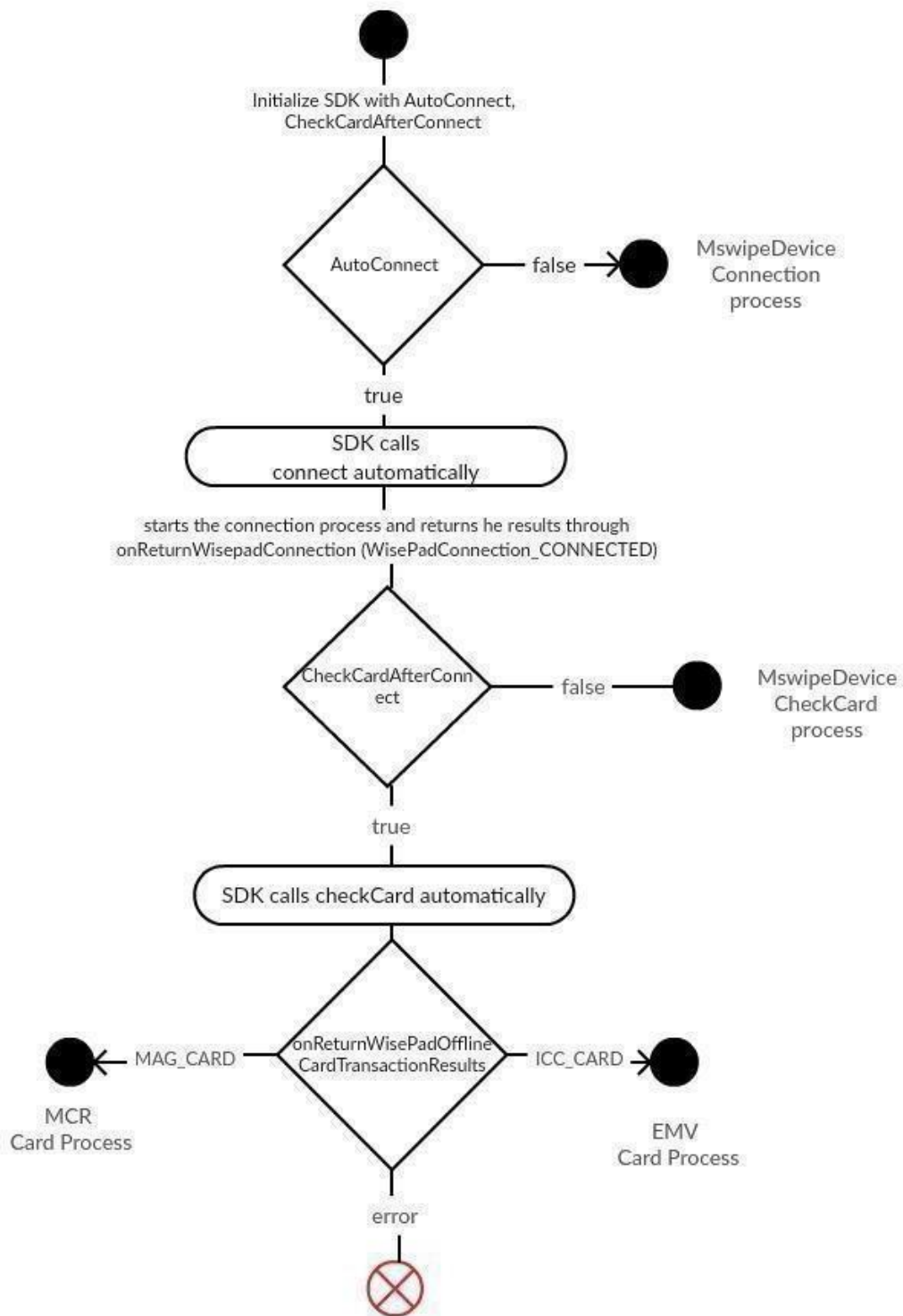
- 5) To disconnect to the device call the below function, with the instance of your interface.
- ```

unbindService(mMSDeviceControllerService).

```

- The MSDeviceController Initialization Activity Flow

## Mswipe SDK Initialization Activity Flow



- The `autoConnect` set to true when the SDK is initialized, the SDK tries to initiate the connection process to Mswipe device, this is similar to just calling the **connect** function if the connection process has to be initiated by the user, this will in a way speed up the connection process and the user does not have to call it separately.
- The `checkCardAfterConnection` set to true when the SDK is initialized, the SDK tries to call the `checkCard` function automatically when the connection to the device is established, this will change the Mswipe device to swipe/insert state, this is again similar to just calling the `checkCard` function once the connection to the device is ready.
- The Mswipe device connection process

The Mswipe Device can be connected through the serial port, when an Application is connected, the device goes to non-discoverable mode as it accepts only one connection, for any reason if the application is not able to connect, use the button next to connect the device.

## 1. Connection

As the device use the serial connection the fastest way to connect to the Mswipe device from the application is using the device. This is available to the application when a successful connection is made to the device through the listener function `onReturnDeviceConnection (DeviceConnection)`.

the application can be checked using the function `isDevicePresent`.

