Network International

N-Genius EPOS Specification

Issue date	05/04/2024
Version	2.2
Author	Network International

Document Approval

Version History

Version #	Dated	Updated by	Change
1.0	30/9/2018	Network International	
1.1	04/04/2023	Network International	- Reviewed and added missing sections
2.0	05/04/2024	Network International	- Updated Flow diagrams - Mandatory Section 6.6 added
2.1	20/05/2024		- Added New Section 6.5(EposMultiMid)
2.2	23/05/2024	Network International	- Added New Section 6.5.2(EposMultiMidRefund)

Sign Off

Name	Position / Title	Signoff Requirement
Ankit Mahajan	Product	
Ali Imran	Delivery Head	

Table of Contents

1 Document Purpose

This document describes how to integrate Host's systems with Network International's N-Genius POS payment platform. The N-Genius payments platform has been designed to be future proof and cater for a wide range of payments without the need to change the interface between the host system and the payment platform. The interface has been designed to be simple to implement and to be card data free so that host applications and networks are free from the burdens of PCI compliance. Payment Card data is never present in the clear in the Host system. The N-Genius platform returns masked data that can be stored by the EPOS host for easy recons.

2 Solution Overview

All N-Genius devices can be configured by Network International (NI) to provide the entire range of services on offer including:

- Sale
- Refund
- Void
- Dynamic Currency Conversion
- Support for Visa, MasterCard, Amex, UPI, Diners, JCB, Mercury and Rupay
- Alipay
- UPI QR code
- VISA QR code
- MC QR code
- EPP Electronic Payment Plan (Installments)
- ADCB TouchPoints
- NOL card acceptance
- Pre-Authorization
- Pre-Authorization Completion
- Tips (Tier 1 and 2)
- Flexible End of Day
- WeChatPay
- XLS Loyalty
- TerraPay

All N-Genius devices share a common interface and there is no need to make changes in Host systems to use different N-Genius devices.

N-Genius devices come with a variety of connectivity options including

- USB
- Serial (Supported on a different Pinpad device type. Please contact your relationship manager for more details)

There are two possible configurations for integrating N-Genius terminals with Host Systems (EPOS tills, Vending machines, etc.). Both configurations use the same messaging between the Host system and the N-Genius device. Once configuration requires a service (the ngpas/androidpas service) to be installed on the Host system. The service runs in the background. The other configuration is only available for IP capable N-Genius devices, but communication can be direct from the Host to N-Genius or using ngpas/androidpas in IP mode.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.

2.1 Using the NGPAS Service

This is the only option that supports connectivity using Serial or USB N-Genius Devices, but can be configured to use Wi-Fi or Ethernet Devices too. For IP communication there is a pairing process that needs to take place. This process is not needed for USB and serial devices.

Network International provide a payments application service (NGPAS) that runs as a service on the host platform. NGPAS provides a set of commands to the host master application, and facilitates an end to end encrypted channel between the N-Genius device and NI's host systems.

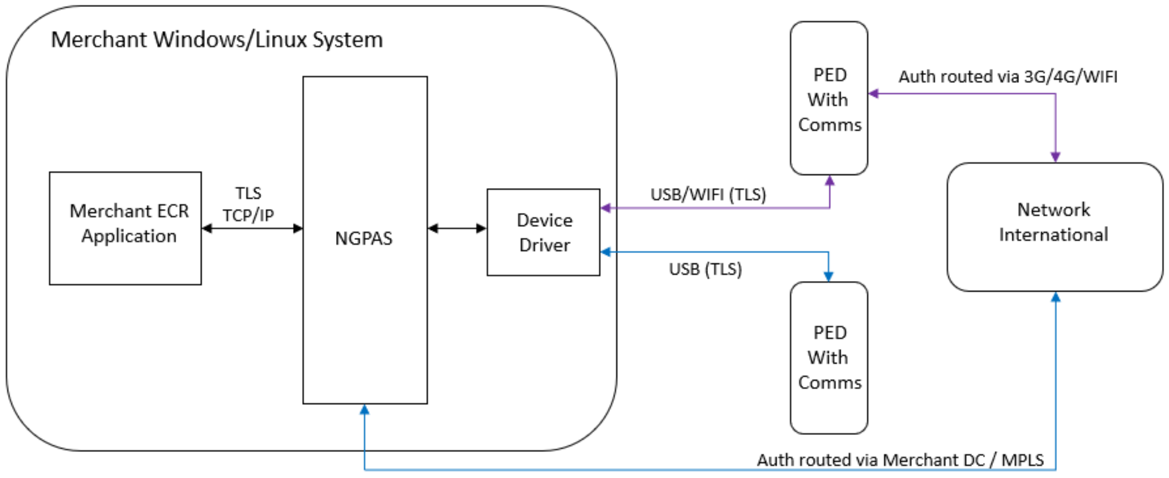
The NGPAS service runs on Windows or Linux hosts, but requires Java 8 to be installed (other versions of Java can be supported on request, but please confirm with NI if you require any Java version other than 8 to be supported). The service can be set to auto run on boot – the instructions to do this are included in the appendix. For development purposes it is useful to run the service in a console window.

NGPAS accepts commands over a TCP/IP socket connection. NI provides a sample EPOS simulator app that sends messages to NGPAS. This sample application is written in Java and the source code is provided. The application is describe in section Error: Reference source not found Error: Reference source not found

Communication to NI's hosts systems is HTTPS which can be over the Internet or a private network. This connectivity can be provided by the host EPOS integrated with N-Genius devices which have their own network connectivity options (Wi-Fi, 3G).

By default, NGPAS logs events to the file logs/servicelog-YYMMDD.txt, where YYMMDD refers to the date. A new file is generated when the current log file exceeds 5mb in size. This log file can be used for troubleshooting.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.

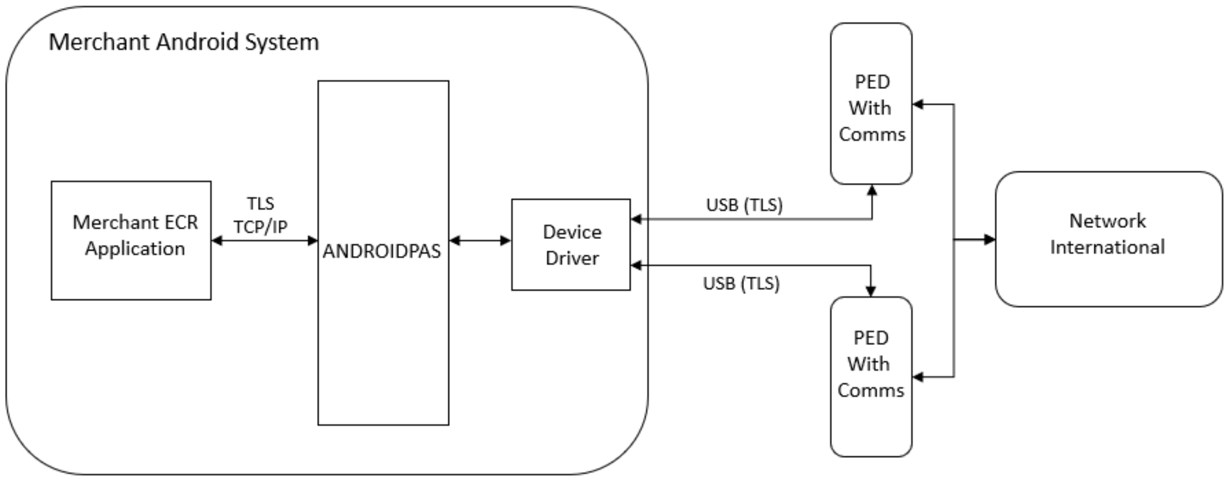


2.2 Using the ANDROIDPAS Service

There is a version of NGPAS which has been written to run as an apk on android devices. This version functions and is used in the same way as described by this document with a small number of differences:

- Android pas is launched like normal app on an android device and no longer required batch file
- The app will run in the background as a service allowing the android device to continue to be used.
- There is a config file which will need to be placed on the Sdcard.
- For USB mode the config file should specify "serial" for the peds Ip address and androidpas will scan for known devices when it starts and attempt to connect automatically.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.



3 N-Genius EPOS software architecture USB connection Interface Specification

NI provides sample code to integrate to the N-Genius platform in the following languages:

- Java (JDK 7 or higher)
- C#
- Delphi

The interface has been designed to be very simple to implement using json strings and simple text commands. Several methods are described briefly below. The following methods are detailed in **6.0 Functional Overview**.

connect

This establishes connection between ngpas/androidpas or the N-Genius device. If using IP communication, the file name of the N-Genius device certificate must be given as a parameter (this is not necessary for USB communication). This command is detailed in **6.1 connect**.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.

getTransactionAndTypes

This function returns the transactions that are available and their type. This function is used to make the introduction of alternative payments seamless as described in section 6 Functional Overview. This command is detailed in **6.4 getTransactionAndTypes**.

startTransaction

This function is used to start transactions. If the transaction being requested requires more parameters, the response requests the required parameters in sequence. This command is detailed in **6.5.1 startTransaction**.

cancelTransaction

This function is used to cancel the current transaction. This is not available after the transaction has been communicated to the acquirer. This command is detailed in **6.5.2 cancelTransaction**.

updateTransaction

If additional input parameters are requested for the transaction, the updateTransaction function can be used to update values in the current transaction by responding with the additional parameters. This function is only available after the transaction has started. This command is detailed in **6.5.3.**

getStatus

This provides the status of the current transaction (if available) and, if the transaction has been communicated to the host, the merchant receipt. The supplied Epos Sim example runs a getStatus() command at least every 3 seconds. Due to this, we recommend 3-seconds as an appropriate time to send getStatus() commands. This command is detailed in **6.5.4 getStatus**.

getResult

This provides the result of the current transaction (if available) and all the details required to print a customer and merchant receipt. Using getResult (sourceid) will return the result of a previous transaction that matches the provided source ID. This will only return transactions that have occurred before an End-Of-Day. This command is detailed in **6.5.5 getResult**.

4 N-Genius Devices

NI provide a range of devices depending on the POS environment. These are shown below. All provide magnetic stripe, contact and contactless chip card support. Some devices have printers which can be used to print receipts. For devices where a printer is not available the host must print the receipt

	OM-A880 **Host Connectivity: USB, Wifi*** **Connectivity to NI Host - 3G/4G, Wi-Fi &** **Ethernet (with optional base & requires development)** **Receipt printing: Host or device** **Battery operated** **Slave Mode or PayBill Mode available** **This device has a rechargeable battery. Charging is through Micro USB/USB-C. A mains powered Micro USB/USB C charger is provided with the device.** ***Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.***

5 Host specification

The host system where NGPAS is running will be required to have Java SE Development Kit (JDK) 8 installed.

JRE 8 may be downloaded from here: https://www.oracle.com/uk/java/technologies/javase/javase-jdk8-downloads.html

If the N-Genius device is connected by USB, then the host system will be required to have USB ports.

6 Functional Overview

The N-Genius platform supports traditional and alternative payment methods e.g. Card (Traditional card and mobile payments), ALIPAY, Loyalty, Voucher etc. The host system should send a getTransactionAndTypes request to discover all the payment types available. The default payment type is Card, however, alternative payment options should be offered prior to starting a transaction if they are available. The transaction types that are available are returned by the getTransactionAndTypes call which returns all the transactions available. Transactions can be one of three types, Sales, Admin, Refund. It is not necessary for the host system to support all transaction types. Systems can just support Sales. Sales are where money is charged to the customers' card for the goods sold. Admin transactions do not have financial value, for instance and end of day transaction. Refund transactions are where money is returned to customer electronically e.g. for a card refund.

Note: Currently Sale, Void, Refund, DCC, QR and Admin (EOD) services are supported.

Commands sent to N-Genius over IP should be followed by a LF or CRLF. N-Genius will respond with a header "transaction", followed by a json string containing data associated with the transaction.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.

A description of the fields that are returned in the json data is give in section 9.1. New fields may be introduced. If the host does not recognize a field name it should be ignored. This will be tested as part of certification.

