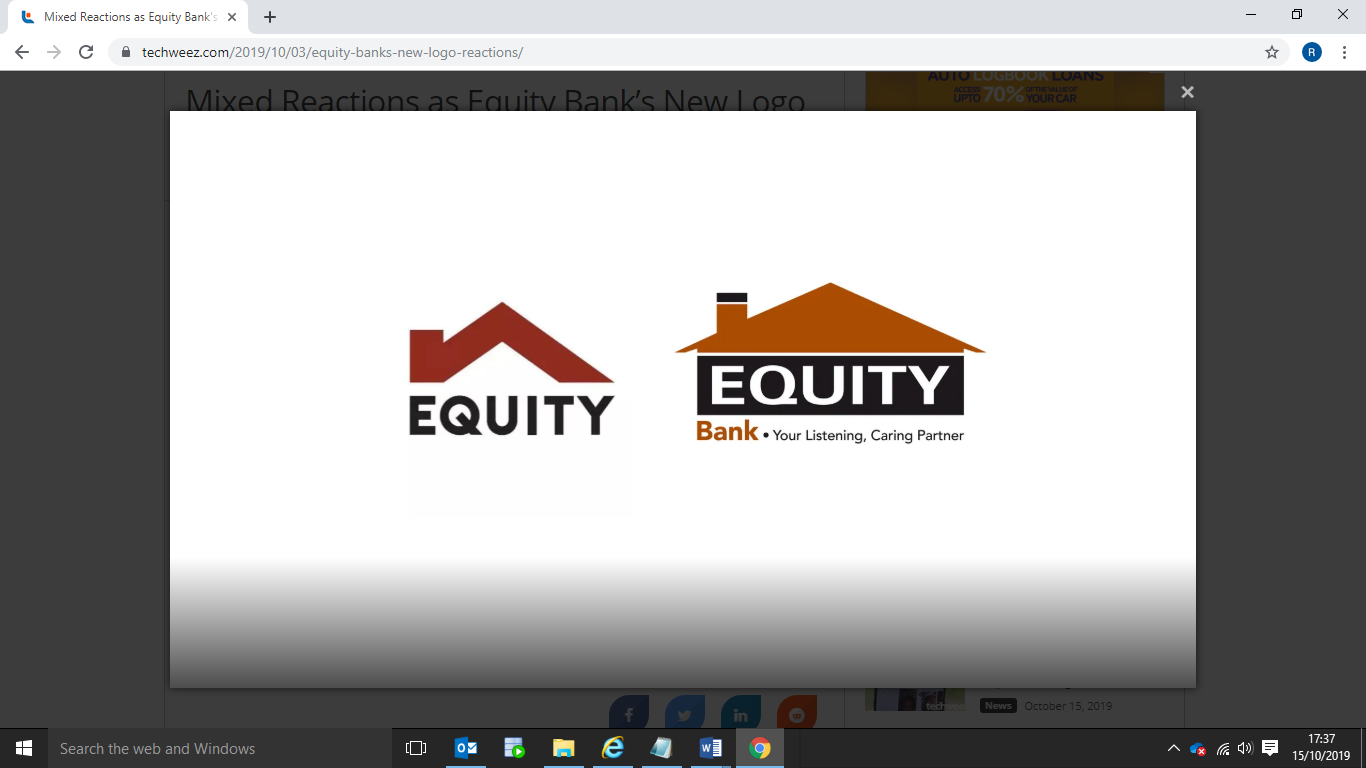
**POS INTERGRATION**

**Author:**



**HEAD OFFICE**

Equity Centre, Hospital Road, Upper Hill.

Address: P.O. Box 75104-00200, Nairobi, Kenya

Email: info@equitybank.co.ke

Website: www.equitybank.co.ke

**CONFIDENTIALITY DISCLAIMER**

THIS DOCUMENT AND THE INFORMATION IN IT ARE PROVIDED IN CONFIDENCE, FOR THE SOLE PURPOSE OF EQUITY BANK LIMITEDAND MAY NOT BE DISCLOSED TO ANY THIRD PARTY OR USED FOR ANY OTHER PURPOSE WITHOUT THE EXPRESS WRITTEN PERMISSION OF EQUITY BANK LIMITED. ANY FORM OF REPRODUCTION, DISSEMINATION, COPYING, DISCLOSURE, MODIFICATION, DISTRIBUTION AND OR PUBLICATION OF THIS MATERIAL IS STRICTLY PROHIBITED.

