

**Version 1.1**

**Date:**

Abstract

**The objective of this document is to describe the Point of Sales requirement as implemented at Engineparts. It covers the receipt of payment from the public in various payment methods and to multiple accounts,**

**Customer experience is addressed using a public announcement system described herein**

Point of Sale

*Cashiering & Banking*

**Table of Contents**

1 Document approval and distribution list 2

2 Introduction 3

3 Audience 3

4 Objectives 4

5 Functional / Operational Flow 5

6 Cashier shift management 8

7 Cashier payment receipt 10

8 Cashier shift closing and banking 12

9 Dependencies 13

10 Design philosophy and implementation 14

11 Database entities and relationships 15

12 Programs 16

12.1 MS Windows Executables 16

12.2 SQL Stored Procedures 16

13 Acceptance 17

# Document approval and distribution list

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Name / Title** | **Signature** | **Date** |
| **Document Type / purpose** | | | |
| Prepared by |  |  |  |
| Reviewed by |  |  |  |
| Approved by |  |  |  |

# Introduction

The organisation provides a walk-in sales service to formal and informal customers with the expectation that informal customer will pay for purchased goods immediately.

Incorporated is the need to allow for the value of returned goods to either pay for replacement parts and / or the return of cash to the customer.

Cashiers are entrusted with cash thus the need to provide cashier transactional detail to support reconciliation and banking per cashier shift.

For an enhanced customer experience, there is an automated annunciation system integrated into ePart that can automatically call customers to the cashier booth as soon as the picked goods are available for collection

# Audience

Sales

Financial department

Accounts receivable

Goods returned

Technical support staff

# Objectives

To optimise the sales process servicing walk-in customers named as the ***Sales Counter.***

Ensure that the cashier shifts reconcile and that the related bank deposits are done according to good governance and prescribed policies and procedures.

Allow for multiple payment methods i.e. cash, cheque, goods returned credit etc.

Incorporate a cashier float of monies required to return ***change*** to customers

Provide for cashier login and to open a new shift.

To support the cashier visually with displayed data in support the cashiering transaction. By way of example:

* Invoice amount to be paid
* Payment methods allowed
* Change to be returned to the customer
* Ability to call up returned goods and credit amount allowed in leu of payment

Ability for a cashier to close a shift for reconciliation. This state will allow the cashier to re-open the shift to make corrections if need be.

Allow cashiers to have multiple ***open*** shifts of which one is active and the others pending reconciliation.

Ability to allow for a cashier shift to be finalised and closed permanently

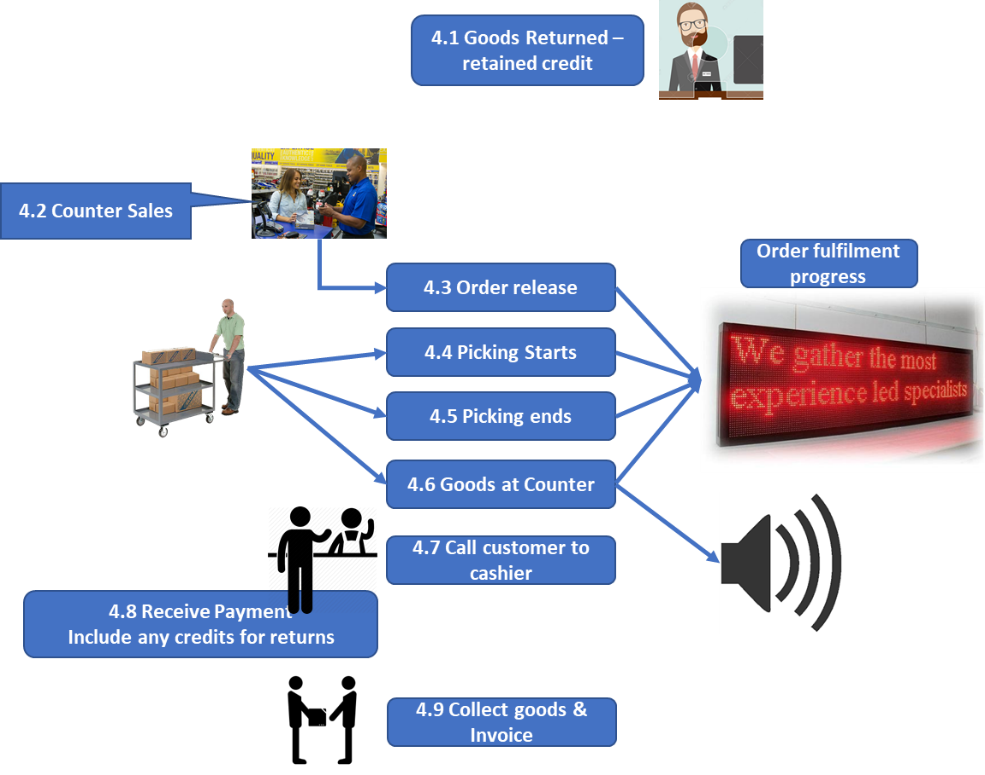
Print banking deposit schedule including coinage analysis