6.1 connect()

connect() is used to determine whether the device is connected.

connect() should always respond in 1s. If no response to connect() is received the device is either switched off or not connected.

```
-> connect()

<- connected
```

6.2 getServerInfo()

getServerInfo() is used to determine the configuration of the device.

```
-> getServerInfo()

<- serverInfo {
    "androidVersion":"6.0.1 (23)",
    "commsService":"4.02 (Built: 2018-10-12 16:38:32)",
    "configProvider":"4.02 (Built: 2018-10-12 12:57:26)",
    "dccMinMax":"1.00 / 9,999,999.99",
    "dccProvider":"Planet Payment (Enabled)",
    "emvService":"4.02 (Built: 2018-10-12 16:37:36)",
    "firmwareVersion":"18082302","merchantId":"316180020",
    "merchantName":"CATER CATERING SERVICE",
    "modules":[
        "sale",
        "refund",
        "void"
   ],
    "nextGen":"4.02 (Built: 2018-10-12 16:39:12)",
    "printService":"4.02 (Built: 2018-10-12 12:41:32)",
    "serialNo":"D2039",
    "simInfo":"3 (Inactive)",
    "tillId":"88880005",
    "wifiInfo":"wl (Active) 192.168.2.64 02:1E:10:1F:00:00"
}
```

getServerInfo() can be used to determine the current state of the device’s software and connectivity.

6.3 updatePed()

UpdatePed is used to update the PED with the latest changes to the configuration file, as well as check the APKs for newer versions. If no APK updates are required, the message content will state that the APKs are on the latest version.

Updating a PED has the following restrictions:

1. **Transaction in progress** – The PED will not update its configuration or check for software updates.
2. **End-of-Day required** – The configuration for the device will update but any available software updates will not proceed.
3. **Battery too low** – The configuration for the device will update but any available software updates will not proceed.

```
->updatePed()

<-updatePed {"androidVersion":"7.1.2 (25)","commsService":"1.61.067 (Built:
2022-11-15 10:05:36)","configProvider":"1.61.067 (Built: 2022-11-15
10:02:18)","emvService":"1.61.067 (Built: 2022-11-15
10:03:54)","firmwareVersion":"7.0.26","message":"Fri Dec 30 15:12:55
GMT+04:00 2022: PED Configuration has been updated. No update required for
the APKs as they are on the latest version.","nextGen":"1.61.067 (Built:
2022-11-15 10:01:00)","printService":"1.61.067 (Built: 2022-11-15
10:03:10)","success":true}
```

6.4 Transaction Commands

The transaction flow is made up of numerous different commands send to and from the device. Each command is detailed below, followed by some examples of different transaction flows.

6.4.1 startTransaction

startTransaction can be called in two different ways to initiate a transaction.

- startTransaction(<TransactionType>)
- startTransaction <JSON>

Both methods will cause the PED to attempt to start the selected transaction type.

On a successful transaction start, the PED will return a *transaction* message back.

If the PED is unable to start the transaction, an *error* will be returned. Example messages are provided in the following section **(6.6 Transaction Flows)**.

```
-> startTransaction
{"success":false,"amount":"2000","sourceid":"345","type":"eposSale"}
```

Example messages are provided in the following section (6.6 Transaction Flows).

6.4.2 cancelTransaction()

cancelTransaction will cause the current transaction to be cancelled. If this command is called after the transaction has been communicated to the acquirer, an error will be returned "Cannot cancel transaction”.

It is recommended that at least 3 seconds is left after sending a startTransaction before sending a cancelTransaction request.

If a cancel request from the EPOS to the PED fails (because it is not possible to cancel the transaction), an error response will be returned, otherwise the cancel request has been successful.

Otherwise, the cancel request has been successful.

The 'inProgress' flag will still be true for a short period whilst the PED is processing the cancellation.

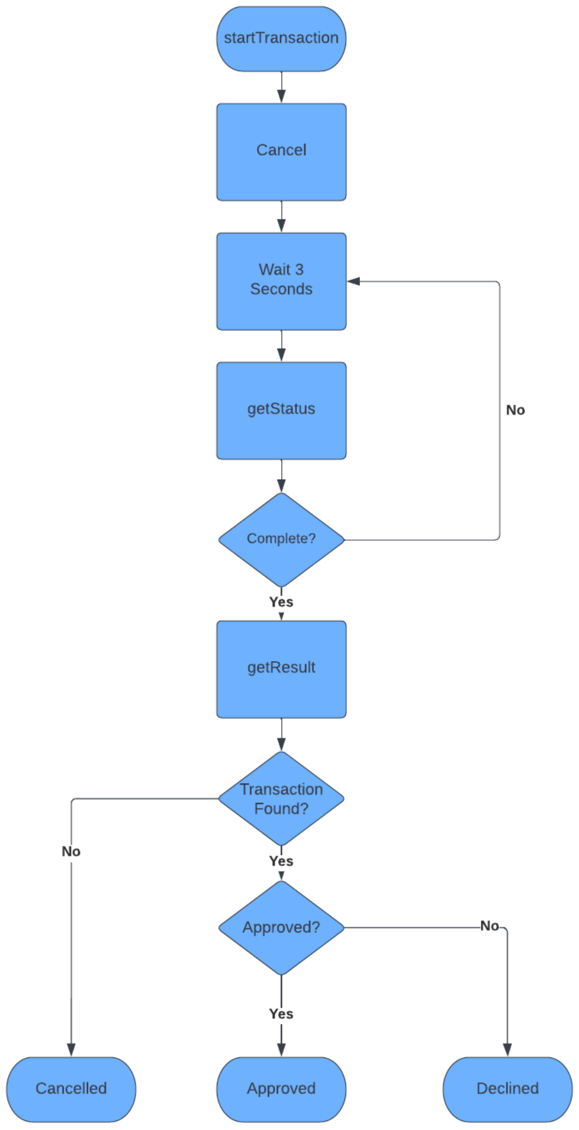
During this period the message displayed to the customer will contained in the 'displayText' field.

The PED will take less than 30 seconds to process the cancellation (typically the period will be much shorter)

get Status can be used to determine when the cancellation is complete, by checking for the inProgress flag to be false.

The following cancel flow could be implemented by the EPOS application to ensure the result of the transaction is accurately known.

```
->  cancelTransaction()
```



The above flow should be used by the EPOS application to ensure the result of the transaction is accurately known.

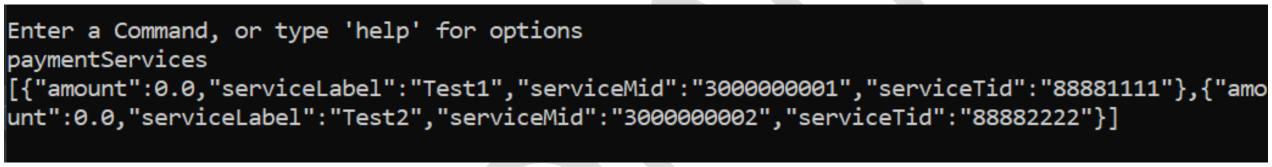
6.5 EposMultiMid

6.5.1 *EposMultiMid - Sale*

This command should be specifically used in scenarios where Merchant/Agency supporting multiple service entities, can collect bills for various services (e.g., Airline, Government etc.)
In this case, a Merchant ECR can trigger transaction for multiple service entities (linked to corresponding MID/TID) via same physical POS device. And facilitate fund settlement to respective service entity merchant. For this TMS to be configured as ***Multi-Auth Mode Processing: (Multi Auth Enabled)***

~ To check for available configured payment Services on EPOS. There is a new command introduced that is to be sent from epos. i.e.,

- ➔ getPaymentService()
  This will return configured payment service on PED to EPOS which is as follows.



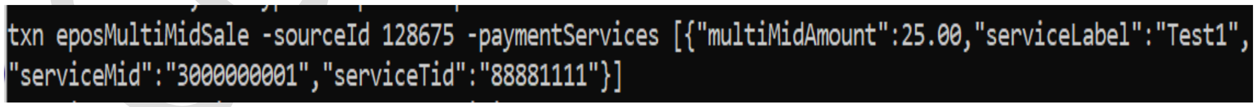
```
Enter a Command, or type 'help' for options
paymentServices
[{"amount":0.0,"serviceLabel":"Test1","serviceMid":"3000000001","serviceTid":"88881111"},{"amo
unt":0.0,"serviceLabel":"Test2","serviceMid":"3000000002","serviceTid":"88882222"}]
```

~ To trigger a MultiMid Sale txn. We can trigger the command in this way.

- ➔ txn eposMultiMidSale

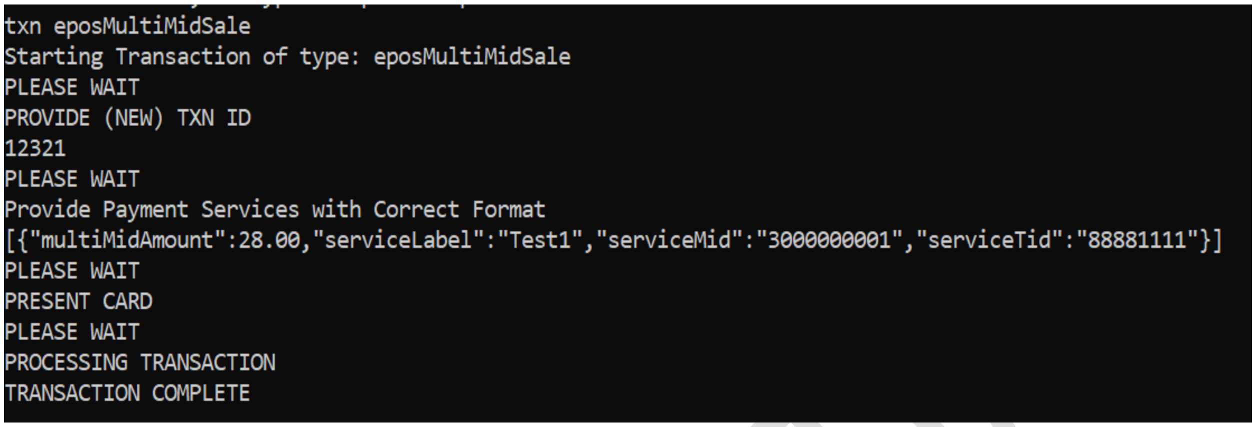
~ To pass the serviceLabel and to trigger a transaction based on the service MID/TID. It can be done in the following ways.

1. txn eposMultiMidSale -paymentServices [<<array json of payment services>>]
   This will take "paymentServices" as a parameter which should be received from EPOS as a JSON. For ex:



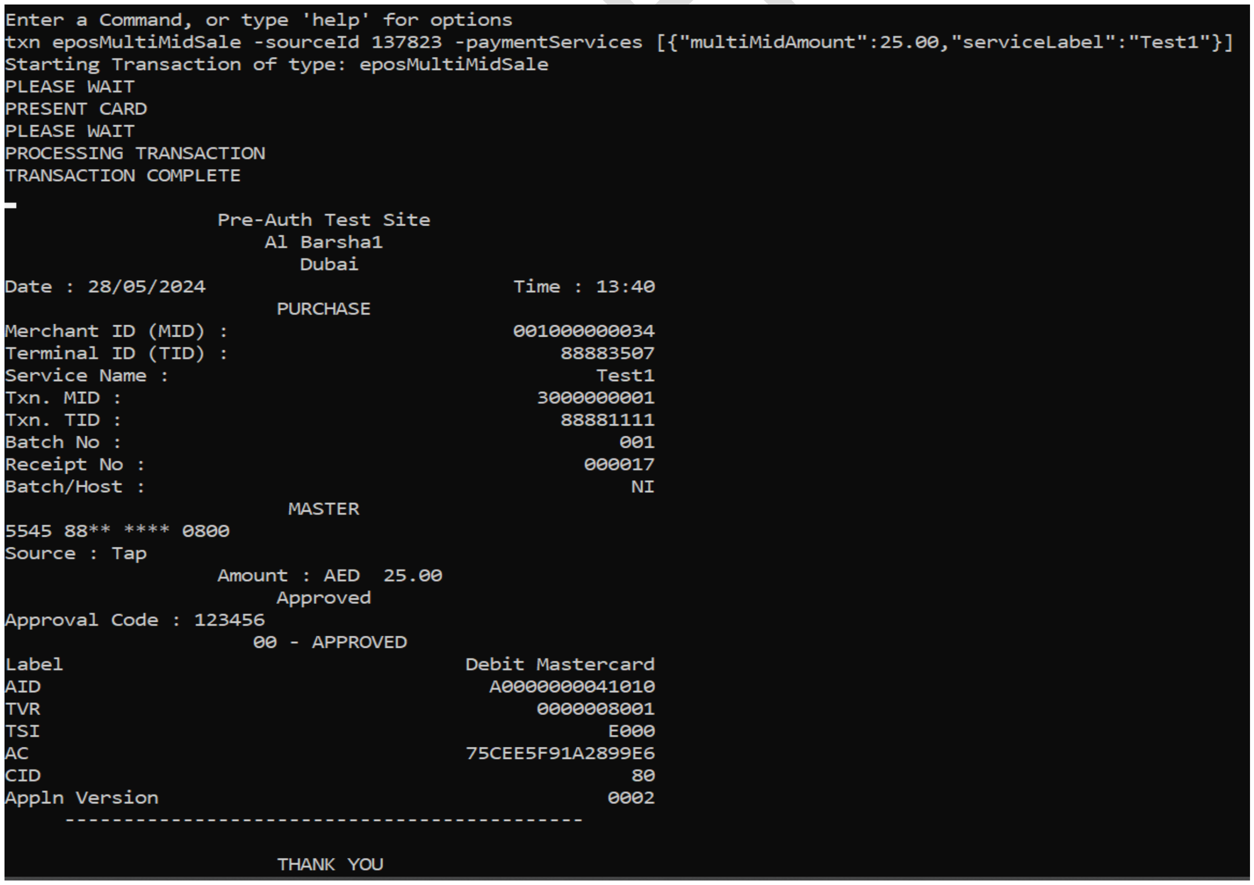
```
txn eposMultiMidSale -sourceId 128675 -paymentServices [{"multiMidAmount":25.00,"serviceLabel":"Test1",
"serviceMid":"3000000001","serviceTid":"88881111"}]
```