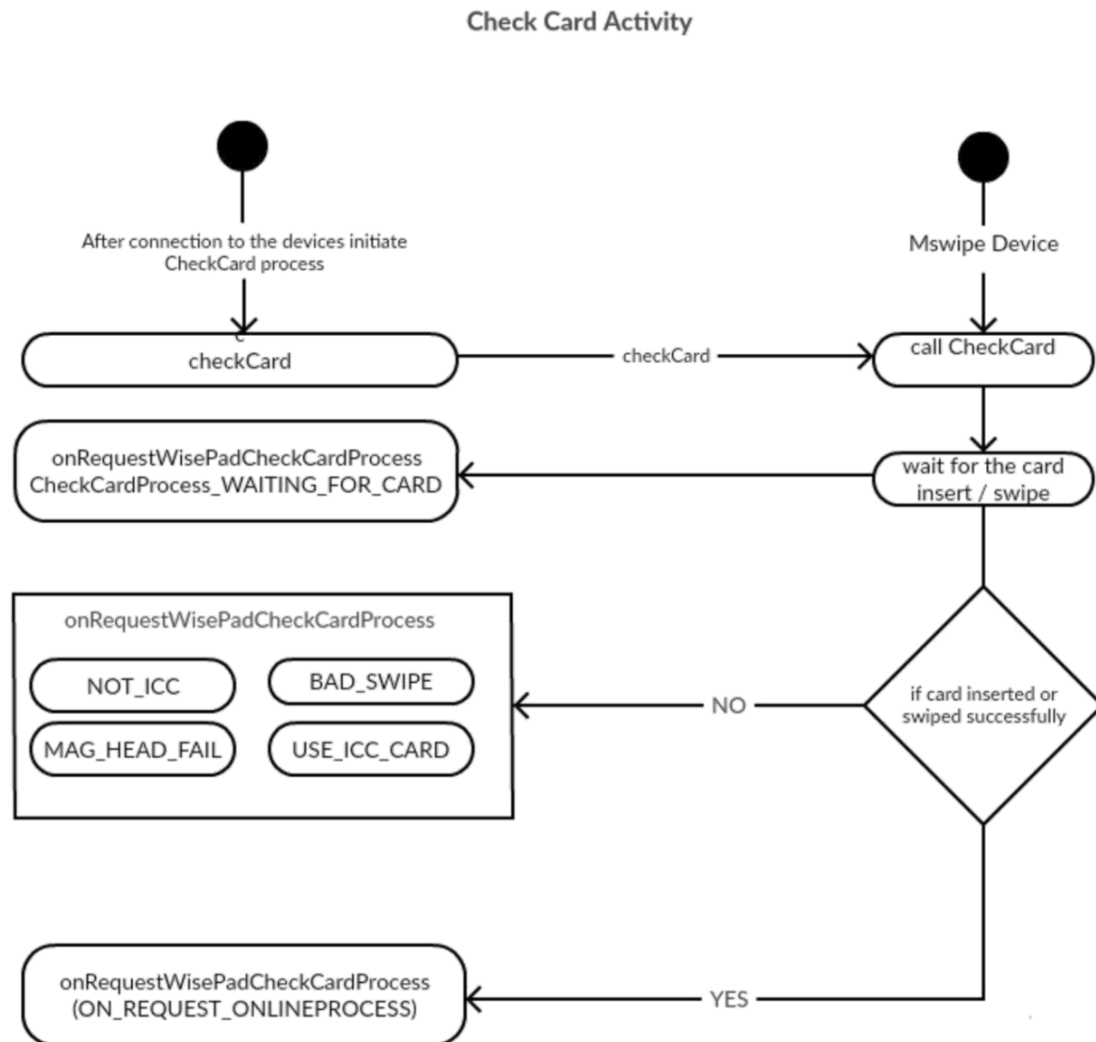
## 2. Disconnect

When the application no longer needs the connection, it should disconnect itself from Mswipe Device, the function that is available for performing this is `disconnect`. This will in fact restart the Mswipe device and flushes all the transaction from the cache, so at any time when a connection is made to the device the application has to make sure to disconnect, this is required since the device will go into un-discoverable mode until the application disconnects from this device.

The application can close itself in many ways in Android, like when the application goes to background either when another application gets a

priority for ex the phone call, or when the home button is pressed, or when it is closed specifically by the user, in this scenario the application has to make sure to disconnect from the Mswipe Device, please refer the test sample application the way this is achieved.

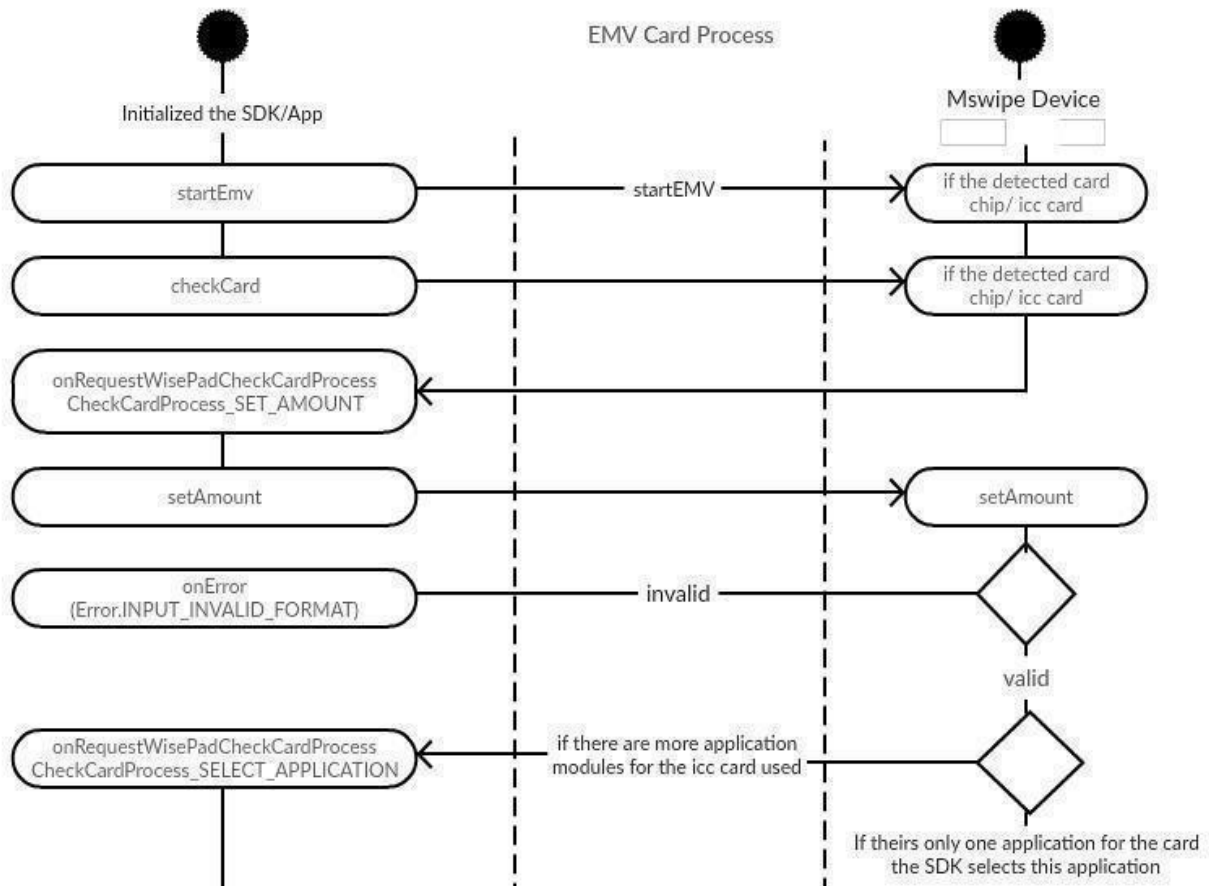
- The Check Card process



Once the connection to the device is ready, the check Card function has to be called, this will trigger the device to accept credit/debit MCR swipes, or ICC/Chip card insert, during this state if the device detects any kind of error, this message

If the card detected is MCR, its details are notified to the Application, this process is explained in detail in the MCR process section. If the card detected is ICC, its details are notified to the Application, this process is explain in detail in the EMV process section.