GL & bank integration

Integrate into the picking, pack and invoicing processing to trigger cashier annunciation functionality

# Functional / Operational Flow

The following diagram is a very high-level presentation of the activities and related aspect of shift related cashiering with order fulfilment to customer requests.

**Diagram 5.1 provides a high-level view of the specific activity**

The diagram detail is as follows:

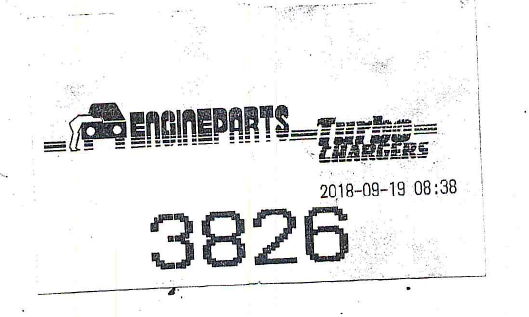
* 1. Goods returned by customers receive a ***pending*** credit amount that is either paid in full to the customer or can be applied against a further purchase.

Should the customer wish to receive the full credit, the supporting documentation as received from the **Goods Returned office**, is presented to the cashier and the cash amount refunded.

* 1. The customer, with the assistance of a counter sales person, makes further purchases and concludes the transaction by releasing the sales order to picking services.

A ticket slip is printed with the last 4 digits of the sales order reference and given to the customer that will be used by the system to call the customer to the cashier booth.

***Image 4.2.1* is a sample of a printed ticket**



* 1. The released sales order status is updated in the database to a status of ready for picking and prioritised at a higher priority as per the picking sub-system configuration where specific routes can be elevated or demoted in a priority scheme relative to route / vehicle departure time.

This is described in more detail in the ***Picking*** documentation

* 1. Pickers announce their availability to accept the next ***prioritised picking job***. This could release to the picker 1 or more sales order(s) depending on the associated ***route(s)*** the

picker is assigned to.

To ensure that the counter sales receive maximum picking prioritisation, a picker will receive a (configurable) limited number of sales orders to fulfil.

The moment a sales order is assigned to a picker, the clock starts running that measures the time to complete. Currently this time footprint is used in management reports and not displayed on the public display as per the diagram. However, it is a planned for enhancement.

* 1. The picker then drops off the goods at the checking counter and confirms the drop-off by scanning the barcode on the picking job sheet.