2. txn eposMultiMidSale



```
txn eposMultiMidSale
Starting Transaction of type: eposMultiMidSale
PLEASE WAIT
PROVIDE (NEW) TXN ID
12321
PLEASE WAIT
Provide Payment Services with Correct Format
[{"multiMidAmount":28.00,"serviceLabel":"Test1","serviceMid":"3000000001","serviceTid":"88881111"}]
PLEASE WAIT
PRESENT CARD
PLEASE WAIT
PROCESSING TRANSACTION
TRANSACTION COMPLETE
```

3. If *serviceMid* and *serviceTid* are not provided from the ECR Machine in the JSON then the PED will take sent serviceLabel(Ex: “FlyDubai”) associated *serviceMid* and *serviceTid* configured on PED and initiate the transaction. If serviceLabel itself is not found, then it will throw appropriate error.



```
Enter a Command, or type 'help' for options
txn eposMultiMidSale -sourceId 137823 -paymentServices [{"multiMidAmount":25.00,"serviceLabel":"Test1"}]
Starting Transaction of type: eposMultiMidSale
PLEASE WAIT
PRESENT CARD
PLEASE WAIT
PROCESSING TRANSACTION
TRANSACTION COMPLETE

                  Pre-Auth Test Site
                      Al Barsha1
                         Dubai
Date : 28/05/2024                           Time : 13:40
                       PURCHASE
Merchant ID (MID) :                         001000000034
Terminal ID (TID) :                             88883507
Service Name :                                     Test1
Txn. MID :                                    3000000001
Txn. TID :                                      88881111
Batch No :                                           001
Receipt No :                                      000017
Batch/Host :                                          NI
                        MASTER
5545 88** **** 0800
Source : Tap
                  Amount : AED  25.00
                       Approved
Approval Code : 123456
                     00 - APPROVED
Label                                   Debit Mastercard
AID                                       A0000000041010
TVR                                           0000008001
TSI                                                 E000
AC                                      75CEE5F91A2899E6
CID                                                   80
Appln Version                                       0002
     ---------------------------------------------

                       THANK YOU
```

Z report:

MultiMid Sale Z Report.docx

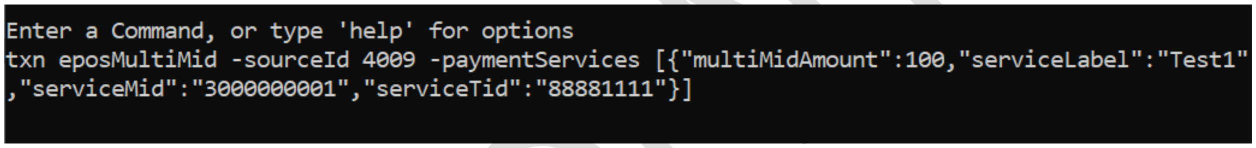
6.5.2 *EposMultiMid – Refund*

- MultiMid Refund has separate txn type and it is triggered as

➔ txn eposMultiMidRefund

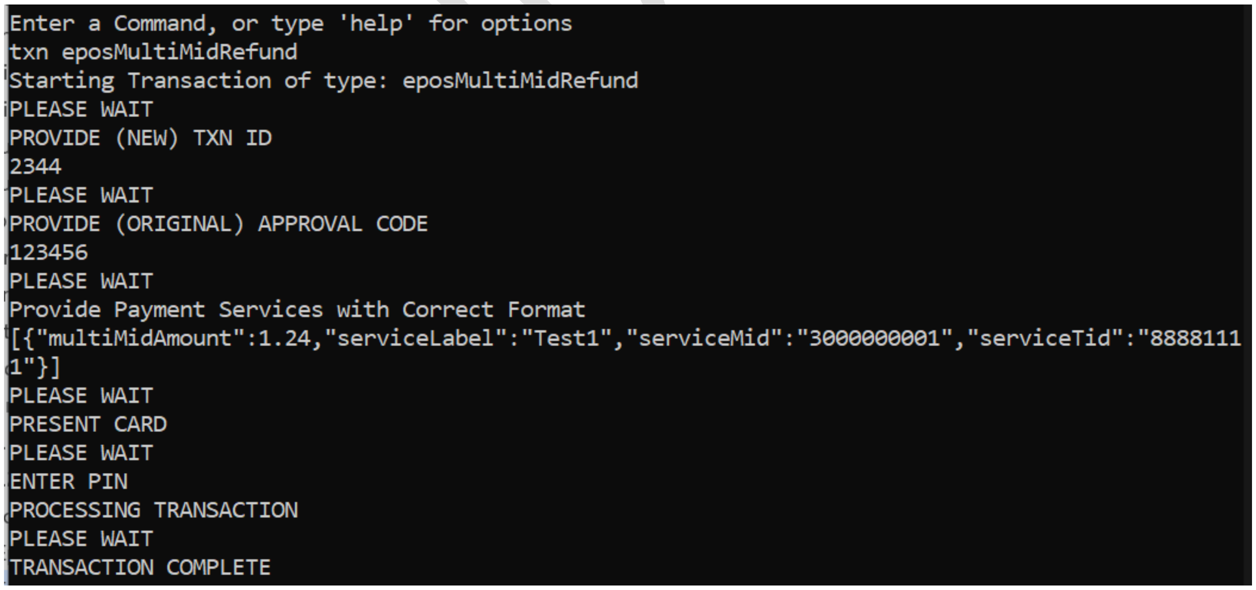
- To pass the serviceLabel and to trigger a transaction based on the service MID/TID. It can be done in the following ways.

1. txn eposMultiMidRefund -paymentServices [<<array json of payment services>>]



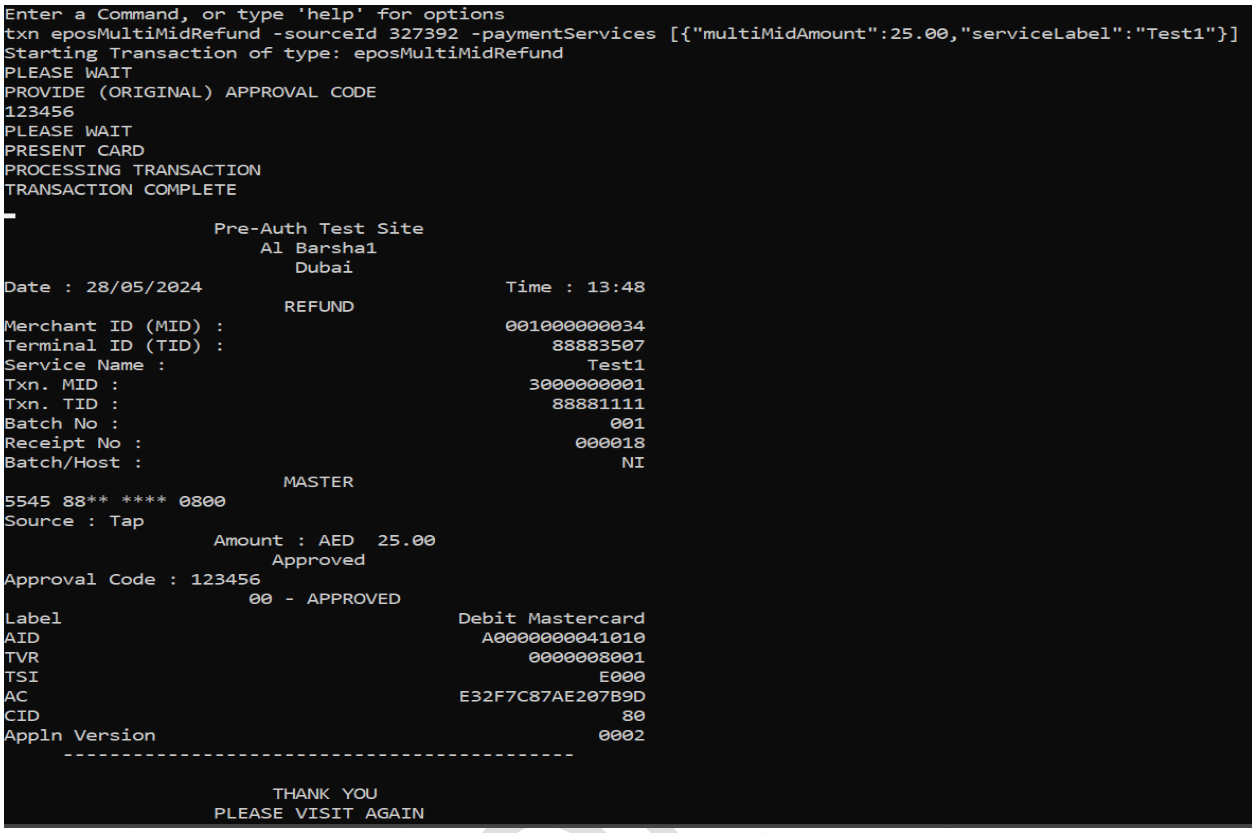
```
Enter a Command, or type 'help' for options
txn eposMultiMid -sourceId 4009 -paymentServices [{"multiMidAmount":100,"serviceLabel":"Test1"
,"serviceMid":"3000000001","serviceTid":"88881111"}]
```

2. txn eposMultiMidRefund



```
Enter a Command, or type 'help' for options
txn eposMultiMidRefund
Starting Transaction of type: eposMultiMidRefund
PLEASE WAIT
PROVIDE (NEW) TXN ID
2344
PLEASE WAIT
PROVIDE (ORIGINAL) APPROVAL CODE
123456
PLEASE WAIT
Provide Payment Services with Correct Format
[{"multiMidAmount":1.24,"serviceLabel":"Test1","serviceMid":"3000000001","serviceTid":"8888111
1"}]
PLEASE WAIT
PRESENT CARD
PLEASE WAIT
ENTER PIN
PROCESSING TRANSACTION
PLEASE WAIT
TRANSACTION COMPLETE
```

3. If *serviceMid* and *serviceTid* are not provided from the ECR Machine in the JSON then the PED will take sent serviceLabel(Ex: “FlyDubai”) associated *serviceMid* and *serviceTid* configured on PED and initiate the transaction. If serviceLabel itself is not found, then it will throw appropriate error.



```
Enter a Command, or type 'help' for options
txn eposMultiMidRefund -sourceId 327392 -paymentServices [{"multiMidAmount":25.00,"serviceLabel":"Test1"}]
Starting Transaction of type: eposMultiMidRefund
PLEASE WAIT
PROVIDE (ORIGINAL) APPROVAL CODE
123456
PLEASE WAIT
PRESENT CARD
PROCESSING TRANSACTION
TRANSACTION COMPLETE
_
                  Pre-Auth Test Site
                      Al Barsha1
                         Dubai
Date : 28/05/2024                         Time : 13:48
                        REFUND
Merchant ID (MID) :                       001000000034
Terminal ID (TID) :                           88883507
Service Name :                                   Test1
Txn. MID :                                  3000000001
Txn. TID :                                    88881111
Batch No :                                         001
Receipt No :                                    000018
Batch/Host :                                        NI
                        MASTER
5545 88** **** 0800
Source : Tap
                  Amount : AED  25.00
                       Approved
Approval Code : 123456
                     00 - APPROVED
Label                                 Debit Mastercard
AID                                     A0000000041010
TVR                                         0000008001
TSI                                               E000
AC                                    E32F7C87AE207B9D
CID                                                 80
Appln Version                                     0002
     --------------------------------------------

                       THANK YOU
                  PLEASE VISIT AGAIN
```

Note:

The service JSON consists of four keys that *should* be in same format as provided i.e., multiMidAmount, serviceLabel, serviceMid, serviceTid that is to be sent from epos. These act as identifier to match the service configuration downloaded on the PED and to send in request as well.
Four keys are case Sensitive, and this should *be sent in exact same way*.