- The EMV process



## CheckCard

This call will trigger the device to either accept for MCR or EMV/Chip/ICC cards, and if the card is EMV/Chip/ICC all the notifications are delegated through onRequestDeviceCheckCardProcess function. **startEMV**

Instead of `checkCard`, `startEMV` can be called, this will trigger the device to accept only EMV/Chip/ICC cards.

## setAmount

A transaction amount is needed for a payment transaction and it has to be entered by the operator (or calculated by the inventory system). The `onRequestDeviceCheckCardProcess`

(`CheckCardProcess_SET_AMOUNT`) delegate method is triggered. The app should prompt the operator to enter the amount, cashback amount, currency code and transaction type and then call the `setAmount()` to send the data back to Mswipe Device.

## selectApplication

An EMV card may support multiple payment applications. The Mswipe Device reads the list of applications supported by the EMV card and asks the customer/operator to select the desired application.

The delegate method `onRequestDeviceCheckCardProcess` (`CheckCardProcess_SELECT_APPLICATION`) is triggered to return an array of application IDs. The app should prompt the user to select one application and then call the `selectApplication()` method.

## PIN Entry

An EMV card required a pin authentication it will notify through the method `onRequestDeviceCheckCardProcess` (`CheckCardProcess_PIN_ENTRY_ICC_CARD`), and at the same time the device waits for the pin input and this results are delegated back to the application through the `onReturnDeviceOfflineCardTransactionResults` (`CheckCardProcessResults.PIN_ENTERED`).

## Online Submit

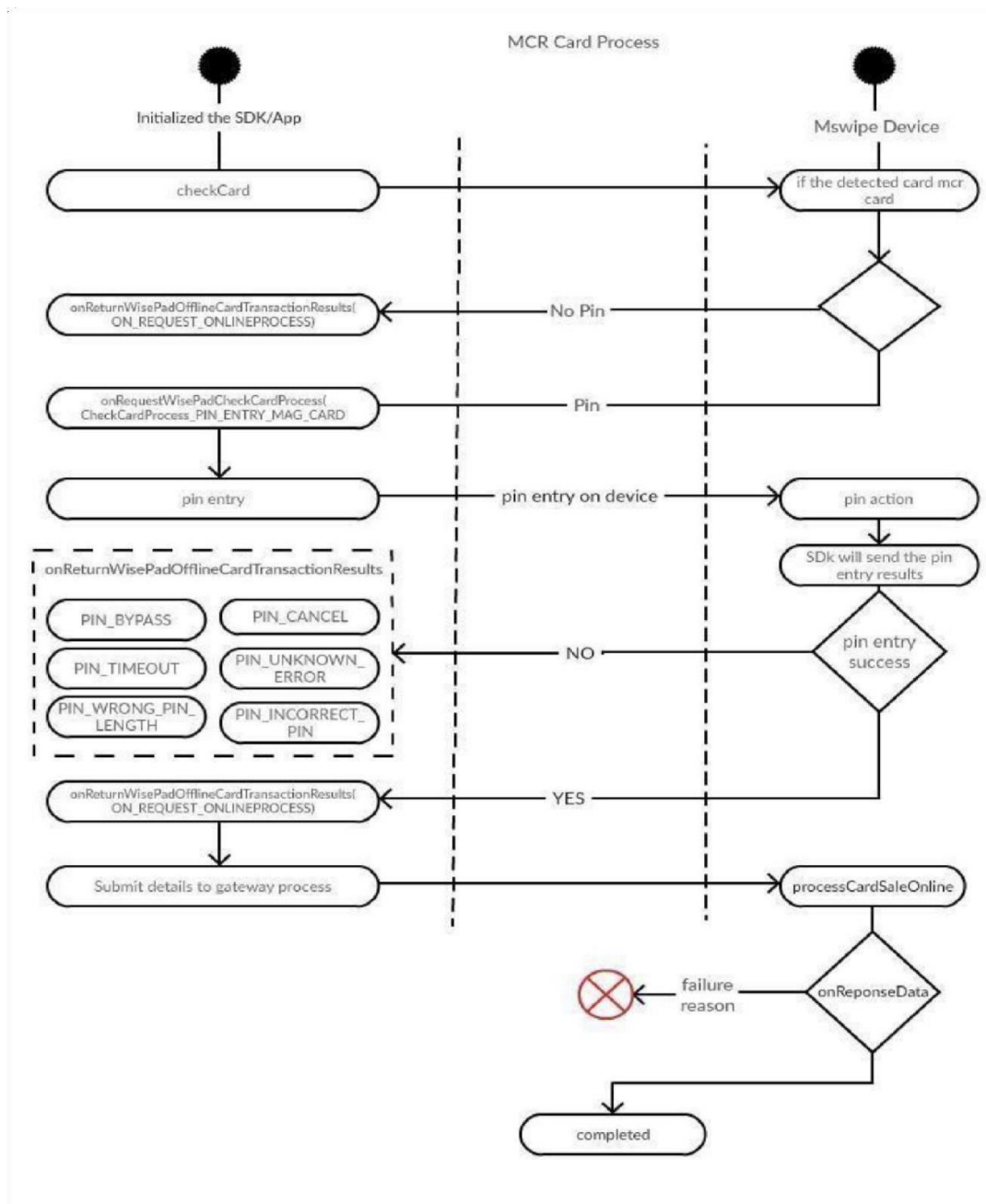
Once the card has been read successfully by the device it will send the response to the method `onReturnDeviceOfflineCardTransactionResults` (`ON_REQUEST_ONLINEPROCESS`) along with the details of the card through the auxiliary class **CardData**.