To note: the system is configured to direct the sales order picking to the counter sales location

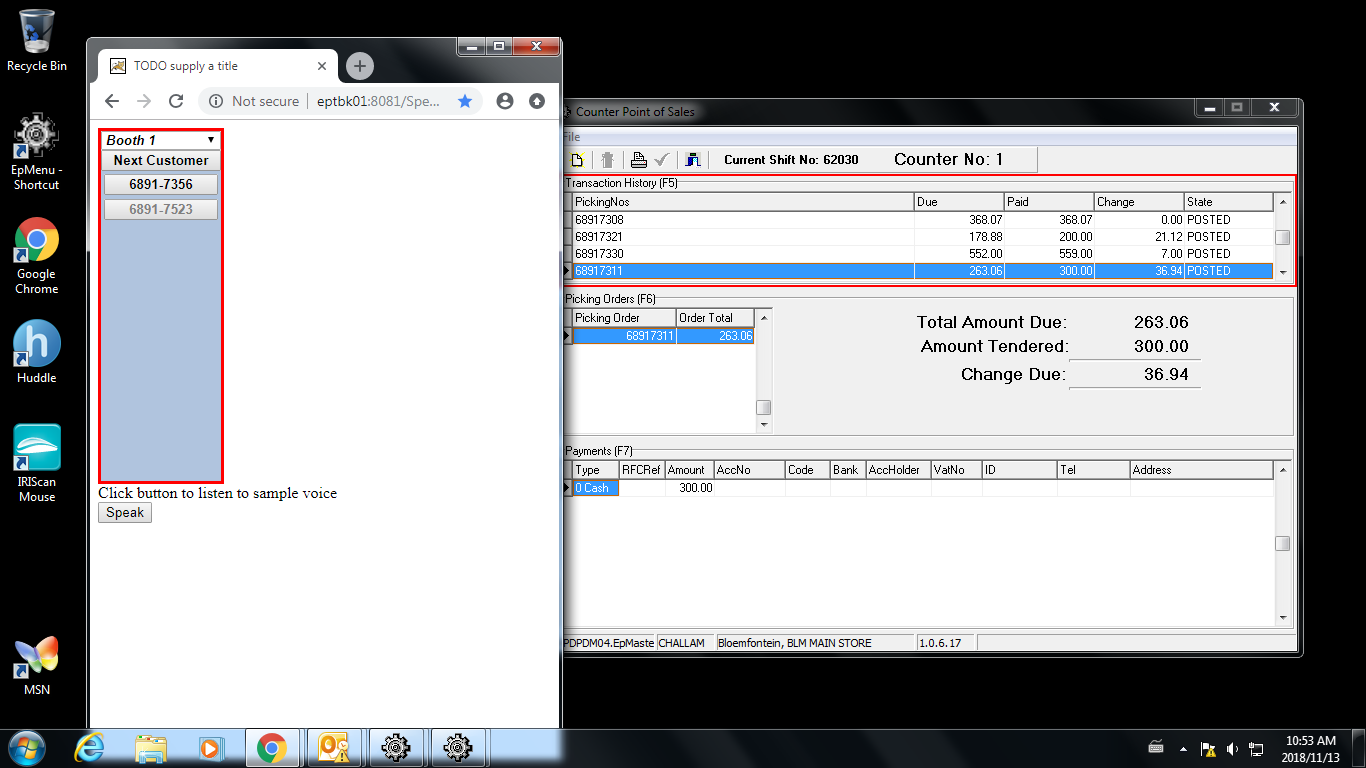
This stops the picker timer and starts the ***checker*** timer. This statistic is used by management. The display on the public display board is a planned for enhancement

* 1. When the checking staff complete their verification of the picked goods and find all OK, the picking sheet is scanned to confirm the status.

This triggers a cashier event and the sales order appears on the list of sales orders ready to be paid for.

This view is presented via a web browser with an Apache Tomcat application (JEE) server supporting the functions in the backend.

***Diagram 4.6.1*** **provides a cashier view of the list of sales orders that are ready to be paid for by waiting customers**



As depicted, there is a list on a web page, of sales orders ready to receive payment. The cashier must click the relevant reference which initiated an announcement on PC connected speakers and displays the reference on a digital LED board.

