POSitive Integration Manual

Document Version: 0.0.4
POSitive Application Version: 0.0.4

last updated 31/03/2020

by

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

Table of Contents

• Confidentiality Statement

warren@eft-solutions.co.uk

• Change History

Date	Who	Ver	Change Description
19/7/19	Philip Clarkson	0.0.1	Initial Draft
21/08/19	Philip Clarkson	0.0.2	Updated to use POSIntegrate (new interface)
09/09/19	Sangeeta Panda	0.0.3	Status Events Updated
31/03/2020	Phil Clarkson	0.0.4	Terminal Profile section updated

• 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
PreAuth	A transaction to reserve funds on an account
Completion	A transaction to complete a previous pre-auth and transfer the funds permanently

• 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.

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.

Prerequisites

- A Pax A920 payment terminal.
- A USB cable (USB-A to USB micro-B).
- 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.
- 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)

POSitiveLauncher – Sample Application

Unzip and import the POSitiveLauncher package into Android Studio.

- 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)
- Comments have been added in the code to explain how it works, but it should be quite self-explanatory
- 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.
- 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
- 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)
- 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)

Terminal Profile

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

This file is delivered to the terminal through the resource packages.

The app being developed must be added to the resource package for each customer.

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

E.g.

```
{
  "displayName": "POSitive Demo",
  "packageName": "com.eft.positivelauncher"
},
```

Please provide this information to PAX UK and ask for an updated resource package to be sent to the development terminal.

• 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

• Transaction Initiation

To initiate a transaction, call the API functions on the PosIntegrate class. The methods take additional configuration, passed in as a HashMap to the function.

The HashMap is populated with additional values identified by the enum CONFIG_TYPE on PosIntegrate.java

Declarations:

- PositiveError **executeTransaction**(Context context, TRANSACTION_TYPE transType, HashMap<CONFIG_TYPE, String> args)
- PositiveError **executeReversal**(Context context, HashMap<CONFIG_TYPE, String> args)

For Example.

This will add an amount to the hashmap of 100 (minor units)

```
HashMap<CONFIG_TYPE, String> args = new HashMap<CONFIG_TYPE, String>();
args.put(CT_AMOUNT, "100");
```

- SALE:
  - *PosIntegrate* . executeTransaction *(this, TRANSACTION_TYPE_SALE, args);*

- REFUND:
    - *PosIntegrate .* executeTransaction *(this, 100, TRANSACTION_TYPE_REFUND, args);*
- REVERSAL:
    - *PosIntegrate .executeReversal(this, args);*
- CANCEL:
    - *PosIntegrate .* executeTransaction *(this, TRANSACTION_TYPE_SALE, args);*
- QUERY:
    - *PosIntegrate.queryTransaction(this.args);*

Reverse transactions based on their UTI, or just reverse the last transaction by passing in a receipt number of 0.
Pass additional arguments to disable printing, and keep the reversal silent (in the background)

Cancel Transaction takes CT_CANCELLED_TIMEOUT as part of argument to cancel the transaction after the entered time.

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. You can also pass in a cashback or gratuity amount and the app will use it if allowed by the card.

• Transaction Results

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

The following method is used to unpack the results into an object.

- PositiveTransResult **unpackResult**(Context context, Intent intent)

The results on the object are split into four separate lists, with a flag that can be checked on the object 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 **transResponse** boolean = true

List Two is returned when the **transDetails** boolean = true

List Three is returned when the **cardType** String = “EMV” or “CTLS”

List Four is returned when there is a serious failure and **transResponse** = false;

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

- **List One - Standard Response Details (transResponse = true)**

Field Name	**Date Type**	**Description**
`UTI`	GUID	Unique Transaction Identifier (e.g. 5594801e-a3e5-da11-8b4600065b3e6c8d)
`amountTrans`	long	Amount of the transaction (minor units, 100 = £1.00)
`amountGratuity`	long	The tip amount (minor units, 100 = £1.00)
`amountCashback`	long	The cashback amount (minor units, 100 = £1.00)
`transApproved`	Boolean	Transaction result
`transCancelled`	Boolean	Set when the user manually cancels the transaction.
`cvmSigRequired`	Boolean	Indicator to let called know if signature is required
`cvmPinVerified`	Boolean	Indicator to say if the pin was verified
`transCurrencyCode`	String	3 character currency code (GBP/EUR etc)
`terminalId`	String	The terminal soft ID (e.g. 12345678)
`merchantId`	String	The merchant ID (e.g. 123456789012345)
`softwareVersion`	String	The software version (e.g. 1.00.00)