The contents of this document have not been independently verified, and they do not purport to be comprehensive, or to contain all the information that a prospective investor may need. No representation, warranty or undertaking, expressed or implied is or will be made or given and no responsibility or liability is or will be accepted by Equity Bank Limited or by any of its directors, employees or advisors in relation to the accuracy or completeness of this document or any other written or oral information made available in connection with the Company.

Contents

[Article I. Contents 2](#_Toc187830904)

[**1.** **INTRODUCTION** 3](#_Toc187830905)

[**2.** **requirements fOR the BANK:** 3](#_Toc187830906)

[**2.1** **sOFTWARE WORKS WITH THE FOLLOWING OS:** 3](#_Toc187830907)

[**3.** **requirements for the merchant** 3](#_Toc187830908)

[**3.1** **system requirements** 3](#_Toc187830909)

[Section 1.01 3\_1\_1 OS Windows 4](#_Toc187830910)

[Section 1.02 3\_1\_2 OS DOS 4](#_Toc187830911)

[Section 1.03 3\_1\_3 OS Linux 4](#_Toc187830912)

[**4.** **HIGH LEVEL ARCHITECTURE OF POS** 4](#_Toc187830913)

[**5.** **STEPS INVOLVED WHILE TRANSACTING** 5](#_Toc187830914)

[**6.** **DATA FORMATS FOR DIFFERENT OPERATIONS** 5](#_Toc187830915)

[**Section 1.04** **PURCHASE** 5](#_Toc187830916)

[**Section 1.05** **CLOSE SHIFT** 6](#_Toc187830917)

[**Section 1.06** **STATEMENTS** 8](#_Toc187830918)

[**Section 1.07** **RECONCILIATIOn** 9](#_Toc187830919)

# **INTRODUCTION**

In this document, we provide a comprehensive guide for customers looking to integrate with Equity Bank's pin pad POS for payment processing. This guide outlines the entire transaction workflow, including the interactions among the merchant POS, Arcus drivers, and the bank's authentication system, and demonstrates how to handle responses at each stage. By following these instructions, customers can ensure a seamless and secure integration, optimizing payment processes while enhancing the customer experience at the point of sale.

# **requirements fOR the BANK:**

* **ARCUS:**  
  Arcus is a library that is used to integrate merchant POS with POS devices. Merchant POS and POS devices use ARCUS protocol for data transfer.   
  ARCUS functions:  
  • ARCUS exchange protocol for POS equipment data exchange.  
  • ARCUS can be used to connect the Bank Host (Cash Reg. IP mode).  
  • User dialog can be displayed on the Cash Register screen (Cash Reg. Dialog mode).

# **sOFTWARE WORKS WITH THE FOLLOWING OS:**

• Windows (XP SP3 and above).   
• MS DOS (ver. 6.22).   
• Linux.

* **API CLIENT:** The API client is a gateway that retrieves responses from Arcus, which are then accessed by the merchant POS.

**N/B**: End point to be provided during installation.

# **requirements for the merchant**

# **system requirements**

## 3\_1\_1 OS Windows

• OS: Windows (XP and above).

• OS Kernel supporting communication devices with USB (Jungo Driver ver.1.5 and above).

## 3\_1\_2 OS DOS

• Cash Register OS: DOS (6.22 and above with DOS 866 encoding).

• MS Network Client for DOS or with another net driver supporting TCP/IP and Cashier software.

• Cash Register with bank host access via TCP/IP, supporting MSSOCKETS (Microsoft) or SOCKETS.

## 3\_1\_3 OS Linux

• OS: Linux (CentOS/Ubuntu/RedHat/SUSI...).

• OS Kernel supporting communication devices with USB (dev/ttySx, dev/ttyACMx).

# **HIGH LEVEL ARCHITECTURE OF POS**

A diagram of a computer hardware system

Description automatically generated

# **STEPS INVOLVED WHILE TRANSACTING**

1. At the merchant POS, a transaction is initiated, which triggers a call to the API client to handle further processing.

2. The API client receives the transaction amount request and forwards it to Arcus drivers for processing.

3. After Arcus drivers receive the transaction request, it sends it to POS device.

4. Upon receiving the transaction details, the POS device communicates with the bank’s system to authenticate the transaction, ensuring the legitimacy of the request.

5. The bank’s system verifies the transaction credentials. Once authentication is complete, whether the transaction is approved or declined, a response is generated and sent back to the POS device.

6. After receiving the bank's response, the POS device sends this information back to Arcus for further processing and logging.

7. The API client reads and processes the response from Arcus, which contains relevant transaction details.

8. The merchant POS retrieves the response from the API client, which includes the final transaction status and may then update their local records.