There is an algorithm which functions as follows:

* The oldest sales order ready for payment is presented at the top of the list
* After 3 customer announcements without response de-lists the priority assigned to the sales order, which then drops the sales order to the bottom of the list
* The next oldest sales order is promoted to the top.
* As soon as the Sales Order is paid – cashier program is depicted above in the same image – the Sales order status changes to a completed state in the database and the web page updated automatically by removing the Sales order form the list.
* The web page depicted is auto refreshed from the server side only when a status change is detected by the server application
  1. Multiple cashiers receive the same web-based view of sales order ready for cashiering. Each cashier that is available to receive a customer, uses a mouse to click on the on-screen ***Button***. This action launches a voice simulator that translates a text message into high quality voice simulation.

The text message ***includes the last 4 digits of the original sales order*** for which a waiting customer would have received a ticket with the same number printed.

Included in the text message that is voice simulated is the cashier booth number. It is expected of the customer move to the cashier booth number and hand over the ticket to verify the sales order that the customer is expecting to pay.

Should the customer not respond, the selected reference to announce will cycle through 3 iterations and then drop down to the last entry on the list. Over time and as other entries are removed from the list the entry will move to the top again and the calling process repeated.

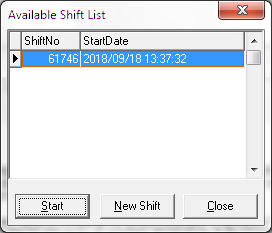
Entries not paid for on the day is assumed to be cancelled and will not be presented the following day. However, this cancellation is not automatic. It is expected that operational staff will cancel the outstanding orders before start of business the next day.

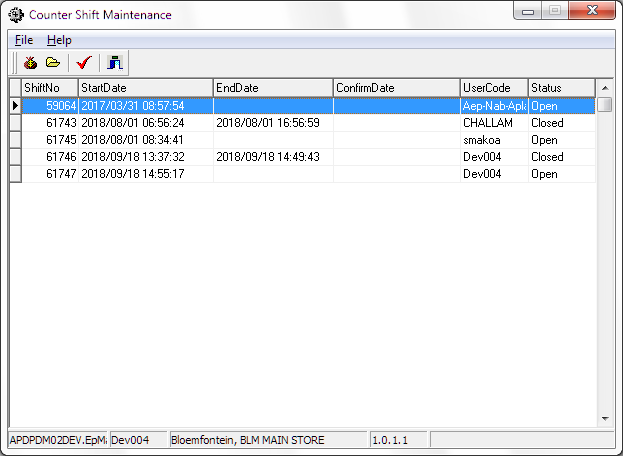
The cashier calls up the sales order to be paid using the ticket number. This action also removes the reference off of the list of sales order to be paid.

The sales order reference displayed on the web page. (Page image to be supplied)

# Cashier shift management

The cashier is required to start a new shift or continue with one of the past created shifts. Importantly, finalised shift will not reflect on the list of available cashier shifts.