- **List Two (transDetails = true)**

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

receiptNumber	int	The number of the receipt
retrievalReferenceNumber	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
authorisationCode	String	The auth code returned from the acquirer (e.g. 123ABC)
merchantTokenId	String	Token from the gateway (if sent)
cardPan	String	The masked pan of the card (e.g. 545454******5454)
cardExpiryDate	String	The expiry date of the card used – YYMM
cardStartDate	String	The start date of the card used (if available) – YYMM
cardScheme	String	The name of the scheme used (e.g. Visa\Mastercard)
cardPanSequenceNumber	String	The pan sequence number (e.g. 001)
cardType	String	The capture method of the card (EMV/MSR/Contactless/Manual)

- **List Three - EMV Details**

Field Name	**Data Type**	**EMV tag data source**	**Description**
emvAid	String	9f06	The application identifier (e.g. A0000000031010)
emvTsi	String	9b	Transaction Status Information (e.g. E800)
emvCardholderName	String	5f20	The cardholder name (if available, e.g. P CLARKSON)
emvCryptogram	ByteArray	9f26	The cryptogram used (e.g. 8754EA78EB65AB65)
emvCryptogramType	String	Derived	The type of the

		from 9f27	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

• Transaction Reports

To initiate a report, call the following API function on the PosIntegrate class.
The HashMap is populated with the report type

Declarations:

- PositiveError **executeReport**(Context context, TRANSACTION_TYPE transType, HashMap<CONFIG_TYPE, String> args)
- PositiveError **unpackReport**(Context context, Intent intent)

For Example.
This will add the report type to the hashmap

```
HashMap<CONFIG_TYPE, String> args = new HashMap<CONFIG_TYPE, String>();
args.put(CT_XREPORT, "TRUE");
```

- X REPORT:
  - *PosIntegrate.executeReport(this, TRANSACTION_TYPE_RECONCILIATION, args);*
- RUN REPORT:
  - *PosIntegrate .* executeReport *(this, args);*
- UNPACK REPORT RESULTS (from receiver)
  - *PosIntegrate .* unpackReport *(content, intent);*

The report should be printed by the terminal and the results returned to the broadcast receiver. The unpackreport method can be used to extract the results into an object.

• Status Events

There are different status events declared in POSitive app.

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

Filter TRANSACTION_STATUS_EVENT in onReceive() of the PositiveLaunchReceiver and receive the status events using below code:

```
String statusEvent = intent.getStringExtra("StatusEvent");
```

List of Status events:

1. Transaction started
2. Transaction Approved
3. Transaction Declined
4. Card type = MSR
5. Card type = EMV
6. Card type = CTLS
7. Card type = manual
8. Transaction Cancelled
9. Transaction Referred
10. Transaction Finished
11. GetCard Screen Displayed
12. Manual Pan Screen Displayed
13. Pin Requested(Offline)
14. Pin Requested(Online)
15. Host Approved
16. Reversal Approved
17. Reversal Declined
18. Transaction Declined
19. Card User Cancelled
20. Printer General Error
21. Printer Out Of Paper
22. Amount High
23. Amount Low
24. Card Blocked
25. Card Expired

26. Pin Invalid Retry
27. Pin Invalid Last Try
28. Cashback Too High
29. Pin Cvm Required
30. Signature Cvm Required
31. Locally Declined
32. Host Declined
33. Issuer Declined
34. Issuer Unavailable
35. Update In Progress Error
36. Update Required Error
37. Reversal Not Possible Error
38. Transaction Type Not Allowed
39. Login Failed
40. Chip Unreadable
41. Chip App Unsupported Please Swipe
42. Chip Rid Unsupported Please Swipe
43. Chip Invalid Please Swipe
44. Chip not allowed Please swipe
45. Chip detected Please Insert
46. Chip detected Please Insert OR Force Fallback
47. Insert Or Swipe Card
48. Magnetic Strip Unreadable
49. Magnetic Stripe Invalid
50. Magnetic Stripe Not Allowed
51. Manual Input Invalid
52. Manual Input Invalid Length
53. Manual Input Invalid Date
54. Cashback Only Allowed Online
55. Transaction Only Allowed Online

56. Approval Code Invalid
57. Password Invalid
58. Close Batch Required
59. Close Batch Not Required
60. Technical Error
61. Hardware Error

After getting results back from the POSitive app , there are options to view transaction results and status events.