6.6 getTransactionAndTypes()

getTransactionAndTypes() provides the host with all the transaction types that can be started by the Epos integration

```
-> getTransactionAndTypes()

<- transactionTypes [{"displayText":"Card
Payment","transaction":"eposSale","type":"sale"},{"displayText":"Refund","tr
ansaction":"eposRefund","type":"admin"},{"displayText":"Refund
Void","transaction":"eposRefundVoid","type":"admin"},{"displayText":"Void","
transaction":"eposVoid","type":"admin"},{"displayText":"Pre-Auth
Sale","transaction":"eposPreAuthSale","type":"sale"},displayText":"X
Report","transaction":"X","type":"report"},{"displayText":"Last X
Report","transaction":"LASTX","type":"report"},{"displayText":"Z
Report","transaction":"Z","type":"report"},{"displayText":"Last Z
Report","transaction":"LASTZ","type":"report"}]
```

6.6.1 updateTransaction<JSON>

updateTransaction is used to respond to requests from the PED for additional values, by including the “parameter”, “parameterType” and “parameterValue” fields.

```
 ->  updateTransaction
{"success":false,"amount":"2000","cashback":"0","tipAmount":"0","sourceid":"
347","currency":"0784","inProgress":true,"displayText":"Error reading card,
remove card and click OK to
retry.","parameter":"confirm","parameterType":"alphanumeric","parameterValue
":"ok"}
```

6.6.2 getStatus()

getStatus can be used to continually poll the PED for the status of the transaction. The supplied Epos Sim example runs a getStatus() command at least every 3 seconds. Due to this, we recommend 3-seconds as an appropriate time to send getStatus() commands.

The following flags should be considered in getStatus response to determine if the transaction is complete & getResult should be called:
Complete - This indicates if the transaction processing has finished and therefore the result of the transaction can be determined so a getResult call will

InProgress - Indicates if a transaction flow is still in progress on the PED or if it is ready to perform a new transaction.

```
 ->  getStatus()
```

6.6.3 getResult([sourceid])

getResult can be used with or without an optional parameter: [sourceid]

If the parameter is included, the requested transaction’s result will be returned from the PED’s internal database.

If the parameter is not included, the device will send the result of the current transaction. If the current transaction has not been completed, an *error* message **(section 6.5.7)** may be returned.

getResult() should be trigger during the transaction when Complete flag changes its values from false to true.

The success flag will indicate whether the transaction was approved or declined along with further information about the transaction being returned in the response to getResult().

```
 -> getResult(345)
```

6.6.4 transaction <JSON>

The PED will return information about a transaction in a *transaction* message. This may be used to reply to a startTransaction, cancelTransaction, getStatus or getResult command from the host system.

The JSON block will contain information relating to the transaction, including any results, and the merchant receipt if it is required for signature confirmation.

Examples of what may be contained in the *transaction* message can be seen in the following section (6.6. Transaction Flow) and a full list of data elements can be found in Appendix 9.1.

getStatus response is given below:

```
-> transaction
{"amount":"2000","cardInserted":false,"cashback":"0","complete":false,"curre
ncy":"0784","currencyDecimalPlaces":"2","dccAccepted":false,"dccOffered":fal
se,"declined":false,"displayText":"PLEASE
WAIT","eposCheckCard":false,"inProgress":true,"offline":false,"signatureRequ
ired":false,"sourceid":"345","success":false,"tokenTransaction":false}
```

6.6.5 **report([ReportType]**, [optional])

When the PED receives the report() command, it will respond with the “report <JSON>”

printFromPed is an optional boolean parameter which controls the printing device. If *“true”* is passed, then the report will be printed via the PED’s built in printer or displayed on the screen in the case of an AP-10. If *“false”* is passed then the printing will be handled by the ECR. The default value for this parameter is *“false”* this means that existing functionality will not need to change.

The JSON block will contain information about the report, including both a receipt and the report totals, detailed in Appendices 9.1.1. This will not be returned in the case of printFromPed being *“true”.*

```
 -> report(Z, false)
```

To print the Z report from PED itself:

```
-> report(Z, true)
```

6.6.6 error <JSON>

When the PED cannot interpret, or complete the requested command, it will respond with an error message, and a block of JSON detailing the issue. Similar to other messages from the PED, any data and potential error details will be sent in JSON format.

For a complete list of all errors, please see section 6.8 Error Responses.

6.6.7 userLogon <JSON> <JSON>

In some scenarios (such as Taxi-EPOS) before a device can perform transactions the user must logon. The fields "user Id" and "vehicleReg" are compulsory and should be included in the JSON components.

A successful logon will result in a user LogonResponse <JSON> response being received with the "SUCCESS" value set to true. If logon is unsuccessful an error <JSON> will be returned.

6.6.8 userLogoff([optional])

This command will log the user off their PED and complete an End of Day. As with the report() command the optional parameter "printFromPed" may be included. In the event of a successful logoff a report <JSON> response will be received. Please see 6.5.7 for an explanation of this response type and the behavior of "printFromPed".
In the event of an error occurring, for example no user is logged on or the End of Day has failed an error<JSON> will be returned.

6.6.9 deviceShutdown

This command will cause the PED to shut down. A "powerResponse" will be returned and then the PED will be shut down.

6.6.10 deviceReboot

This command will cause the PED to shutdown and then restart. A "deviceShutdownResponse" will be returned and then the PED will be rebooted.

6.7 Mandatory Section for EPOS Integration Implementation

Transaction Sequence:

1. Connect() command should be triggered and PED will be responding with Connected if the connectivity established.
2. GetStatus() should be called to check if the PED is in ready for txn state, PED will be responding with NO TXN/System Idle if it's ready to transact.
3. StartTransaction() should be triggered with required parameters.

GetStatus()
GetStatus should be called every 3 sec to check the status of the transaction.

GetResult(SourceId)

GetResult should be called with SourceId and GetResult(SourceId) should be called once the "Complete" flag changes to true from false.

In case of any cable disconnection during the transaction, GetResult(SourceId) should be called to check whether the transaction was completed on the PED.
If the transaction is already completed, PED will respond with Transaction details else PED will be responding as "**Transaction Not found**"
Attached Cable disconnect scenario logs for reference:

GetResult(SourceId).txt

Update Transaction:
Update Transaction should be triggered in case of below scenarios where PED waits for an action from ECR.

In case of any prompt for parameter confirmation, OK Command should be passed in Update Transaction

Example :

[EPOS   NGPAS <- PED] transaction
{"amount":"100","cardInserted":false,"cashback":"0","complete":false,"currency":"0784","currencyDecimalPlaces":"2","dccAccepted":false,"dccOffered":false,"declined":false,**"displayText":"No card detected. Cancelling transaction.","**eposCheckCard":false,"inProgress":true,"offline":false**,"parameter":"confirm","parameterType":"alphanumeric","**signatureRequired":false,"sourceid":"1001","success":false,"tokenTransaction":false}

EPOS -> NGPAS   PED] **updateTransaction**
{"success":false,"amount":"100","cashback":"0","sourceid":"1001","currency":"0784","inProgress":true,"displayText":"No card detected. Cancelling transaction.","parameter":"confirm","parameterType":"alphanumeric"**,"parameterValue":"ok"}**

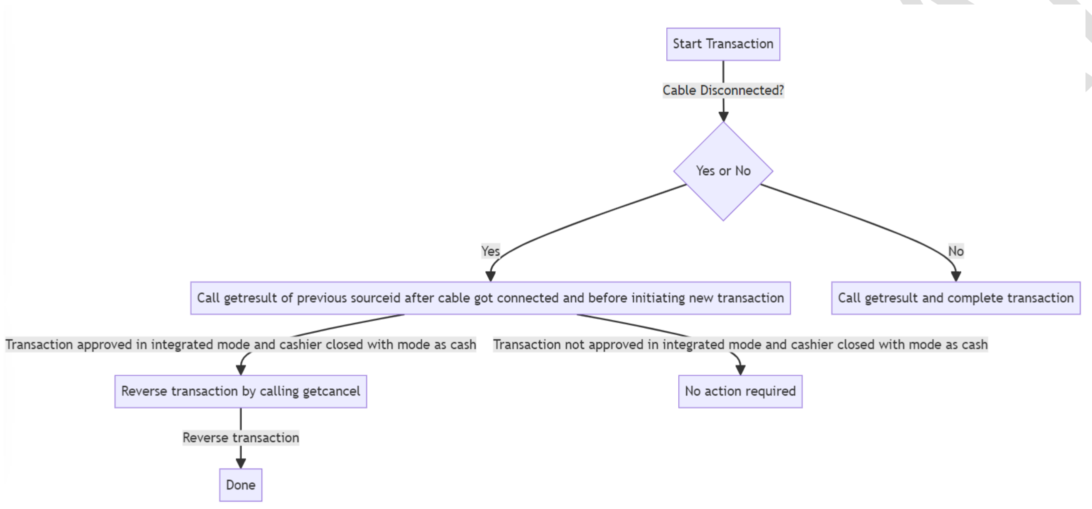
Possible Update Transaction scenarios are listed on the attached doc.

Epos Update Transaction Scenarios

Note: All the scenarios need to be handled to avoid the PED stuck scenarios

In case cable disconnection during the transaction below needs to be followed once the cable connected

- If the response for the transaction is unavailable in the ECR system for the last transaction.
- Prior to initiating the next transaction, a "Get Result" for the last transaction should be conducted.
- This will provide the status of the previous transaction, allowing the ECR system to send a cancel/void request for the previous transaction to prevent a double debit scenario.
- Refer the below flow diagram


Start Transaction
Cable Disconnected?
Yes or No
Yes
No
Call getresult of previous sourceid after cable got connected and before initiating new transaction
Call getresult and complete transaction
Transaction approved in integrated mode and cashier closed with mode as cash
Transaction not approved in integrated mode and cashier closed with mode as cash
Reverse transaction by calling getcancel
No action required
Reverse transaction
Done

Receipt Printing:

If Receipts are printed from ECR, All the fields in Receipt data should be printed as is.

Z report:

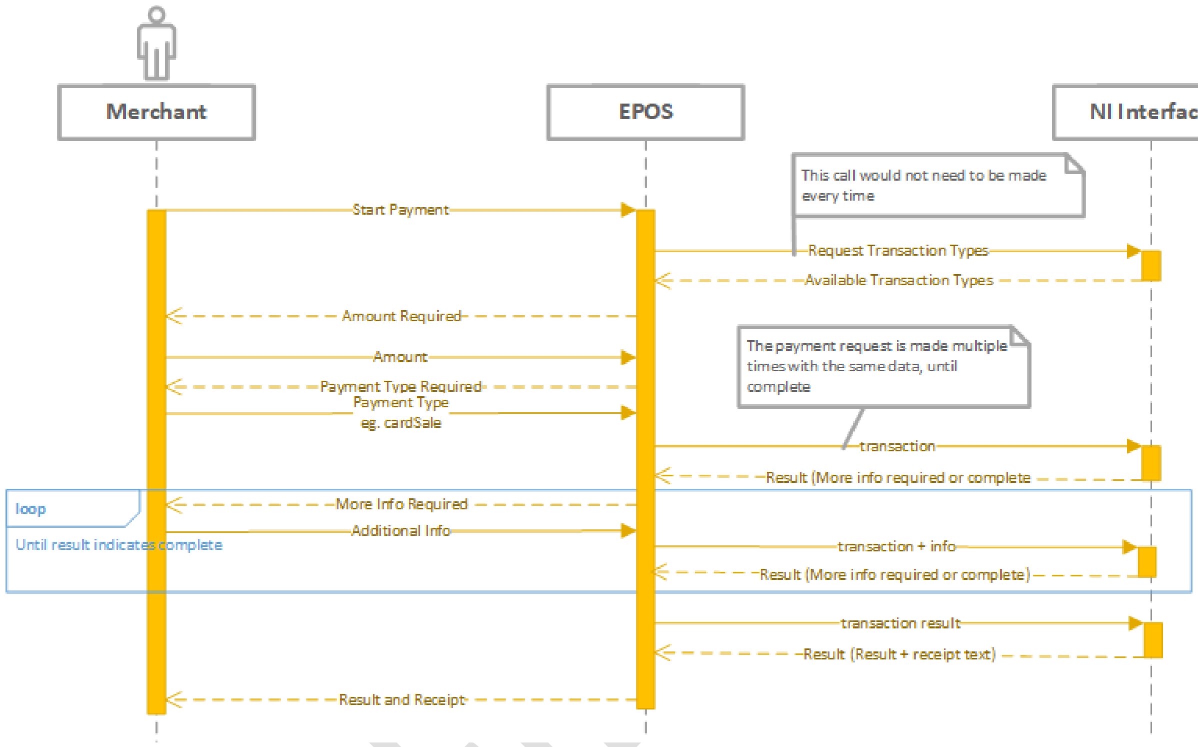
Its mandatory to trigger the Z report from the device on a daily basis to make sure that there is no performance issues due to storage issues.