The card details can be processed further by calling an online gateway function which will validate the details by contacting the issuer bank, and the results are posted back to the caller through the method callbacks.

Use the method `processCardSaleOnline` located in the **MSDeviceController** class along with the callback method.

The results are notified back to the application through the function `onResponseData`.

- The MCR process



## CheckCard

This call will trigger the device to either accept for MCR or EMV/Chip/ICC cards, and if the card is MCR all the notifications are delegated through onRequestDeviceCheckCardProcess function.

## Pin Entry

An EMV card required a pin authentication it will notify through the method **onRequestDeviceCheckCardProcess (CheckCardProcess\_PIN\_ENTRY\_MCR\_CARD)**, and at the same time the device waits for the pin input and this results are delegated back to the application through the **onReturnDeviceOfflineCardTransactionResults (CheckCardProcessResults.PIN\_ENTERED)**.

## Online Submit

Once the card has been read successfully by the device it will send the response to the method **onReturnDeviceOfflineCardTransactionResults (ON\_REQUEST\_ONLINEPROCESS)** along with the details of the card through the auxiliary class **CardData**.

For the MCR if the card used is **Amex**, this card can be differentiated through the **enumCARDSCHMERRESULTS.MAG\_AMEXCARD, CARDSCHMERRESULTS.MAG\_CARD**, for the **Amex** card we need to accept the security code from the application, and this details have to be sent to the online gateway.

The card details can be processed further by calling an online gateway function which will validate the details by contacting the issuer bank, and the results are posted back to the caller through the method callbacks. Use the method **processCardSaleOnline** located in the **MSDeviceController** class along with the callback method.

The results are notified back to the application through the function **onResponseData**.

- The MSDeviceController Functions

This class provided various method to interact with the Mswipe device, once the container service gets initiated with the initial properties for the device which are configured through a utility function **initDeviceController**, the set of parameters will kick start the Serial communications channel from the Mobile to the Mswipe Device. Each of these methods functioning are described in detail.

| Method               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| initDeviceController | <p>Initiates the device with different properties, which will ensure the Communications between the device and the applications. Each of the properties that should be set to are</p> <ol style="list-style-type: none"> <li>1.MSDeviceControllerResponseListener, callback interface through with the device controller communicates back the state of the Mswipe Device to the listener Application.</li> <li>2.AutoConnect, this will drive the device controller to start connecting to the Mswipe device as soon as it Initializes.</li> <li>3. UseBLEConnection, not applicable for MswipeDevice.</li> <li>4. CallCheckCardAfterConnection,call up the checkCard immediately after the Mswipe Device get connected which triggers the Mswipe Device to start detecting either MAG or ICC cards.For the check card if the card to be used is ICC then the Amount has to set through the setAmount function, until this is set the checkCard or startEmv would not be initiated for the ICC card.</li> <li>5. DeviceCheckCardMode, forces the Mswipe Device to detect MAG, ICC or both when the Mswipe Device automatically calls up CheckCard.</li> </ol> |
| connect              | <p>Initiates the connection to the Mswipe Device,the connect process status are returned through the call back method onReturnDeviceConnection.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| disConnect           | <p>Disconnects the communication link to the Mswipe Device, the status related to this state is sent back to the application callback method onReturnDeviceConnection. If the connection is already close to the status, the connection already closed to the Mswipe Device is returned to the callback onError method.</p> <p>When the services which host this class are unbind will automatically disconnect the connection to the Mswipe Device, this is essential to save the battery when the Mswipe Device is no longer required.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                                                                                                                                                                                                                                                                                                                                                                                                            |
| getDeviceConnectionState  | Any time during the connection process to the Mswipe Device the class maintains the connection related information, this function will return the DeviceConnection of the Device.                                                                                                                                                                                                                          |
| getDeviceTransactionState | Once the connection link is patched to the Mswipe Device, the class registers the state of the card transaction process using the DeviceTransactionState enums.                                                                                                                                                                                                                                            |
| isDevicePresent           | Checks if the Mswipe Device holds a connection to the Mswipe Device.                                                                                                                                                                                                                                                                                                                                       |
| getDeviceInfo             | Returns a parameter object about the Mswipe device. Results are returned through the callback method onReturnDeviceInfo.                                                                                                                                                                                                                                                                                   |
| checkCard                 | <p>Initiates the Mswipe Device to Insert or swipe mode, and then appropriately the state of the Mswipe Device is delegated back to the callback onRequestDeviceCheckCardProcess method.</p> <p>If the detected card is an EMV then the startEMV will be called to initiate the EMV process, if the device specifically has to detect only the EMV card then StartEmv can be used instead of checkCard.</p> |
| cancelCheckCard           | Stop the check card process. This will be effective only before the card has been inserted or before the Mswipe Device has detected the card.                                                                                                                                                                                                                                                              |
| startEmv                  | Initiates EMV transaction. After receiving this command, Mswipe Device will take control and execute the EMV operations flow.                                                                                                                                                                                                                                                                              |
| selectApplication         | Select an application from a list of applications acceptable by the terminal and the EMV card in response to onRequestDevisedCheckCardProcess method, with the CheckCardProcess parameter set to                                                                                                                                                                                                           |

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | <p>CheckCardProcess_SELECT_APPLICATION with the list of application modules.</p> <p>If the EMV card presents with only one Application module this callback will not be triggered and it will use this particular Application internally.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| cancelSelectApplication | Cancel the process of selecting the EMV application modules. This will move Mswipe Device Transactionstate (DeviceTransactionState) to ready mode, here in this current state only CheckCard or StartEmv can be initiated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| cancelOnlineProcess     | Cancel validating referral results of the EMV card from the processor back to Mswipe Device in response to CheckCardProcessResults set to ON_REQUEST_ONLINEPROCESS, this call will essentially move the Mswipe Device to ready mode and will terminated the initiated EMV card process.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| setAmount               | <p>Set the amount required for EMV transaction, this would enable the Mswipe Device to encrypt and bundle it with the other card information and return to the application which in turn will further process this data by submitting to the online gateway.</p> <p>The amount can be set before the startEmv or CheckCard process has been initiated. In the example the connection and check card process are initiated in the background before it collect the amount information from the user, at this juncture the Mswipe Device waits at CheckCard process even though it was set to be called automatically once the Mswipe Device is connected, since the amount required is not set through this particular method, in this scenario the checkCard has to be called from the application.</p> |
| resetSetAmount          | the amount the new amount has to set through setAmount.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## 7. MSWIPE DEVICE LISTENER METHODS

MSDeviceControllerResponseListener interface will accentuate the state of the Mswipe Device to the Application, which registers itself to listen to all the available callbacks. All the callbacks are notified to the Application asynchronously. All the callbacks are grouped to represent a state of the Mswipe Device, appropriately the application has to process them accordingly. The complete list of callback are described below in the following sections, and also refer to the example Application for more information this will help in understanding the way to handle different types of callback, each of callback represent a case as the Mswipe Device goes through the different stages for processing the payment cards.