**Diagram 6.1 provides a view of the cashier shift user interface and shift supervisor view:**

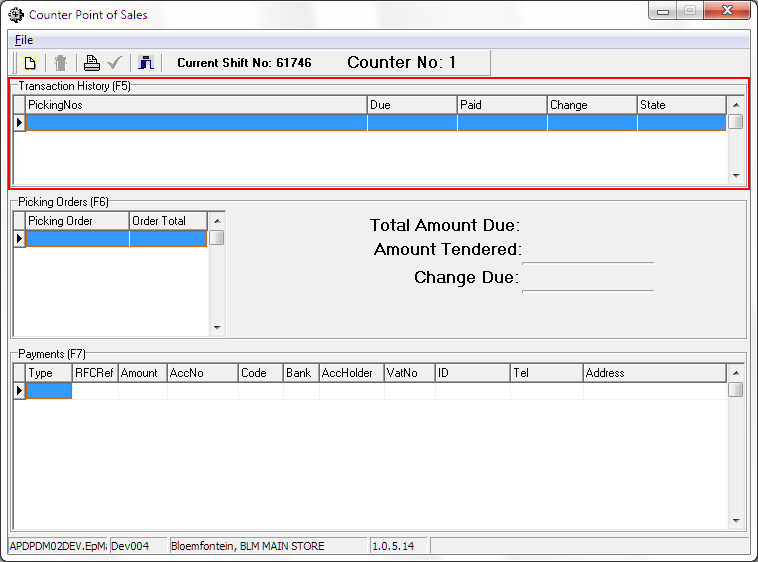


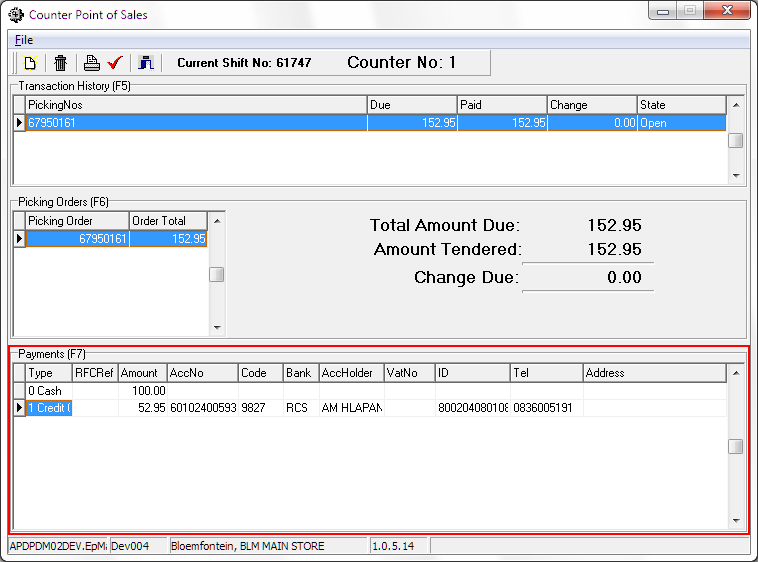
# Cashier payment receipt

The ePart POS intrinsically has a number of transactional perspectives based on, especially the payment methods. The following set of screen views represent a sample of the overall functionality and should be enhanced over time

**Diagram set 7.1 provides a view of how the cashier program progresses through some of the screen views**

This is a view of the cashier program ready for business



This view represents Sales Order to be paid and has been identified by the last 4 digits of the ticket issued to the customer at the sales counter

* + 1. The cashier advises the customer of the amount to be paid
    2. The customer is expected to present the cashier with any of the following combinations of payment methods:
  + Cash
  + Credit card (one or more)
  + Cheque (one or more)
  + Return for credit (RFC) documents (one or more). To note – the RFC documents are verified, and the credit amount reflected automatically.

After concluding the transaction, the RFC reference cannot be presented as payment again.

* + 1. On conclusion of the payment transaction, the sales order is released for invoice printing and the customer moves to the good collection and invoicing counter.
  1. The customer presents the collection counter clerk with the sales order reference, the clerk collects the goods from the staging shelf, scans the document barcode and the invoice is printed.

The goods with the invoice is handed to the collecting customer

# Cashier shift closing and banking

The cashier is fully accountable for managing the receipting process and is tasked with ensuring the accuracy of cash and payment collections.

The application supports this notion applying best practice from a usability and operator guided step by step processing.