APK Update/ Config Updates:

updatePed can be triggered to update the PED configurations/APK Update.

6.8 Example Transactions

6.8.1 Transaction Flow



Merchant
EPOS
NI Interface
Start Payment
This call would not need to be made every time
Request Transaction Types
Available Transaction Types
Amount Required
Amount
Payment Type Required
Payment Type eg. cardSale
The payment request is made multiple times with the same data, until complete
transaction
Result (More info required or complete
loop
Until result indicates complete
More Info Required
Additional Info
transaction + info
Result (More info required or complete)
transaction result
Result (Result + receipt text)
Result and Receipt

Figure 2- Sequence diagram showing transaction flow

Sample Logs for few transaction types are attached in section Sample Transactions

6.8.2 Successful Transaction

Successful Transaction.txt

6.8.3 Cancel Transaction

Cancel Transaction.txt

6.8.4 Failed Cancellation

Failed Transaction.txt

6.9 Transaction Parameters

6.9.1 Parameter Requests

For some transaction flows N-Genius may require additional parameters to be received to complete the transaction. Additional parameters may be required once the card has been recognized and after card insertion. For example, if a card supports cashback a cashback amount may be requested once the card has been recognized.

Parameters requested will either be "Numeric" – e.g., Phone number, Alphanumeric or a barcode scan. If the host is not able to capture the parameter e.g. no alphanumeric input or no barcode reader, it should return a default value or an empty string.

If the PED responds to getStatus() with a value in both "parameter" and "parameterType", the flow will halt until an updateTransaction <JSON> command has been sent, with the same "parameter" and "parameterType" fields, and a valid value in the "parameterValue" field.

Current parameters that may be requested are:

- sourceid
  - This is the ID of the transaction from the host EPOS. It should be unique. It is used to retrieve the transaction if it is needed in case of a VOID, or to return a result of a previously completed transaction.
    Recommendation format, 15 digit unique nos. ex. YYYYMMDDHHMMSSn, where YYYYMMDD - date, HHMMSS – time, n– shift nos.
- amount
  - This is the amount for the transaction. It is in the currency that the PED has been configured. (For now, AED)
  - The amount is entered as a string, in the lowest decimal currency. i.e. "1000" = "10.00 AED"
- voidOriginalId
  - This is the original source ID for voiding a transaction. This should be the source id of the transaction the EPOS is attempting to void
- confirmtxn
  - This is a Boolean value, respond with "true" or "false" to confirm that the transactions details returned are the details of the txn you wish voided
- refundapprovalcode
  - This is the approval/auth code of the transaction you wish to refund
- voicereferralcode
  - This is the approval/auth code acquired when performing voice referral transactions
- alipaybarcode

- This is the barcode, scanned from the customer's phone while performing an Alipay transaction. If you wish to perform the transaction and have the customer scan the device's screen, return "SCAN_PED" to this parameter

- uniqueOrderId
  - Used for Alipay refund and WeChat Refund

- barcode: SCAN_PED
  - To generate QR codes for Merchant presented QR transactions (UPI, Wechat, Visa,MC, Alipay)
- phone
  - To pass the terrapay mobile number
- signature
  - This is a Boolean value, respond with "true" or "false" to confirm that the signature on the receipt matches the one on the back of the customer's card
- checkcard
  - If "checkCard" is true in the startTransaction request, the EPOS will pause after card entry to request this parameter. This should be provided as one of the following:
    - cashback
    - amount
    - continue
    - cancel
  - If "cashback" or "amount" is returned, the PED will then request that parameter, and then send the "checkcard" parameter again, to see if there are other changes to amounts. See 6.6.2
  - If "continue" is provided, the PED will continue with the transaction.
  - If "cancel" is provided, the transaction will be cancelled.
- magstripeconfirm
  - If a magstripe card is swiped, the PED will request this parameter. The response should be the last four digits of the swiped card.

- offlineMode (Not available as a standard feature)

  - If "offline" is true in the startTransaction request, the PED will authorize the transaction without going online if the transaction is within the offline authorization limits. See 10

- UpiQrcVoucherNo

  - 'PROVIDE UPI VOUCHER NO' – This asks for the UPI VOUCHER NUMBER. This is a long series of numeric characters. E.g. 20206127797333890671. This appears on the receipt as QRC VOUCHER NUMBER.

- visaQrRefundCode
  - 'PROVIDE VISA QR REFUND CODE' – This asks for the VISA QR Refund Code. This is a series of alphanumeric characters. E.g. 34354345ABE. This appears on the receipt as APPROVAL NUMBER.

- tokenTransaction
    - If “tokenTransaction” is present and set to true, the PED will treat the transaction as a token transaction for both input and generation. 2nd-part transactions (PRE-AUTH COMPLETE, REFUND, VOID) will all prompt for:
        - Enter Token
        - Enter Auth Code
        - Enter Source ID
        - Enter Amount
    - If “tokenTransaction” is present and set to true, the PED will request a token to be generated from TM for all transactions. That means that all transactions will display a token on the receipt. This token can be used on future 2nd-part transactions.
      Note: Currently not supported

- userId
    - Required parameter for user logon
    - Alphanumeric value up to 10 characters in length representing the ID of the individual user
- vehicleReg (Specific to Taxi)
    - Required parameter for user logon
    - Alphanumeric value
- paymentServices (For MultiMid)
    - Required parameter for multiMid
    - Alphanumeric value

6.9.2 Post-insertion check card

If the flag “eposCheckCard” is set in the message JSON, the PED will pause after it has retrieved the card information, setting a parameter of “checkcard”, and providing the known card information to the EPOS.

The EPOS should respond to this parameter with one of the following strings; “cashback”, “amount” “continue” or “cancel” – in the method detailed above in 6.6.1
This can be used to change the amount based on the card number, or add a cashback amount to supported cards.

The PED will then request the returned parameter (i.e. if “cashback” is returned, the PED will request a parameter value for “cashback”), and then perform another “checkcard” parameter request.

This will continue until the PED receives a response of “continue” or “cancel” to the “checkcard” parameter request.

Note that for Contactless transactions, the amount cannot be changed after card read, as the EMV processing has already been completed when the tap occurred. The EPOS should not update the amount after insertion if “cardReadMode” returns as “CTLS”

For Contact transactions, some additional EMV processing will occur if the total amount has changed, this is required so that the card can validate the cashback amount that has been added to the transaction. The transaction may become declined at this stage by the card.

6.9.3 Parameter Request Example

```
-> startTransaction
{"success":false,"amount":"1230","sourceid":"135","type":"eposSale"}
<- transaction
{"amount":"1230","cardInserted":false,"cashback":"0","complete":false,"curre
ncy":"0784","currencyDecimalPlaces":"2","dccAccepted":false,"dccOffered":fal
se,"declined":false,"displayText":"PLEASE
WAIT","eposCheckCard":false,"inProgress":true,"offline":false,"signatureRequ
ired":false,"sourceid":"135","success":false,"tokenTransaction":false}

-> getStatus()
<-transaction
{"amount":"1230","cardInserted":false,"cashback":"0","complete":false,"curre
ncy":"0784","currencyDecimalPlaces":"2","dccAccepted":false,"dccOffered":fal
se,"declined":false,"displayText":"PLEASE
WAIT","eposCheckCard":false,"inProgress":true,"offline":false,"signatureRequ
ired":false,"sourceid":"135","success":false,"tokenTransaction":false}

-> updateTransaction
{"success":false,"amount":"3000","cashback":"0","sourceid":"38","currency":"
0784","inProgress":true,"displayText":"Error reading card, remove card and
click OK to
retry.","parameter":"confirm","parameterType":"alphanumeric","parameterValue
":"ok"}
<- transaction
{"amount":"3000","cardInserted":false,"cashback":"0","complete":false,"curre
ncy":"0784","currencyDecimalPlaces":"2","dccAccepted":false,"dccOffered":fal
se,"declined":false,"displayText":"PLEASE
WAIT","eposCheckCard":false,"inProgress":true,"offline":false,"signatureRequ
ired":false,"sourceid":"38","success":false,"tokenTransaction":false}

->  cancelTransaction()
<-  transaction
{"amount":"3000","cardInserted":false,"cashback":"0","complete":false,"curre
ncy":"0784","currencyDecimalPlaces":"2","dccAccepted":false,"dccOffered":fal
se,"declined":false,"displayText":"TRANSACTION
CANCELLED","eposCheckCard":false,"inProgress":true,"offline":false,"signatur
eRequired":false,"sourceid":"38","success":false,"tokenTransaction":false}
```

6.9.4 Report Example

```
-> report(X)
<- report {“reportType”:”X”, “batchNo”:”1”,
“captureCardCountDebitSales”:”0”, ”captureCardAmountDebitSales”:”0”,
”captureCardCountCreditRefunds”:”0”, ”captureCardAmountCreditRefunds”:”0”,
”debitCardCountDebitSales”:”0”, ”debitCardAmountDebitSales”:”0”,
”debitCardCountCreditRefunds”:”0”, ”debitCardAmountCreditRefunds”:”0”,
”authoriseCardCountDebitSales”:”0”, ”authoriseCardAmountDebitSales”:”0”,
”authoriseCardCountCreditRefunds”:”0”,
-> report(LASTX)
<- report {“reportType”:”LASTX”, “batchNo”:”1”,
“captureCardCountDebitSales”:”0”, ”captureCardAmountDebitSales”:”0”,
”captureCardCountCreditRefunds”:”0”, ”captureCardAmountCreditRefunds”:”0”,
”debitCardCountDebitSales”:”0”, ”debitCardAmountDebitSales”:”0”,
”debitCardCountCreditRefunds”:”0”, ”debitCardAmountCreditRefunds”:”0”,
”authoriseCardCountDebitSales”:”0”, ”authoriseCardAmountDebitSales”:”0”,
”authoriseCardCountCreditRefunds”:”0”,
”authoriseCardAmountCreditRefunds”:”0”, “receipt”: [RECEIPT_LINES]}
```

```
-> report(Z)
<- report {“reportType”:”Z”, “batchNo”:”1”,
“captureCardCountDebitSales”:”0”, ”captureCardAmountDebitSales”:”0”,
”captureCardCountCreditRefunds”:”0”, ”captureCardAmountCreditRefunds”:”0”,
”debitCardCountDebitSales”:”0”, ”debitCardAmountDebitSales”:”0”,
”debitCardCountCreditRefunds”:”0”, ”debitCardAmountCreditRefunds”:”0”,
”authoriseCardCountDebitSales”:”0”, ”authoriseCardAmountDebitSales”:”0”,
”authoriseCardCountCreditRefunds”:”0”,
”authoriseCardAmountCreditRefunds”:”0”, “receipt”: [RECEIPT_LINES]}
```

```
-> report(LASTZ)
<- report {“reportType”:” LASTZ”, “batchNo”:”1”,
“captureCardCountDebitSales”:”0”, ”captureCardAmountDebitSales”:”0”,
”captureCardCountCreditRefunds”:”0”, ”captureCardAmountCreditRefunds”:”0”,
”debitCardCountDebitSales”:”0”, ”debitCardAmountDebitSales”:”0”,
”debitCardCountCreditRefunds”:”0”, ”debitCardAmountCreditRefunds”:”0”,
”authoriseCardCountDebitSales”:”0”, ”authoriseCardAmountDebitSales”:”0”,
”authoriseCardCountCreditRefunds”:”0”,
”authoriseCardAmountCreditRefunds”:”0”, “receipt”: [RECEIPT_LINES]}
```