- The onReturnDeviceConnection

This callback method informs the application about the status of the Mswipe Device connection this is in response to the connect method been called, the connection states are par metered through CheckCardProcessenum, sent as first argument, and if this is associated with any data this will sent as a second parameter.

| Method                             | Description                                                                                                                                                                                  |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeviceConnection_CONNECTING        | In response to the connect method being called, this will move the Mswipe Device to the connecting state.                                                                                    |
| DeviceConnection_DIS_CONNECTED     | When the disconnect Method is called.                                                                                                                                                        |
| DeviceConnection_CONNECTED         | If the connection to the Mswipe Device is patched, the Mswipe Device moves to a state where it is ready to accept reading of the cards initiated through the checkCard or startEmv function. |
| DeviceConnection_FAIL_TO_START_BT2 | If the connection to the Mswipe Device fails.                                                                                                                                                |

- onRequestDeviceCheckCardProcess

This callback method informs the application about the status of the CheckCard process, this will get triggered when a call to checkCard or StartEmv, this will instruct the Mswipe Device to start detecting the card and collect the information of the card swiped (MAG) or Inserted (ICC).

| Method                              | Description                                                                                                                                                                                                                            |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CheckCardProcess_WAITING_FOR_CARD   | When the checkCard or startEmv are called the Mswipe Device goes into waiting for card mode and intercepts any swipes or inserts.                                                                                                      |
| CheckCardProcess_SET_AMOUNT         | Not applicable for Mswipe Device                                                                                                                                                                                                       |
| CheckCardProcess_PIN_ENTRY_ICC_CARD | If the detected card is ICC the Mswipe Device will accept the input of the card's pin through the on board keyboard mounted on the device, and the keyed pins are notified to the user through CheckCardProcessResults.PIN_Asterisk.   |
| CheckCardProcess_PIN_ENTRY_MAG_CARD | If the detected card is MAG the Mswipe Device will accept the input of the card's pin through the on board keyboard mounted on the device, and the keyed in pin are notified to the user through CheckCardProcessResults.PIN_Asterisk. |
| CheckCardProcess_SELECT_APPLICATION | If the ICC card has more than one Application module, Mswipe Device will inform the Application through this function along with the list of Applications modules.                                                                     |

- onReturnDeviceOfflineCardTransactionResults

When the Mswipe Device recognizes the card that's been swiped or inserted, it collects the results and notifies back to the Application, this callback method informs the application about the status of the CheckCard process, this will get triggered when a call to checkCard or StartEmv this will instruct the Mswipe Device to start detecting the card and collect the information of the card swiped (MAG) or Inserted (ICC).

| Method            | Description                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| NOT_ICC           | If the card detected is not an ICC card and it was inserted in the slot located on the Mswipe device.                                                |
| BAD_SWIPE         | The swipes are not recognized.                                                                                                                       |
| MAG_HEAD_FAIL     | Fails to connect to the device.                                                                                                                      |
| NO_RESPONSE       | IF the device fails to respond.                                                                                                                      |
| USE_ICC_CARD      | If the cards are provisioned with both ICC and MAC and if the user tries to use the swipe the device will prompt the user saying to use an ICC card. |
| KEY_ERROR         | Key error                                                                                                                                            |
| CANCEL_CHECK_CARD | When the cancelCheckCard is called.                                                                                                                  |
| PIN_ENTERED       | The pin input is completed.                                                                                                                          |

|                                 |                                                                   |
|---------------------------------|-------------------------------------------------------------------|
| PIN_BYPASS                      | Without any pin input on selecting the green arrow on the keypad. |
| PIN_CANCEL                      | On selecting the x on the keypad.                                 |
| PIN_TIMEOUT                     | When the pin is not provided within the specified time.           |
| PIN_KEY_ERROR                   | For any error related to the pin input.                           |
| PIN_WRONG_PIN_LENGTH            | Wrong pin length.                                                 |
| PIN_INCORRECT_PIN               | Incorrect pin.                                                    |
| PIN_UNKNOWN_ERROR               |                                                                   |
| PIN_Asterisk                    | This will notify the application when any of the keys is pressed. |
| TRANSACTION_TERMINATED          | Transaction terminated.                                           |
| TRANSACTION_DECLINED            | Transaction declined.                                             |
| TRANSACTION_CANCEL              | On canceling the transaction.                                     |
| TRANSACTION_DEVICE_ERROR        |                                                                   |
| TRANSACTION_APPLICATION_BLOCKED | Application blocked                                               |
| TRANSACTION_ICC_CARD_REMOVED    | Card removed                                                      |

|                          |                         |
|--------------------------|-------------------------|
| TRANSACTION_CARD_BLOCKED | If the card is blocked. |
|--------------------------|-------------------------|

- onError

This callback method informs the application error state, when the Mswipe Device processes the card which are either swiped or inserted during this process if the Mswipe Device fails to detect the card the relevant message is relayed back to the application through this function by setting the Error enum.

The application has to intercept this message and based on the type of the erroneous behavior of the Mswipe Device, subsequently the application has to take the necessary actions and notify to the user about this error, please refer the example file for more detailed information about the way to handle all errors that are thrown by the device.

- onRequestDisplayWispadStatusInfo

This callback method informs the application about the status of the Mswipe Device process, this delegate method provides the various states the Mswipe devices goes through and this data is only informational and can be used to present to the user, which will help to access the state of the Mswipe Device.

## 8. MSWIPE DEVICE CONTROLLER INITIATION

A singleton instantiation pattern will ensure a proper set of properties to be defined for the entire session of its existence, since properties related to Mswipe network Gateway environment and its Connection can be configured through this instance.

- Development and Live Environment

The development and the production network environment can be configured through the same SDK, by default the application connects to the development environment.

