POSitive Integration Manual

Document Version: 0.0.1
POSitive Application Version: 0.0.1

last updated 19/07/2019

by

EFT Solutions Ltd
www.eft-solutions.co.uk

Table of Contents

1. Confidentiality Statement

warren@eft-solutions.co.uk

2. Change History

Date	Who	Ver	Change Description
19/7/19	Philip Clarkson	0.0.1	Initial Draft

3. Glossary

Term	Definition
UTI	Unique Transaction Identifier (used for transaction lookup)
MSR	Mag Stripe Transaction (Card Swiped)
CTLS	Contactless Transaction (Card Tapped)
EMV	Insert card transaction
Reversal	Used for a full cancellation of a previous transaction
Refund	Used to credit a cardholders account (Not linked to previous sale)
Sale	A purchase transaction that debits the cardholders account

4. Introduction

This document is intended to be read by application integrators, who want to integrate card payments into their Android-based Point of Sale application via the POSitiveLib library.

POSitiveLib provides an easy-to-use API interface to initiate card transactions, in a java library.

5. A920 3rd Party Integration

This document provides the instructions for the POSitiveLauncher demo application required for 3rd party developers to write applications that integrate with POSitive on the Pax A920 payment terminal.

1. Prerequisites

1. A Pax A920 payment terminal.
2. A USB cable (USB-A to USB micro-B).
3. The terminal must be installed with at least version 1.00.00 of the POSitive payment application, ideally configured in demo mode to assist development.
4. The terminal must be put into debug mode using the Pax website (PAX developers should know how to do this, so instructions have not been added, see Pax for more details)

2. POSitiveLauncher – Sample Application

Unzip and import the POSitiveLauncher package into Android Studio.

1. The application provides a very simple program that will allow you to see how a transaction and a reversal should be performed. Test buttons are provided on the main activity (see MainActivity.java)
2. Comments have been added in the code to explain how it works, but it should be quite self-explanatory
3. The code relies on a library that is included called POSitivelib-X.X-release.aar. Look at the manifest and the build.gradle files to see how it is integrated.
4. The IPC comms rely on a BroadcastReceiver being used to send/receive the messages. Look at the manifest to see how this should be declared. The example implementation is in TestLaunchReceiver.java
5. It is the responsibility of the calling app to return itself to the foreground once the transaction is completed (The sample app also does this in the POSitiveLauncReceiver by calling startActivity() once the result has been received)
6. EFT Solutions intend to update the POSitiveLib with more functionality as required, so more details of the transaction will be added as requested. Please email specific requests/comments to philip@eft-solutions.co.uk

In Summary: you should be able to declare a receiver and use the included library to do IPC comms to and from the POSitive app (Using this app as a working example)

3. Terminal Profile

There is a config file on the terminal called profile.xml. This file determines which apps are on display from the main menu.

There is an example of the proile.xml file containing POSitiveLauncher in the root directory of the test app.

The file needs to contain the name you want to display on the menu and the package name, so that the main service can identify your application.

E.g.

```
<menuItem>
   <packageName>eft.com.positivelauncher</packageName>
   <displayName>Test Launcher</displayName>
</menuItem>
```

You can update the file and install it to display your app using the following adb.exe command. In a live environment this configuration will be pushed down from the TMS.

```
adb.exe push profile.xml /data/data/eft.com.positivesvc/files
```

4. Integration Rules

1. The apps must not bring down the network stack. Both the payment application and the paxstore app require internet connectivity to upload transactions in the background.

2. New applications must not install themselves as a launcher process

5. Transaction Initiation

To initiate a transaction, call the API functions on the PositiveLib class.

Declarations:

- **boolean** runTrans(Context context, **int** amount, String transType);
- **boolean** runTrans(Context context, **int** amount, String transType, HashMap<String, String> args);
- **boolean** runReversal(Context context, **int** amount, **int** receiptNumberForReversal) ;
- **boolean** runReversal(Context context, **int** amount, String utiForReversal, **boolean** disablePrinting, **boolean** silent)

For Example.