6.10 Error Responses & recommendations

The same error may be returned for different reasons; the returned txt will give the specific reason. The recommended Action should be based on the error number

Num	Error	Returned Text String(s)	Description	EPOS Recommended Action
100	API Error	Command: *+ message +* is not supported JSON Message Format Error Invalid Format - expected: *Expected Content* *Invalid Message Content*	There was an error using the API. This is likely to be a formatting error and will need to be corrected before the transaction can be attempted again	Development error. This should never occur once development and certification is complete.
101	Command Timed Out	Command timed out	The PED took more than twenty five seconds to respond to the previous command	This is a transient error, retry after 15 seconds, if the error persists, reboot the N-Genius device. If this occurs during a transaction, the result of the transaction can be retrieved once connectivity has been restored using the getResult command with the correct source ID.
110	Terminal Busy	Previous command still in progress	The PED is currently processing another command. Wait for a response from the PED before sending subsequent commands	The EPOS and PED are out of sequence. The EPOS should send getStatus messages at 3 seconds intervals until the response shows the PED is no longer busy. getTransactionResult can then be used to determine the result of the last transaction
200	Processing Error	*Message Content* Cannot cancel transaction	There was an error processing the message. The error may be transient and a retry should be attempted.	This is a transient error, retry after 15 seconds, if the error persists, reboot the N-Genius device. This can be caused by the PED receiving a cancel request after the transaction has been sent to the host. In this scenario, if it is still necessary to cancel the transaction, it can be voided.
201	Cannot start transaction	PED is not ready to transact, please try again Cannot cancel transaction	The N-Genius Device may still be initializing. The N-Genius Device is not in a state where it can start a transaction. A retry should be attempted.	This should only occur after the PED has been reset, if the error persists for more than 2 minutes, there may be an issue with the PED. No transaction will have

		Cannot start transaction		been started so there is no need to cancel or void any transaction.
202	Invalid Transaction Type	Invalid type: *Transaction Type*	The EPOS attempted to start a transaction for a type that was not recognized. Please check your request for capitalization and typographical errors.	Development error. This should never occur once development and certification is complete.
203	Missing Parameter	Missing Parameter: Transaction Type Missing parameter: *Parameter* Parameter not supplied	The EPOS attempted to start a transaction without supplying a transaction type. Please check the EPOS logic.	Development error. This should never occur once development and certification is complete.
204	Operation can't be performed	*Recalled Txn: - why the operation can't be performed* *Parameter Error :- Detail of what is wrong with the parameter*	The EPOS parameters supplied were either incorrectly formatted, or not valid for the operation that was being attempted. Examples are: Unexpected parameter Invalid amount Invalid parameter value: expected 'true' or 'false' Recalled TXN is an Alipay transaction. Please use Alipay void. Recalled TXN is a Refund. Please use refund void. Recalled TXN is a void transaction. Recalled TXN has been settled. Please perform action as a refund. Recalled TXN was declined. Recalled TXN has already been voided. Recalled TXN is an Alipay transaction. Please use Alipay void. Recalled TXN is not a Refund. Please use regular void.	A transaction is not possible with the supplied parameters. The error should be displayed to the operator, so the parameter can be corrected and the transaction re-attempted.

			Recalled TXN has been settled. Please perform action as a refund. Recalled TXN was declined. Recalled TXN has already been voided. Recalled TXN is not an Alipay transaction. Recalled TXN was not performed today. Please perform action as a refund. Recalled TXN was declined. Recalled TXN is a refund. Refund transactions cannot be voided. Recalled TXN is a void transaction. Recalled TXN has already been voided. Recalled TXN is not a Pre-Auth Sale Transaction Recalled TXN has been voided. Recalled TXN has already been completed. Recalled TXN is not a Pre-Auth Sale Transaction Recalled TXN has already been voided. Recalled TXN has been completed. Recalled TXN is not a Tip Sale Transaction Recalled TXN has been voided. Recalled TXN has already been completed. Tip must be greater than or equal to 0 Tip must be no more than n of transaction amount Total Amount (Txn + Tip) must be no more than n Tip Amount must be no more than n Unexpected parameter Offline Limit would be exceeded Invalid amount	
205	EOD Required	Could not start transaction: EOD Required	Please perform a Z Total Report	The N-Genius Device is configured to limit the number of days between Z reports. This limit has been exceeded so a Z report must be performed before another transaction can be started.
206	Reversal in Progress	Could not start transaction: Cannot establish connection to host	The N-Genius Device is currently attempting a reversal. Transactions cannot be started until the reversal has been completed.	This error will persist until the reversal is successfully processed by Network International. Retry once the connection to Network International is available. If the

				problem persists and connectivity is available, contact Network International
207	Original transaction not found	Transaction Not Found Original txn not found	The N-Genius Device was unable to find the original transaction. Prompt user to re-enter transaction reference.	A void has been attempted and the original transaction not found on this Device. Retry with the correct original transaction reference. If this is received when EPOS is trying to recover from a failure, it can be assumed that the transaction was never received by the PED so no further action is necessary.
208	Report Failed	Report Failed	The N-Genius Device was unable to perform the requested report. Check error message and try again.	Development error. This should never occur once development and certification is complete.
209	Not Supported	Offline Transaction not supported Offline Pre-Auth not supported Offline Alipay Transaction not supported	The transaction specified is not supported by the device. Check error message and try again.	A transaction type that is not supported by Device configuration was attempted. Check with Network International to ensure the site is configured correctly
210	Offline Unavailable	Offline transaction limit + *Limit Value* + has been exceeded Offline transaction limit + *Limit Value* + Hours) has been exceeded	Offline transactions are not supported by the device. Check error message and try again.	An offline transaction was attempted but the device configuration does not permit offline transactions. Check with Network International to ensure the site is configured correctly
211	Report Empty Batch	Empty Batch	There are no transactions in the current batch, the report could not be completed	There are no transactions available for the report attempted. Z report is not required if no transactions have been performed so no retry is required
212	Report Need Offline Upload	End of Day cannot be completed until all offline transactions are uploaded	Reports cannot be run until all offline transactions have been uploaded	The N-Genius device has been performing offline transactions and has not yet been able to upload them. Wait until connectivity is restored and all the offline transactions have been uploaded before trying again.

213	Report Tip Complete Required	Tip Complete must be run against all sale transactions before Z-report can be created	Reports cannot be run until TIP completion has been run on all Tier 2 Sales	Tier 2 tip transactions that have not been completed are stored on the device. Complete or void all Tier 2 tip transactions before trying again
214	Report Invalid Type	Invalid report type	Invalid report type provided	Development error. This should never occur once development and certification is complete.
220	Duplicate Transaction	Duplicate Source ID not allowed + *source id*	The Source ID provided is used by another transaction in the current batch	An attempt has been made to start a transaction with the same source ID as one that has already been performed in the current batch. Perform an EOD and try again, or use a different source ID
300	Communication Error	PED Not Connected	There was a communication error, the message should be retried.	This is a transient error, retry after 15 seconds, if the error persists, reboot the N-Genius device. If this occurs during a transaction, the result of the transaction can be retrieved once connectivity has been restored using the getResult command with the correct source ID.
400	Driver Logon Error	Error logging driver onto PED	There was an error while attempting to log the driver onto the PED. Potential causes are: * Driver already logged on * Missing logon parameters	Driver already logged on – before a new driver can be logged onto the PED, the previous driver must logoff. Missing logon parameters - Development error. This should never occur once development and certification is complete.
401	No logged on Driver Error	No driver logged on	Returned when transactions are attempted without first performing a successful taxiLogon() command.	Ensure that the driver has successfully logged on before attempting transactions.
402	Invalid Parameter	Final amount must be no less than the minimum redemption amount of	Returned when transaction amount supplied is less than the minimum redemption amount set	Ensure the amount supplied is greater than the minimum redemption amount set

6.11 Power Failure or PED reset

In the event of a power failure or PED reset, the PED will reverse any transaction that was in progress before the next transaction is able to be started.

To determine the result of the transaction a get Transaction Result message should be sent by the EPOS once the PED is communicating.

6.10 Hardware Failure or any other communication error

In the event of a Hardware failure or any other communication error PED / NGPAS/androidPas will reply with proper

error message as mentioned in section 6.8.

NI recommends to build a timeout mechanism at EPOS end if transaction is not completed within 90

seconds. Also timer value should be incremented only once with another 60 seconds with an update

transaction response if sent from EPOS.

If a transaction is not complete and a timer reached 90 seconds (150 seconds in case if incremented on

update request) then EPOS should cancel the current transaction by following a Cancel Transaction

mechanism mention in section 6.5.2 and initiate a new transaction

7 Setting up Development Environment

Please contact your NI Representative to get the development software environment.

The environment provides an example dummy EPOS application, a command line tool for sending N-Genius commands, the NGPAS/AndroidPas service and a simulator that runs on an N-genius device to simulate a payment transaction.

7.1 Getting Started

To run the dummy EPOS Simulation tool (epossim). See section 11 for EPOS SIM Guide.

NI provides an EPOS simulation tool as a reference implementation. The EPOS simulation tool is a command line tool that can simulate an EPOS transaction. All communication is logged to a file so developers can see the commands and responses received in different scenarios. The EPOS simulation tool can be used for IP and USB communication. Where USB communication is used the NGPAS must be running as a service, where ip communication is used the command line option `–cert ng.cert` with a valid cert file must be used. When calling epossim the application calls the getTransactionAndTypes and returns the result.

```
epossim

1->Card Sale - Sale

2->Phone Pay - Sale

3->End of day - Admin

4->Refund - Refund

5->Voucher – Refund
```

Once a type has been selected, the tool will prompt for parameters, required for that type before starting a transaction. If once a transaction has been started further parameters are required prompts will be displayed. Please note that if the program is terminated unexpectedly, the transaction on the PED will eventually timeout until connection is re-established, where the timed-out transaction will sent to Transaction Manager as an auto-void.

Overleaf is a flow for a standard card transaction:

```
epossim

1->CardSale - Sale

2->PhonePay - Sale

3->EndofDay - Admin

4->Refund – Refund

5->Voucher – Payout

6->Bill - BillPay
```

```
Amount:3500

Currency:AED

Present Card

Enter PIN

"success": true,

"resultCode": "00",

"authCode": "005924",

"displayText": "Successful Transaction",

"receiptLines": [

{"style":"bold-text medium-text align-center-text", "text":"ALL DAY
MINI MART LLC"

{"style":"bold-text medium-text align-center-text", "text":"PURCHASE"

{"style":"small-text small-text", "textLeft":"Transaction Date",
"textRight":"17/07/2018"}],

"rrn": "000000022767"

"token": "5423786542685678"

"cardType": "MASTER"
```

7.2 Nginstall (IP connection only)

To set up IP communication between the N-Genius device and the EPOS simulation tool the N-Genius device must be configured with a fixed IP address. To do this switch on the device. Select Wifi settings option to set the IP address.

Run the utility provided by NI Nginstall with the command line option of the ip address of the N-Genius device. E.g.

```
nginstall
```

This will start the paring process to download the certificates and private key to communicate with the EPOS simulation tool.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.

8 Design Considerations for Host System

NI's N-Genius POS system removes much of the complexity of integrating payments with POS applications. The functional changes that need to be made to a host system to provide integrated payments are as follows:

- Implement the Payment Interface
- Manage Cash and Card totals
- End of Day
- Printing Receipts
- Audit Logging

NI provides a test environment and test scripts for developers to validate that they have correctly implemented each of these functions.

8.1 Implementing the Payment Interface

When a host system successfully sends a starts a transaction it MUST keep polling for a response until the transaction is either successful or failed. The host can cancel a transaction, but MUST check the transaction outcome (which may be success) as a transaction could have completed before the cancellation request was received.

If an N-Genius device is unplugged before the host has received a transaction outcome, the host system should prompt the operator to reattach the N-Genius device and keep polling for a transaction outcome. The transaction may have completed, before the device was unplugged.