In the demo sample application the setting can be switched, this selection will appear on the title of the application suggesting the current network vertical configured.

The user can change the environment to production or development by using the “Server Environment” option from the menu list.

- MSDeviceController General Functions to Login

This API will authenticate and identify the Merchant in MSwipe database, on successful authentication the API will return back the details of the Merchant through the call back MSDeviceControllerResponseListener. The resultant responses are delegated as an Instance of LoginResponseData.

The response attributes specially the sessionTokeniser and referenceId have to be passed to the rest of the API, since the sessionTokeniser will serve as the OAUTH authentication for the rest of the API's.

This particular detail can be re-used by saving it in an in-app database, the sessionTokeniser is specific to the device and this will get renewed when UN-till the user re-login or changes his password on the same device.

Method:

```
Public void authenticateMerchant(String aMerchantId, String aMerchantPassword, final MSDeviceControllerResponseListener aMswipeDeviceControllerResponseListener)
```

Input Parameters:

1. String aMerchantId

The mobile no of the Merchant

2. String aMerchantPassword

The password of the Merchant

3. MSDeviceControllerResponseListener aMswipeDeviceControllerResponseListener  
Callback interface, which listens to all the responses of the API calls.

Returned Parameters:

The function returns the result using the asynchronous function Handler

MSDeviceControllerResponseListener as an object of class LoginResponseData

- 1) BoolResponseStatus

The Boolean value false suggests the API call failed and the reason for the failure can be known from the ResponseFailedReason.

- 2) String ResponseFailedReason

The exact failure reason

3) boolean firstTimePasswordChanged

If the returned value is true, then the merchant has to change his/her password, this will indicate that he's not changed his password since his user has been created.

4) String firstName

The first name of the Merchant logged in.

5) String referenceId

The reference Id of the Merchant.

6) String sessionTokeniser

The session identifier of the Merchant, this will be renewed when the user logs in again when he changes his password.

7) String currency

The currency denominator configured for the Merchant.

8) Boolean tipsRequired

Allows tips for the Card Sale transaction, if any tip has to be designated for the transaction this field can be used for processing the tips as an additional to the base amount.

9) float conveniencePercentage

If enabled conveniencePercentage can be levied for the transaction.

10) float serviceTax

Denotes the serviceTax to be levied for the conveniencePercentage amount

- Change Password

This API will change the password of the merchant in MSwipe database, on successful changing the password the API will return back the status through the callback `MSDeviceControllerResponseListener`. The resultant responses are delegated as an Instance of `ChangePasswordResponseData`.

Method:

```
Public void changePassword(String aMerchantId, String aSessionTokeniser, String aPassword,
String aNewPassword, final MSDeviceControllerResponseListener
aMswipeDeviceControllerResponseListener)
```

Input Parameters:

1) String a MerchantId

The mobile no of the Merchant.

2) String aPassword

The old password of the Merchant.

3) String aNewPassword

The new password of the Merchant.

4) `MSDeviceControllerResponseListener aMswipeDeviceControllerResponseListener`  
Callback interface, which listens to all the responses of the API calls.

Returned Parameters:

The function returns the result using the asynchronous function Handler

`MSDeviceControllerResponseListener` as an object of class `LoginResponseData`

1. BoolResponseStatus

The Boolean value false suggests the API call failed and the reason for the failure can be known from the `ResponseFailedReason`.

2. String ResponseFailedReason

The exact reason for failure.

### 3. String ResponseSuccessMessage

The exact reason for failure.

### 4. String sessionTokeniser

The session identifier of the Merchant, this will be renewed when the user logs in again when he changes his password.

### ● .Card Sale Transactions

This function will allow online processing of Magnetic or EMV Card transactions.

Method:

```
public void processCardSaleOnline(String aReferenceId, String aSessionTokeniser, String
aAmount, String aTipAmount, String aMobileNo, String aInvoiceNo, String aEmail, String
aNotes, boolean aIsTipEnabled, boolean aIsReceiptEnabled, String aAmexSecurityCode, double
aConveniencefeePercentage, double aServiceTaxPercentage, String aClientId, String
aExtraNote1, String aExtraNote2, String aExtraNote3, String aExtraNote4, String aExtraNote5,
String aExtraNote6, String aExtraNote7, String aExtraNote8, String aExtraNote9, String
aExtraNote10, String aStrLatitude, String aStrLongitude, final
MSDeviceControllerResponseListener aMSDeviceControllerResponseListener)
```

Input Parameters:

#### 1. String aReferenceId

Reference id of the Merchant returned from the Login API.

#### 2. String aSessionTokeniser

Session tokenizer returned from the Login API.

#### 3. String aAmount

The amount for the transaction, the string format should always have two decimals.

#### 4. String aTipAmount

The tip amount to be levied for the transactions, this will be applicable only when the Tip enabled is configured for the Merchant, the string format should always have two decimals.

#### 5. String aMobileNo

The mobile no of the purchaser, an SMS will be sent to the provided no, the mobile no has to be prefixed with the country code, for ex if the no is from India it should be sent as +91 9999999999.

#### 6. String alInvoiceNo

If Single Invoice No. Is configured for the Merchant then this has to be unique for every transaction.

#### 7. String aEmail

Email address of the purchaser, a receipt would be sent to the email provided.

#### 8. String aNotes

Any description can be attached to the transaction.

#### 9. booleansTipsEnabled

Returned from the Login API

#### 10. String aAmexSecurityCode

The security code located at the back of the AMEX card, this is applicable only for the MAG cards.

#### 11. float aConveniencefeePercentage

Returned from the Login API

#### 12. float aServiceTaxPercentage

Returned from the Login API

13. float aServiceTaxPercentage,

Returned from the Login API

14. String aClientId

ClientId of the purchaser, need to send if required otherwise can send empty string for this.

15. String aExtraNotes1 to aExtraNotes1

Extra pre-reserved fields for the feature enhancement.

16. MSDeviceControllerResponseListener aMswipeDeviceControllerResponseListener  
Callback interface, which listens all the response of the API call.

Returned Parameters:

The function returns the result using the asynchronous function Handler  
MSDeviceControllerResponseListener as an object of class CardSaleResponseData

1. BoolResponseStatus

The Boolean value false suggests the API call failed and the reason for the failure can be known from the ResponseFailedReason.

2. String ResponseFailedReason

The exact reason for failure.