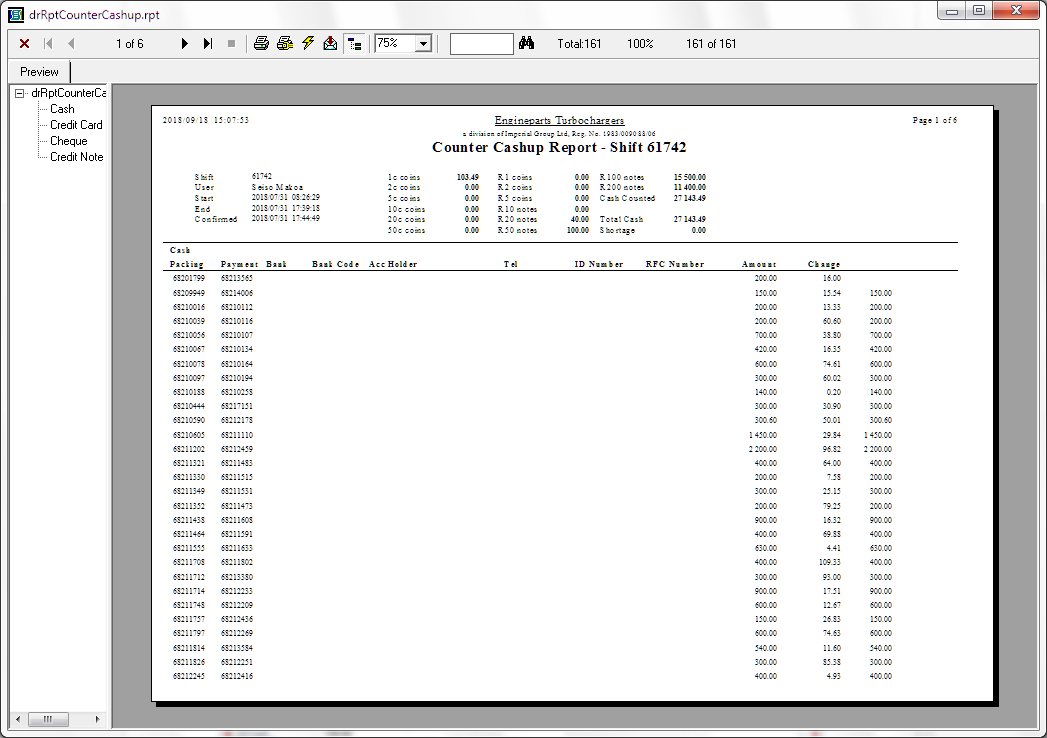
Cashiers are required to open a cashier shift under their assigned ePart user id and password.

Should the cashier be issued with a ***float*** amount (used for cash and change to customers) which the cashier signs receipt for. At cashier shift final close, the cashier is to return the float amount issued ***minus the amount returned to customers with RFC’s.*** This will only be evident if the cash collected during the shift was less than paid out to customer RFC’s.

The shift close report reflects the transaction audit trail and the monies received by the cashier. The physical payments received are reconciled and compared to the shift close report.

***Given that the reconciliation balances*** then the cashier and supervisor prepare a bank deposit, inclusive of a coinage analysis as dictated by the bank.

**Diagram 8.1 provides a view of the cashier cash-up report form which the banking detail is created**



***Should there be a surplus*** (Cashier collected more than required), the surplus is banked or retained subject to a defined procedure in case a customer query.

***Should there be a shortfall*** (Cashier collected less than required), the cashier is required to make up the shortfall. However, subject to a policy and approval, the shortfall can be made up from the surplus fund.

The banking is to be captured as per expectation for monies received. Shortfall or surplus is to be reflected in the actual banking which will result in a variance in the Nominal Ledger. The variance will require a manager’s journal entry to ensure recording of the non-conformance position

# Dependencies

|  |  |  |
| --- | --- | --- |
| # | Description | Action / By whom |
| 1 | Accounts receivable |  |
| 2 | Sales order |  |
| 3 | Picking |  |
| 4 | Invoicing |  |
| 5 | Warehouse activities |  |
| 6 | Cashbook |  |
| 7 | GL - bank |  |

There is a hardware and software dependency outside of the standard ePart framework which has come about due to the ***age*** of Borland C++ builder.