If the host system is power cycled before it has received a transaction outcome, it must check the transaction outcome on power up. If a transaction completed, it must be voided and the event logged.

8.2 Managing Cash and Card Totals and End of Day

The host will need to keep a record of totals for balancing the till at the end of day. Transactions return sale amounts and optionally, if supported by the EPOS, tip amounts and a cashback amounts. Cashback and tip amounts must be kept as separate totals for end of day reporting.

If the host EPOS systems support Refund, BillPay and Payout transactions these totals should also be maintained separately for end of day reporting.

8.3 End of Day

At End of Day the EPOS should send and end of day transaction to N-Genius. N-Genius will return an end of Day receipt to be printed. N-Genius can be configured to auto end of day at a specific cut off time if required, in which case it is not necessary to send end of day messages. All transactions will return the N-Genius business day.

8.4 Audit log

The Host application should keep an audit log of all communication to and from N-Genius for trouble shooting.

NI will require to see audit logs as part of application certification.

N-Genius will never return card data or data that should not be logged.

8.5 QR Code Scanning

Some payment methods (eg. Alipay) can be initiated by scanning a barcode or QR code. To validate that the barcode scanner being used is correctly configured, please validate that QR codes from the following web site are scanned successfully. You may contact the NI representative for more details on Alipay acceptance and how to get the Sandbox app

https://global.alipay.com/service/transaction_QR_Code/1

9 Appendices

9.1 Data Elements

ELEMENT	DATA TYPE	NOTES
success	Boolean	See point 9.3
declined	Boolean	See point 9.3
signatureRequired	Boolean	
amount	String	$10.00 = “1000”
cashback	String	
tipAmount	String	
dccAmount	String	
sourceid	String	
type	String	
currency	String	
authCode	String	See point 9.2
resultCode	String	
resultCodeDescription	String	00 - Description
rrn	String	
token	String	
tokenTransaction	boolean	Will request a token from TM and will also result in a different (token based) txn flow for completions, voids, refunds
cardType	String	
cardReadMode	String	SCR, CTLS, MSR, MSRC, MSRF or MAN
dccCurrencyCode	String	
panMasked	String	
track2data	String	Passed in during startTransaction, never returned by PED. To be used if a card was swiped on the EPOS.
cvm	String	PIN, SIGN or NONE
dccOffered	boolean	
dccAccepted	boolean	
dccDecimalPlaces	String	
dccExchangeRate	String	
dccMargin	String	
dccCurrencyCode	String	
dccExchangeRateProvider	String	
custReceipt	TxnReceiptLine[]	See point 9.4
merchReceipt	TxnReceiptLine[]	See point 9.4

offline	Boolean	Set “true” to perform transaction offline for later upload.
eposCheckCard	Boolean	Set “true” to perform post-insertion checks.
maxCashBack	String	
maxTipAmount	String	
newAmount	String	
discountAmount	String	
barcode	String	
refundApprovalCode	String	
voidOriginalId	String	
inProgress	Boolean	
complete	Boolean	
cardInserted	Boolean	
displayText	String	
result	String	
parameter	String	
parameterType	String	Numeric, alphanumeric, barcode or boolean
parameterValue	String	
userId	String	10 char max

9.1.1 Report Data Elements

ELEMENT	DATA TYPE	NOTES
reportType	String	
batchNo	Integer	
success	Boolean	
captureCardCountDebitSales	Integer	These values do not yet correspond to any sales data
captureCardAmountDebitSales	String	
captureCardCountCreditRefunds	Integer	
captureCardAmountCreditRefunds	String	
debitCardCountDebitSales	Integer	These values correspond to sales
debitCardAmountDebitSales	String	
debitCardCountCreditRefunds	Integer	These values correspond to refunds
debitCardAmountCreditRefunds	String	
authoriseCardCountDebitSales	Integer	These values correspond to voids
authoriseCardAmountDebitSales	String	
authoriseCardCountCreditRefunds	Integer	
authoriseCardAmountCreditRefunds	String	

9.2 Transaction Auth Codes

00	Approved
01	Call issuer
02	Call issuer
03	Invalid Merchant
05	Do Not Honor
12	Invalid Txn
13	Invalid Amount
14	Invalid Card
19	Retry the Txn
25	Declined
30	Format Error
31	Unsupported Txn
41	Please Call - LC
43	Please Call - CC
51	Declined
54	Expired Card
55	Incorrect Pin
58	Txn not allowed
65	Perform Contact Txn
74	Txn Unavailable
78	Invalid amount
89	Invalid terminal
91	Auth timed out
94	Already voided / Duplicate TXN
95	Txn Cancelled
96	Declined
97	Signature Mismatch
98	Card Removed
99	Comms Error

9.3 Success / Declined / Status Combos

The declined flag indicates the transaction was declined on the PED.

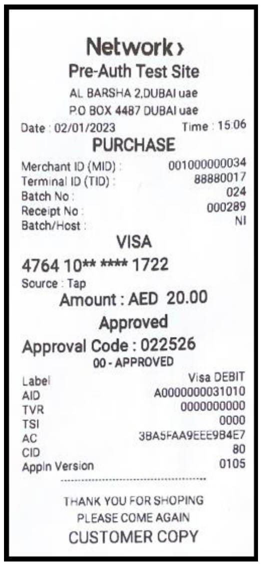
SUCCESS	DECLINED	STATUS
True	False	00 (0x3030) Approved
False	False	01 (0x3031) Call Issuer
False	False	51 (0x3531) Declined
False	True	96 (0x3936) Offline Declined (e.g. Wrong PIN)

9.4 Receipt Lines

Receipt lines can be made up of text and image elements. Both have style attributes.

Style attributes describe how text and images should be styled and aligned.

There are two types of receipt lines, one column and two columns. One column receipt lines have a single 'text' or 'img' element. Two column receipt lines have 'textLeft' and 'textRight' elements.



Network›
Pre-Auth Test Site
AL BARSHA 2,DUBAI uae
P.O BOX 4487 DUBAI uae

```
Date : 02/01/2023                Time : 15:06
```

PURCHASE

```
Merchant ID (MID) :          001000000034
Terminal ID (TID) :              88880017
Batch No :                            024
Receipt No :                       000289
Batch/Host :                           NI
```

VISA
4764 10** **** 1722
Source : Tap
Amount : AED 20.00
Approved
Approval Code : 022526
00 - APPROVED

```
Label                          Visa DEBIT
AID                        A0000000031010
TVR                            0000000000
TSI                                  0000
AC                       3BA5FAA9EEE9B4E7
CID                                    80
Appln Version                        0105
```

THANK YOU FOR SHOPING
PLEASE COME AGAIN
CUSTOMER COPY

One column example (first line of image).

{"style":" bold-text medium-text align-center-text", "text":"ALL DAY MINI MART LLC"}

Two columns example (third line of image).

{"style":" small-text small-text", "textLeft":"Transaction Date", "textRight":"17/07/2018"}

Example Image element:

Receipt lines with image elements have width, height and src attributes.

{"style":"align-center-text", "width":"350", "height":"116", "src":"data:image/bmp;base64,iVBORw0KAA8 … BJRg=="}

Note: NI would pass first 6 and last 4 digits of the Card number, however EPOS should only print the last 4 digits of the Card number on receipt printed by ECR. NI would also pass last 2 digits of Expiry date; however, EPOS should not print the Expiry date on the receipt.

Example Receipt Lines:

```
"receiptLines":[{"style":"bold-text medium-text align-center-text", "text":"ALL
DAY MINI MART LLC"},{"style":"small-text align-center-text",
"text":""},{"style":"bold-text medium-text align-center-text",
"text":"PURCHASE"},{"style":"small-text align-left-text",
"text":""},{"style":"small-text small-text", "textLeft":"Transaction Date",
"textRight":"17/07/2018"},{"style":"small-text small-text",
"textLeft":"Transaction Time", "textRight":"15:55"},{"style":"small-text small-
text", "textLeft":"Merchant ID (MID)",
"textRight":"001180220030"},{"style":"small-text small-text",
"textLeft":"Terminal ID (TID)", "textRight":"12345"},{"style":"small-text small-
text", "textLeft":"Receipt No", "textRight":"000012"},{"style":"small-text small-
text", "textLeft":"Batch / Host", "textRight":"NI"},{"style":"small-text small-
text", "textLeft":"Batch No", "textRight":"001"},{"style":"small-text small-
text", "textLeft":"Card Type", "textRight":"MASTER"},{"style":"small-text small-
text", "textLeft":"Card Number", "textRight":"5413 33** ****
3537"},{"style":"small-text small-text", "textLeft":"Card Expiry Date",
"textRight":"**12"},{"style":"small-text small-text", "textLeft":"Source",
"textRight":"Tap"},{"style":"small-text align-left-text",
"text":""},{"style":"bold-text medium-text small-text", "textLeft":"Amount",
"textRight":"AED  333.33"},{"style":"bold-text small-text small-text",
"textLeft":"Approval Code", "textRight":"191919"},{"style":"bold-text small-text
small-text", "textLeft":"Txn Status", "textRight":"Approved"},{"style":"bold-text
small-text small-text", "textLeft":"Status Reason", "textRight":"00 -
Approved"},{"style":"x-small-text align-left-text", "text":"---------------------
-----------------------"},{"style":"x-small-text align-left-text", "text":"------
-------------------------------------"},{"style":"small-text align-left-text",
"text":""},{"style":" small-text align-center-text", "text":"THANK
YOU"},{"style":"small-text align-left-text", "text":""},{"style":"bold-text
small-text align-center-text", "text":"<< CUSTOMER COPY >>"}]
```

9.5 Running NGPAS as a Service

9.5.1 Linux

As root user, create the file /etc/systems/system/ngpas.service containing the lines below:

```
[Unit]

Description=ngpas

[Service]

Type=simple

User=ngapp

# or account you want the service to run as

WorkingDirectory=/home/ngapp

+ExecStart=/usr/bin/java -Xmx256m -jar ngpass.jar 8080 192.168.2.18 8081

Restart=on-failure

[Install]

WantedBy=multi-user.target
```

Secondly, notify systemd of the new service file:

systemctl daemon-reload

and enable it, so it runs on boot:

systemctl enable ngpas.service

You can use the following commands to start/stop your new service:

systemctl start ngpas

systemctl stop ngpas

systemctl restart ngpas

systemctl status ngpas

9.5.2 Windows

Run a command prompt as administrator and navigate to the service directory.
Run ngpas_service_install.bat
Run ngpas_service_config.bat

Go to Startup tab and change the arguments to be as follows, changing 192.168.2.11 to be your ped IP or com port.

```
start

8085

192.168.2.11

8081
```

Save your settings by clicking 'apply', then go to the general tab where you can start and stop the service.

Note: in case of WIFI/MPLS connectivity option, Merchant would come under PCI scope.

9.6 Sample Transactions

9.6.1 Sale Transaction

Sale Transaction.txt

9.6.2 Void Transaction

Void Transaction.txt

9.6.3 Refund Transaction

Refund Transaction.txt

9.6.4 Alipay Sale Transaction (Scan code in mobile APP)

Alipay_Sale_Customer Presented QR.txt

9.6.5 Alipay Sale Transaction (Scan code in PED)

AliPay_Sale_Merchant Presented QR.txt

9.6.6 Alipay Void Transaction

AlipayVoid.txt

9.6.7 Alipay Refund Transaction

AlipayRefund.txt

9.6.8 Visa Qr Sale (Scan code in PED)

VisaQrSalePedQr.txt

9.6.9 Visa Qr Refund

VisaQrRefund.txt

9.6.10 Mastercard Qr Sale (Scan code in PED)

MCQrSalePedQr.txt

9.6.11 UPI Sale (Scan code in PED)

UPI Merchant PresentedQR.txt.txt

9.6.12 UPI Void

UPI Void.txt

9.6.13 UPI Refund

UPI Refund.txt

9.6.14 UPI Sale (Scan consumer QR)

txnEposUpiSale_consumer_qr.txt

9.6.15 Pre-Auth Sale

PreAuth.txt

9.6.16 Pre-Auth Completion

PreAuthCompletion.txt

9.6.17 Pre-Auth Completion Void

PreAuthCompletionVoid.txt

9.6.18 Pre-Auth Cancel

PreAuthCancel.txt

9.6.19 EPP Sale

EPP Sale.txt

9.6.20 EPP Void

EPP Void.txt

9.6.21 ADCB TouchPoint Balance check

ADCB TouchPoint Balance check.txt

9.6.22 ADCB TouchPoint Redeem

ADCB TouchPoint Redeem.txt

9.6.23 ADCB TouchPoint Void

ADCB TouchPoint Void.txt

9.6.24 TerraPay Sale

TerraPaySale.txt

9.6.25 TerraPay Refund

TerraPayRefund.txt

9.6.26 XLS Sale full redemption

Xls sale full redemption.txt

9.6.27 XLS Sale partial redemption (Card payment + XLS points)

Xls sale partial redemption (Cardpay

9.6.28 XLS Void (Only to Void XLS full redemption)

XLS Void (Full redemption void).txt

9.6.29 Full Void (Card Payment + XLS points)

Full void (Cardpayment + XLS

9.6.30 WeChatPaySale – Customer Presented QR

WeChatPaySale_CustomerPresentedQR.txt

9.6.31 WeChatPaySale – Merchant Presented QR

WeChatPaySale_MerchantPresentedQR.txt

9.6.32 WeChatPayRefund

WeChatPayRefund.txt

10 Offline Authorization (Not available as standard feature)

This will allow the EPOS to use PED's in stand in authorization mode (offline mode).