3. String EmvCardExpdate

The card expiry data of the EMV Card

4. String StandId

5. String AuthCode

6. String RRNO

7. String Date

Transaction date.

8. String TID

Merchant terminal ID.

9. String MID

Merchant ID.

10. String BatchNo

Batch process No for the EOD processes.

11. String VoucherNo

12. String TrxAmount

The transaction amount.

- Preauth Sale Transactions

This function will allow online processing of Magnetic or EMV Card transactions.

Method:

```
public void processPreauthSaleOnline(String aReferenceId, String
aSessionTokeniser, String aAmount, String aTipAmount, String aMobileNo,
String aInvoiceNo, String aEmail, String aNotes,
boolean aIsTipEnabled, boolean aIsReceiptEnabled, String aAmexSecurityCode,
String aClientId, String aExtraNote1, String aExtraNote2, String aExtraNote3,
String aExtraNote4, String aExtraNote5, String aExtraNote6, String aExtraNote7,
String aExtraNote8, String aExtraNote9, String aExtraNote10, String aStrLatitude,
String aStrLongitude, final MSDeviceControllerResponseListener
aMSDeviceControllerResponseListener)
```

Input Parameters:

1. String aReferenceId

Reference id of the Merchant returned from the Login API.

2. String aSessionTokeniser

Session tokenizer returned from the Login API.

3. String aAmount

The amount for the transaction, the string format should always have two decimals.

4. String aTipAmount

The tip amount to be levied for the transactions, this will be applicable only when the Tip enabled is configured for the Merchant, the string format should always have two decimals.

5. String aMobileNo

The mobile no of the purchaser, an SMS will be sent to the provided no, the mobile no has to prefixed with the country code, for ex if the no is from India it should be sent as +919999999999

6. String aInvoiceNo

If Single Invoice No. Is configured for the Merchant then this has to be unique for every transaction.

7. String aEmail

Email address of the purchaser, a receipt would be sent to the email provided.

8. String aNotes

Any description can be attached to the transaction.

9. boolean aIsTipEnabled,

Returned from the Login API

10. String aAmexSecurityCode

The security code located at the back of the AMEX card, this is applicable only for the MAG cards.

11. String aClientId

ClientId of the purchaser, need to send if required otherwise can send empty string for this.

12. String aExtraNotes1 to aExtraNotes10

Extra pre-reserved fields for the feature enhancement.

13. MSDeviceControllerResponseListener aMSDeviceControllerResponseListener  
Callback interface, which listens all the response of the API call.

Returned Parameters:

The function returns the result using the asynchronous function Handler  
MSDeviceControllerResponseListener as an object of class CardSaleResponseData

1. BoolResponseStatus

The Boolean value false suggests the API call failed and the reason for the failure can be known from the ResponseFailedReason.

2. String ResponseFailedReason

The exact reason for failure.

3. String EmvCardExpdate

The card expiry data of the EMV Card

4. String StandId

5. String AuthCode

6. String RRNO

7. String Date

Transaction date.

8. String TID

Merchant terminal ID.

9. String MID

Merchant ID.

10. String BatchNo

Batch process No for the EOD processes.

11. String VoucherNo

12. String TrxAmount

The transaction amount.

● Void Trx

This API will get the transaction details of the transaction.

MSDeviceControllerResponseListener the resultant responses are delegated as an Instance of TransactionDetailsResponseData.

Method:

```
Public void getCardSaleTrxDetails(String aMerchantId, String aSessionTokeniser, String aTrxDate, String aTrxAmount, String aLast4Digits, final MSDeviceControllerResponseListener aMswipeDeviceControllerResponseListener)
```

Input Parameters:

1. String aMerchantId

The mobile no of the Merchant

2. String aSessionTokeniser

Session tokenizer returned from the Login API.

3. String aTrxDate

The current date in the format of yyyy-mm-dd

4. String aTrxAmount

The amount for the transaction, the string format should always have two decimals.

5. String aLast4Digits

The last four digits of card number.

6. MSDeviceControllerResponseListener Callback interface, which listens to all the response of the API calls.

Returned Parameters:

The function returns the result using the asynchronous function Handler  
MSDeviceControllerResponseListener as an object of class TransactionDetailsResponseData

1. BoolResponseStatus

The Boolean value false suggests the API call failed and the reason for the failure can be known from the ResponseFailedReason.

2. String ResponseFailedReason

The exact reason for failure.

3. TrxDate

4. StanNo
5. VoucherNo
6. CardLastFourDigits
7. TrxAmount
8. AuthNo
9. RRNO

This API void the transaction.

#### 10. MSDeviceControllerResponseListener

The resultant responses are delegated as an Instance of TransactionDetailsResponseData.

Method:

```
Public void processVoidTransaction(String aMerchantId, String
aSessionTokeniser, String aTrxDate, String aTrxAmount, String aLast4Digits,
String aStanId, String aVoucherNo, String aClientId, final
MSDeviceControllerResponseListener aMSDeviceControllerResponseListener)
```

Input Parameters:

1. String aMerchantId  
The mobile no of the Merchant
2. String aSessionTokeniser  
Session tokenizer returned from the Login API.
3. String aTrxDate  
The current date in the format of yyyy-mm-dd
4. String aTrxAmount  
The amount for the transaction, the string format should always have two decimals.
5. String aLast4Digits  
The last four digits of card number.
6. String aStanId

stanID that will returned from the getCardtransactionDetails API

#### 7. String aVoucherNo

VoucherNo that will returned from the getCardtransactionDetails API

#### 8. String aClientId

ClientId of the purchaser, need to send if required otherwise can send empty string for this.

9. MSDeviceControllerResponseListener Callback interface, which listens to all the response of the API calls.

#### Returned Parameters:

The function returns the result using the asynchronous function Handler MSDeviceControllerResponseListener as an object of class VoidTransactionResponseData

##### 1. BoolResponseStatus

The Boolean value false suggests the API call failed and the reason for the failure can be known from the ResponseFailedReason.

##### 2. String ResponseFailedReason

The exact reason for failure.

##### 3. TrxDate

##### 4. CardType

This API will get the transaction details of the transaction.

MSDeviceControllerResponseListener: The resultant responses are delegated as an Instance of TransactionDetailsResponseData.