The further intent was to introduce a modern technological framework for informative reasons

The dependencies are:

* Hardware display panel (LED based with 4 lines of display – no images) that is interacted with using RS232
* Cable running from display to server room with a USB to RS232 converter.
* Tomcat web server which is part of the ePart solution on which the display panel data is presented.
* Browser deployment using Chrome that also communicates with the Tomcat server

There is a reserve display panel and USB / RS232 converter in case of failure

The way of application design i.e. all server-side logic, the functionality can easily be migrated from old style display panel to a large TV display unit connected to a local PC in shared mode. This was reviewed with management and acceptance received for the proposed implementation on the basis that greatly improved customer interaction and experience can be had. The task will require prioritisation.

# Design philosophy and implementation

The cashiering solution is developed using C++ Builder on the presentation side (15%) whilst the business logic (85%) is written using MSSQL stored procedures. The implementation model is described under the relevant documentation set dealing with the developer environment.

On the customer progress display and electronic voice simulation the implementation is as follows:

* + The developer implementation aims at developing (server side) business logic using Java
  + The server accesses the digital display panel using USB to RS232 converter, displaying textual data inline with cashier web-based selections aimed at announcing to the waiting customer which sales order reference is required to be at which specific booth number.
  + The persistence implementation is established using JPA (Java persistence architecture) allowing for true data independence. Currently the interface is for MSSQL using Stored Procedures with a limited set of JPA calls
  + The specific Java JEE server deployed (many others are available as well) Apache Tomcat
  + The presentation is implemented using HTML5 / java-script
  + The browser (Chrome) capability to translate text to speech is exploited. The text to speech is transmitted (transported) from the server side using WebSockets in JSON format
  + The communications between browser and server uses HTML5 defined WebSockets aimed at mimicking fat client way of working.

The underlying benefit to this approach is simplicity and maintainability.

The obvious notion to this is that the presentation logic can theoretically be swapped out retaining the business and persistence logic intact.

This has largely proven to be the case when looking at the web-based functionality incorporated herein.

# Database entities and relationships

# Programs

# MS Windows Executables

|  |  |
| --- | --- |
| **Name** | **Description** |
| drCounterPoS.exe | The main counter cashiering application; used to capture payment information. |
| drCounterShift.exe | Is used to verify cash-up and post the verified shift to the cash book. |
| drCounterRFCApprove.exe | When an RFC document is used in lieu of payment, but the RFC is older than 14 days, the sales manager must use this application to approve its use. |

# SQL Stored Procedures

|  |  |
| --- | --- |
| **Name** | **Description** |
| drCounterPoSValidateOrderNumber | Validates the order number before it can be added to the payment. |
| drCounterPoSGetPrintParams | Gets the invoice number and number of copies to print. |
| drCounterShiftClose | Verifies that the shift is ready to be cashed up and marks it as such. No further payments can be added. |
| drCounterNew | Creates a new payment group on the shift. |
| drCounterNewShift | Creates a new shift for the current user. |
| drCounterDelete | Deletes a payment group and all the data in it. |
| drCounterRefresh | Calculates the change and related totals of the current payment group. |
| drCounterRFCDet | Verifies an RFC number and gets the RFC value. |
| drCounterPost | Posts the payment group, indicating that the customer has finished paying and has been given their change. |
| drCounterOrderAdd | Adds a sales order or picking job to the current payment group. |
| drCounterShiftPost | Posts a checked shift to the cash book, indicating that the cash can now be deposited. |
| drCounterShiftOpen | Re-opens a shift so that mistakes can be fixed. |
| drCounterRFCApproveAdd | Adds an RFC document to the approved list. |

# Acceptance

I hereby confirm that I have been fully informed of the documents content and, received training to understand how the detailed instructions are to be applied

Name …………………………………………………………………………….

Job Title ………………………………………………………………………….

Signed ……………………………………………………………………………

Date ………………………………………………………………………………