The EPOS will control when the terminal is in offline mode. If offline mode is not specified, the PED will default to the normal online mode.

As authorizing cards offline carries an increased risk, it should only be used in exceptional circumstances.

A new parameter will be added to the start transaction request "offline:true".

If this is present and true the PED will attempt to authorize that transaction offline.

No indication will be made to customer that the PED is in offline mode

There will be a non-obvious indication on the receipt that a transaction was authorized offline

The merchant will be prompted to enter a 6-digit approval code, when not provided by voice referral.

Voids of offline transactions should be performed offline, even if the terminal is back in online mode.

Offline transactions should be uploaded from the PED 'oldest first'

The PED will attempt to upload stored offline transactions automatically.

The Upload will be an advice message to the existing Base24 interface, format TBD

DCC will not be supported in offline mode

Alipay will not be supported in offline mode

Online transactions will take priority over offline uploads, if a transaction is in progress, no offline uploads will be started.

EOD will not be available in offline mode

Central configuration for each PED will have the following parameters:

Offline mode allowed

Schemes allowed and individual transaction limits per scheme

If the transaction limit is hit, the PED will prompt for voice referral. (See below)

Maximum total offline limit (total value of stored transactions)

If this limit is hit, the PED will prompt for voice referral. (See below)

Maximum offline duration (How far in the past the oldest offline transaction can be)

If this limit is hit, the PED will display a message and prevent further transactions until the offline transactions have been uploaded

Upload frequency (the max number of offline transactions that can be uploaded per min)

Voice Referral

In the event that the transaction limit would be exceeded for the current transaction, the PED will prompt for the 'voicereferralcode' parameter after card insertion.

This parameter should be responded to with a six-character approval/auth code retrieved when performing voice referral authorization with the host.

Once this approval code is retrieved, the PED will continue as normal with the offline transaction, however instead of prompting the user to enter an approval code, it will use the voice referral auth code when submitting the advice message to Base24, and on all receipts

This step is <u>only</u> carried out if the PED is in offline mode, or if the transaction is a refund.

11. EPOS SIM GUIDE

11.1 Introduction

The EPOS simulator is designed to test the functionality of the N-Genius PED EPOS functionality through the NGPAS/AndroidPas interface.

11.2 Application

The application is a windows command line utility written in go.

It allows the user to send commands to the PED and view the responses.

It connects to a NGPAS/AndroidPas service running on the local machine.

Configuration is passed in using command line parameters

epossim.exe –? will give a list of parameters

No parameters are required for normal operation.

11.3 Usage

Run NGPAS/AndroidPas on the computer and ensure a PED is connected

Run epossim

When epossim runs, it connects to the PED.

Information about the connected PED is shown in the console.

The list of available transactions is then displayed (this is retrieved from the PED)

Commands can be sent to the PED by entering them on the command line

The response is shown in the console window. If more information is needed, this is shown and the details (e.g. auth code) should be entered on the command line.

If an error message is displayed 'ok' should be entered on the command line to allow the PED to continue.

A list of commands can be shown by typing *help* into the console

11.4 Additional Command Information

11.5 txn

Start a transaction, should be followed by one of the available transaction types e.g.

txn *eposSale*

If no parameters are specified, the user will be prompted through the steps of the transaction

Parameters available are:

-sourceId

-amount

-refundApprovalCode

-voidOriginalId

-uniqueOrderId

-alipayRefundId

-visaQrRefundCode

-UpiQrcVoucherNo

-barcode

-track2

-offline

-checkCard

-maxCashback

-maxTipAmount

-tokenTransaction

To start an offline transaction, ensure the TMS is correctly configured and that your PED has completed an EOD (if necessary) enter the below command:

txn eposSale -offline

To start a track2 sale:

Txn eposSale -track2 XXX (with XXX being the track2 data)

11.5.1 result

Gets the result of the current transaction if one is in progress. If the result of a previous transaction is needed, it should be followed by the source id e.g.

result 1234

11.5.2 batch

Performs an End of Day, should be followed by a X or Z, e.g.

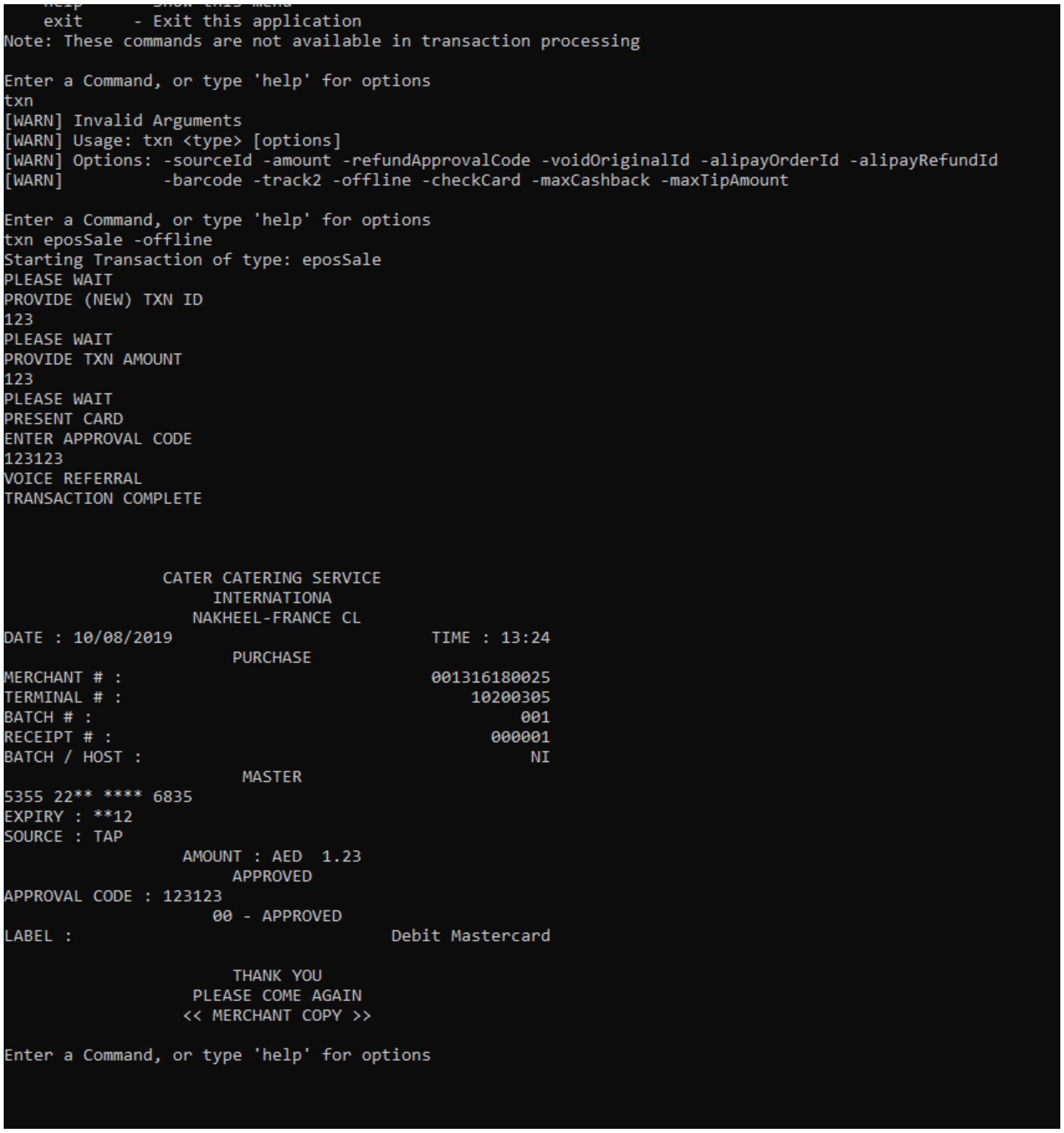
batch X

11.5.3 cancel

Attempts to cancel a transaction. This can be sent when in the middle of another transaction

11.6 Example

The below example shows EPOS Sim completing an offline txn



```
    exit     - Exit this application
Note: These commands are not available in transaction processing

Enter a Command, or type 'help' for options
txn
[WARN] Invalid Arguments
[WARN] Usage: txn <type> [options]
[WARN] Options: -sourceId -amount -refundApprovalCode -voidOriginalId -alipayOrderId -alipayRefundId
[WARN]          -barcode -track2 -offline -checkCard -maxCashback -maxTipAmount

Enter a Command, or type 'help' for options
txn eposSale -offline
Starting Transaction of type: eposSale
PLEASE WAIT
PROVIDE (NEW) TXN ID
123
PLEASE WAIT
PROVIDE TXN AMOUNT
123
PLEASE WAIT
PRESENT CARD
ENTER APPROVAL CODE
123123
VOICE REFERRAL
TRANSACTION COMPLETE



                CATER CATERING SERVICE
                     INTERNATIONA
                   NAKHEEL-FRANCE CL
DATE : 10/08/2019                          TIME : 13:24
                       PURCHASE
MERCHANT # :                               001316180025
TERMINAL # :                                   10200305
BATCH # :                                           001
RECEIPT # :                                      000001
BATCH / HOST :                                       NI
                        MASTER
5355 22** **** 6835
EXPIRY : **12
SOURCE : TAP
                  AMOUNT : AED  1.23
                       APPROVED
APPROVAL CODE : 123123
                     00 - APPROVED
LABEL :                                Debit Mastercard

                       THANK YOU
                   PLEASE COME AGAIN
                  << MERCHANT COPY >>

Enter a Command, or type 'help' for options
```

12. Android Pas

There is a version of NGPAS which has been written to run as an apk on android devices. This version functions and is used in the same way as described by this document with a small number of differences:

- Android pas is launched like normal app on an android device and no longer required batch file
- The app will run in the background as a service allowing the android device to continue to be used.
- There is a config file which will need to be placed on the Sdcard.
- For USB mode the config file should specify "serial" for the peds Ip address and androidpas will scan for known devices when it starts and attempt to connect automatically.

13. Troubleshooting

13.1 NGPAS/AndroidPas won't connect to the Ped

This is the most common problem and can be caused by several things:

Ensure the PED is in the correct mode for the connection method being used (i.e., CDC enabled for serial)

Confirm the Ped's Ip / port is specified and is correct.

If using Wi-Fi, ensure the Ped is reachable on the network of the machine running NGPAS/AndroidPas.

13.2 When NGPAS/AndroidPas is connected, the Ped isn't responding to pings

The Ped will only respond when NextGen is running and in a valid epos available mode.

13.3 Command Timeouts are seen during transactions

There are several common reasons for command timeouts to occur

There may be a genuine connection issue causing 1 of the components in the chain to not receive / respond to messages.

The Ped may be busy, this will occur if the Ped is in the middle of a process and further commands are send out of order.

It can also happen if the Ped is waiting for a user input which it hasn't yet received / rejects the subsequent commands due to not being valid for the expected input request.

13.4 Connectivity issues: COM Port Validation

As the COM port is defined by operating system, there are high chances of Changes in COM Port due to USB Port changes.

To avoid this issue, make sure that Device COM port is inline with the COM port configured in NGPAS/AndroidPas config

13.5 Connectivity issues: PED Detection

- Disconnect and connect the Cable back
- Try restarting the NGPAS/AndroidPas Application.
- Try restarting the PED