## 8. MSWIPE AAR INTEGRATION

AAR distribution are bundled with the following files mswipesdk. To call up the services of Mswipe SDK, the aar UTapUniversalSDKVer1.0.1.aar has to be added to libs folder of your project. With this integration we just need to call the card sale section through Intent call, from there Mswipe will take care of the device

connection and transaction process and send the response to the users. With this option we cannot change the UI theme. The demo sample application demonstrates the integration process and also the mechanism of the calling the Card Sale process.

### Cardsale Process:

This will allow it to perform the Magnetic Card transaction and EMV transactions, and would communicate back the status of the transaction to the host application through the method onActivityResult(). And the request data as followed

### Request

```
Intent intent = new Intent(ARRActivityPaymentView.this, MSAARHandlerActivity.class);

intent.setType(MS_CARDSALE_ACTIVITY_INTENT_ACTION); intent.putExtra("username",
 "1234567980"); intent.putExtra("password", "111111");
 intent.putExtra("production", false); intent.putExtra("Amount", "10.00")
 intent.putExtra("MobileNumber", "2222222222"); intent.putExtra("Reciept",
 "reciept1111");
 intent.putExtra("Notes", "");
intent.putExtra("MailId", ""); intent.putExtra("extra1", ""); intent.putExtra("extra2",
 "");
 intent.putExtra("extra3", ""); intent.putExtra("extra2", ""); intent.putExtra("orientation",
"auto"); intent.putExtra("isSignatureRequired", "false"); intent.putExtra("isPrinterSupported",
false); intent.putExtra("isPrintSignatureOnReceipt", false); startActivityForResult(intent, "1001")
```

| Type   | Parameter Name | Default Value | Comments |
|--------|----------------|---------------|----------|
| String | username       |               |          |
| String | password       |               |          |

|         |              |  |                                                                                             |
|---------|--------------|--|---------------------------------------------------------------------------------------------|
| Boolean | production   |  | Pass “true” if you want to connect to the production server and “false” for the lab server. |
| String  | Amount       |  |                                                                                             |
| String  | MobileNumber |  |                                                                                             |
| String  | Receipt      |  | The order id, this parameter will get reflected in the ERP for this particular transaction. |
| String  | MailId       |  |                                                                                             |
| String  | Notes        |  | Additional information, which will be logged for the transaction in the ERP                 |
| String  | extra1       |  | Additional information, which will be logged for the transaction in the ERP                 |
| String  | extra2       |  | Additional information, which will be logged for the transaction in the ERP                 |
| String  | extra3       |  | Additional information, which will be logged for the transaction in the ERP                 |
| String  | orientation  |  | Pass portrait/landscape/auto                                                                |

|         |                    |  |                                                                       |
|---------|--------------------|--|-----------------------------------------------------------------------|
| Boolean | isPrinterSupported |  | Pass "true" in case receipt printing is required or else pass "false" |
|---------|--------------------|--|-----------------------------------------------------------------------|

## Response:

The details are notified back to the host application through onActivityResult() method with

onActivityResult :

```
Protected void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);
```

```
if(requestCode == 1001)
```

```
{
```

```
if(resultCode == RESULT_OK)
```

```
{
```

```
boolean status = data.getBooleanExtra("status", false);
```

```
String statusMessage = data.getStringExtra("statusMessage"); String AuthCode =
data.getExtras().getString("AuthCode"); String RRno =
data.getExtras().getString("RRNo"); }
```

```
Else{
```

```
boolean status = data.getBooleanExtra("status", false);
```

```
String statusMessage = data.getStringExtra("statusMessage");
```

```
}
```

}

}

| Type    | Parameter Name | Default Value | Comment                    |
|---------|----------------|---------------|----------------------------|
| Boolean | status         |               | Transaction status         |
| String  | statusMessage  |               | Transaction status message |
| String  | errNo          |               | Response code              |

|        |               |  |                          |
|--------|---------------|--|--------------------------|
| String | AuthCode      |  |                          |
| String | RRNo          |  |                          |
| String | TVR           |  |                          |
| String | TSI           |  |                          |
| String | receiptDetail |  | Transaction receipt data |

The exact reason for failure.

## 9. PAY BY LINK SALE

To generate PaybyLink:

```
generatePayByLink(String aReferenceld, String aSessionTokeniser, String aCustCode, String aAmount, String aMobileNo, String aEmailId, String alnvoiceNo, String aMerchantTid, String aExtraNote1, String aExtraNote2, String aExtraNote3, String aExtraNote4, String aExtraNote5,String aExtraNote6, String aExtraNote7, String aExtraNote8, String aExtraNote9, String aExtraNote10,final MSDeviceControllerResponseListener aMSDeviceControllerResponseListener)
```

On Success:

A link will be sent to mobile no and email , through which the payment can be done .And in onResponseData() , will receive response in PayByLinkResponseData,in which have Invoice ID, that can be used for further response.

### To Check status:

```
public void checkPayByLinkStatus(String aReferenceld, String aSessionTokeniser, String aTrnxId, final MSDeviceControllerResponseListener aMSDeviceControllerResponseListener) onSuccess: onResponseData,will receive response in PayByLinkResponseData , in which status is obtained
```

### To Get PaybyLink TRnx List:

```
public void getPayByTrxList(String aReferenceld, String aSessionTokeniser, String aCustCode, final MSDeviceControllerResponseListener aMSDeviceControllerResponseListener)
```

onSuccess:

onResponseData , will receive response through PayByLinkResponseData , in which it contains arraylist<PayByLinkTrxData> of all pay by link transations .

or Bank Mapping:

```
To get BankList: public void getIPGBankList(String aReferenceld, String aSessionTokeniser, final MSDeviceControllerResponseListener aMSDeviceControllerResponseListener)
```

onSuccess:

onResponseData , will receive response through IPGBankResponseData , in which it contains arraylist<IPGBankListData > of all Banks.

**For OTP generation:** public void sendIPGBankOTP(String aReferenceld, String aSessionTokeniser, final MSDeviceControllerResponseListener aMSDeviceControllerResponseListener )

onSuccess:

otp will be sent to registered mobile no ,which should be used for authentication in order to map the particular bank .

### **To Update BankList :**

**public void** updateIPGBankDetails(String aReferenceld, String aSessionTokeniser, String aBankCode, String aLocationCode , String aOtp , **final** MSDeviceControllerResponseListener aMSDeviceControllerResponseListener )

onResponseData,will receive a response inIPGResponseData, in which status is obtained. On success, your bank will be mapped to the particular user.