- SALE:
  - *PositiveLib.runTrans(this, 100, TRANSACTION_TYPE_SALE);*
- REFUND:
  - *PositiveLib.runTrans(this, 100, TRANSACTION_TYPE_REFUND);*
- REVERSAL:
  - *PositiveLib.runReversal(this, 100, 0);*

You can reverse transactions based on their UTI, or just reverse the last transaction by

passing in a receipt number of 0. You can also use the additional last two arguments to disable printing, and keep the reversal silent (in the background)

To future-proof the API, an additional runTrans function has also been added:

boolean runTrans(Context context, int amount, String transType, Hashmap<String, String> args)

This allows us to pass additional custom name-value pair arguments to the transaction with new releases of the library, for example user ID/password etc

Purchase With Cashback: The sale transaction will offer cashback where the terminals configuration indicates that cashback is supported for the card type presented. If you want to enable\disable this feature use paxstore to update your terminals settings.

6. Transaction Results

Different results can come back from the POSitive app based on the transaction result.

They are split into four separate lists, with a flag that can be checked to indicate if the list is present.

E.g.

For a successful CTLS transaction, lists 1-3 would be returned.

For a successful MSR transaction, lists 1 and 2 would be returned.

For a critical failure resulting from a crash or a programming error then list 4 would be returned.

List One is returned when the “Extra” from the intent contains **TransResponse** = true

List Two is returned when the “Extra” from the intent contains **TransactionDetails** = true

List Three is returned when the “Extra” from the intent contains **CardType** = EMV or CTLS

List Four is returned when there is a serious failure, the “Extra” from the intent will contain **TransResponse** = false;

The broadcast receiver has code to demonstrate extracting the results for the calling app to use.

1. **List One - Standard Response Details (TransResponse = true)**

Field Name	**Date Type**	**Description**
UTI	GUID	Unique Transaction Identifier (e.g. 5594801e-a3e5-da11-8b4600065b3e6c8d)
Amount	int	Amount of the transaction (minor units, 100 = £1.00)
Tip	int	The tip amount (minor units, 100 = £1.00)

Cashback	int	The cashback amount (minor units, 100 = £1.00)
Approved	Boolean	Transaction result
Cancelled	Boolean	Set when the user manually cancels the transaction.
SigRequired	Boolean	Indicator to let called know if signature is required
PINVerified	Boolean	Indicator to say if the pin was verified
Currency	String	3 character currency code (GBP/EUR etc)
Tid	String	The terminal soft ID (e.g. 12345678)
Mid	String	The merchant ID (e.g. 123456789012345)
Version	String	The software version (e.g. 1.00.00)

2. **List Two - Transaction Details**

Field Name	**Data Type**	**Description**
ReceiptNumber	int	The number of the receipt
RRN	String	The retrieval reference number (matches the RRN for the protocol being used)
ResponseCode	String	The response code from the acquirer (e.g. 00 = Approved)
Stan	String	The stan used in the protocol messages
AuthCode	String	The auth code returned from the acquirer (e.g. 123ABC)
MerchantTokenId	String	Token from the gateway (if sent)
PAN	String	The masked pan of the card (e.g. 545454******5454)
ExpiryDate	String	The expiry date of the card used – YYMM
StartDate	String	The start date of the card used (if available) – YYMM
Scheme	String	The name of the scheme used (e.g. Visa\Mastercard)
PSN	String	The pan sequence number (e.g. 001)
CardType	String	The capture method of the card (EMV/MSR/Contactless/Manual)

3. **List Three - EMV Details**

Field Name	**Data Type**	**Description**

AID	String	The application identifier (e.g. A0000000031010)
TSI	String	Transaction Status Information (e.g. E800)
CardHolder	String	The cardholder name (if available, e.g. P CLARKSON)
Cryptogram	ByteArray	The cryptogram used (e.g. 8754EA78EB65AB65)
CryptogramType	String	The type of the cryptogram (e.g. AAC/TC/ARQC)

4. **List Four – Error Details**

Field Name	**Data Type**	**Description**
TransResponse	Boolean	Always false as there has been an error
Approved	Boolean	Always false as there has been an error
Error	Enum	See the PositiveErrors enum type in the library
ErrorText	String	Additional error information