# **DATA FORMATS FOR DIFFERENT OPERATIONS**

## **PURCHASE**

When performing a purchase, the merchant POS should call the following endpoint using the **POST** method:  
**Endpoint:** <http://127.0.0.1:9910/v1/purchase>

The API client expects the merchant POS to send a JSON payload containing the transaction amount.

Sample payload:

**REQUEST:**

{

"amount": 10.00

}

**RESPONSES:**

**Successful transaction payload:**

{  
    "code": "000",  
    "message": "Succesfully",  
    "status": "success",  
    "transaction": {  
        "amount": "0.01  KES",  
        "auth\_code": "762924",  
        "code": "00",  
        "message": "Succesfully",  
        "receipt\_no": "5",  
        "rrn": "500813945774",  
        "status": "success",  
        "tid": "00337470",  
        "tran\_date": "08/01/25 12:30:43"  
    }  
}  
**Failed transaction payload:**  
  
{  
    "code": "212",  
    "message": "Refusal of card input",  
    "status": "error",  
    "transaction": {  
        "message": "Refusal of card input",  
        "status": "error"  
    }  
}

## **CLOSE SHIFT**

For closing a shift, the merchant POS should call the following endpoint using the GET method:  
**Endpoint:** <http://127.0.0.1:9910/v1/close/shift>

**Sample response:**

{  
  "status": "Success",  
  "total": "2.04",  
  "transactions": [  
    {  
      "time": "12:45:38"  
    },  
    {  
      "amount": "1.00",  
      "auth": "681114",  
      "card": "447815\*\*\*\*\*\*1214:01",  
      "op": "PAYMENT",  
      "receipt\_number": "6",  
      "rrn": "500713837930"  
    },  
    {  
      "amount": "0.01",  
      "auth": "763629",  
      "card": "447815\*\*\*\*\*\*1214:01",  
      "op": "PAYMENT",  
      "receipt\_number": "5",  
      "rrn": "500813946713"  
    },  
    {  
      "amount": "0.01",  
      "auth": "282924",  
      "card": "519601\*\*\*\*\*\*1911:01",  
      "op": "PAYMENT",  
      "receipt\_number": "5",  
      "rrn": "500814138181",  
      "tid": "00337470"  
    }  
  ]  
}

**When no transaction has happened:**

{

"message": "Shift closing report not found in the output file.",

"status": "error"

}

## **STATEMENTS**

This functionality is used to retrieve statements based on specific dates. The merchant POS should call the following endpoint using the POST method:  
**Endpoint:** <http://127.0.0.1:9910/v1/statement>

**Sample request:**

{

"date": "18/12/2024"

}

**Sample response:**

{  
  "message": "Successful",  
  "transactions": [  
    {  
      "amount": "0.01  KES",  
      "auth\_code": "762924",  
      "code": "00",  
      "message": "Succesfully",  
      "receipt\_no": "5",  
      "rrn": "500813945774",  
      "status": "success",  
      "tid": "00337470",  
      "tran\_date": "08/01/25 12:30:43"  
    },  
    {  
      "amount": "0.01  KES",  
      "auth\_code": "282924",  
      "code": "00",  
      "message": "Succesfully",  
      "receipt\_no": "5",  
      "rrn": "500814138181",  
      "status": "success",  
      "tid": "00337470",  
      "tran\_date": "08/01/25 12:38:01"  
    },  
    {  
      "amount": "0.01  KES",  
      "auth\_code": "763629",  
      "code": "00",  
      "message": "Succesfully",  
      "receipt\_no": "5",  
      "rrn": "500813946713",  
      "status": "success",  
      "tid": "00337470",  
      "tran\_date": "08/01/25 12:38:20"  
    }  
  ]  
}

## **RECONCILIATIOn**

This is a functionality used in reconciling transactions done within a particular period to determine whether those in merchant POS marches with whatever is recorded by the pin pad.

The default reconciliation done is for the current day while those of past days can also be done.

The output is all the discrepancies in the merchant file and those in the POS.

**Endpoint:** <http://127.0.0.1:9910/v1/reconcile>

**Sample response:**

{  
  "discrepancies": [  
    {  
      "bank\_amount": "0.01",  
      "customer\_amount": "1",  
      "narration": "Amount mismatch",  
      "rrn": "500814138181"  
    },  
    {  
      "bank\_amount": "0.01",  
      "customer\_amount": "1",  
      "narration": "Amount mismatch",  
      "rrn": "500813946713"  
    },  
    {  
      "bank\_amount": "0.01",  
      "customer\_amount": "1",  
      "narration": "Amount mismatch",  
      "rrn": "500813945774"  
    }  
  ]  
}