

**Version 1.1**

**Date: 2018/04/09**

Abstract

**The objective of this document is to describe the application that is used to create and specify sales orders.**

Sales Order

*Creation and Maintenance of Sales Orders*

**Table of Contents**

Document approval and distribution list 2

1. Introduction 3

2. Audience 3

3. Objectives 3

4. Business Flow 3

5. Detailed description of functionality 3

6. Dependencies 4

7. Application design philosophy 4

8. Database design philosophy 4

9. Catalogue lookup to sales-order 4

10. Database entities and relationships 6

11. Programs 8

11.1 MS Windows Executables 8

11.2 SQL Stored Procedures 8

12. Acceptance 10

# Document approval and distribution list

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

# Introduction

While the catalogue lookup application is used to determine what items the customer is willing to purchase, the sales order application is used to capture all the additional detail, such as the delivery address, and also to override prices so as to make the purchase more enticing.

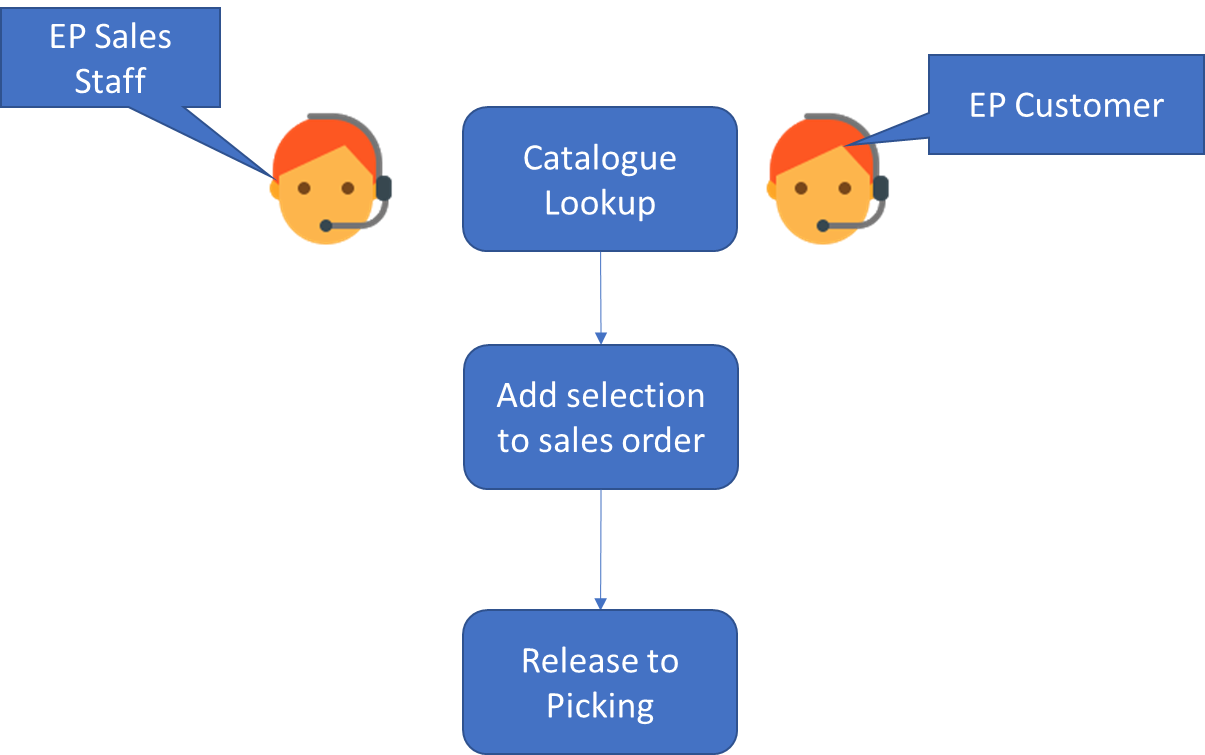
# Audience

* Salespeople
* Programmers

# Objectives

Like the catalogue lookup application, the sales order application was designed for deployment across public networks, so that qualified customers may order products directly from multiple warehouse locations.

# Business Flow

The following diagram is a very primitive presentation of the bigger business / operational flow and needs to be expanded on:

# Detailed description of functionality

**<This section still needs a lot of work done>**

System administrators provide qualified end users with a user identity and a temporary password.

When logging on for the first time, a better password is requested from the user.

In the case of on-line sales, the user code must be identical to their debtors account code, and all their sales orders will be limited to that one account code.

* 1. Credit limit   
     Additional sales will be blocked should this limit be exceeded. However, senior management are permitted to override this temporarily.
  2. Payment terms (30 days, 60 days etc.) and settlement discounts
  3. A discount matrix that the system uses to calculate selling prices
  4. Essential customer detail
  5. Delivery addresses
  6. Contact details

# Dependencies

* Accounts Receivable
* Pricing
* Cataloguing
* Warehousing
* Master Data from Sage X3

# Application design philosophy

The same design philosophy is used as in most other ePART applications. It consists of 3 basic components:

* 1. Presentation – A C++Builder application with limited, if any, business logic.
  2. Business logic – Implemented as stored procedures on an MSSQL database server.
  3. Data persistence – The MSSQL database server is responsible for persistence.

This modular approach should increase the longevity of the product by allowing developers to replace the business logic and presentation independently of one another.

# Database design philosophy

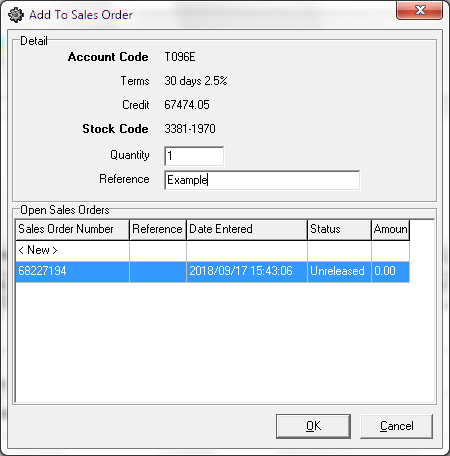
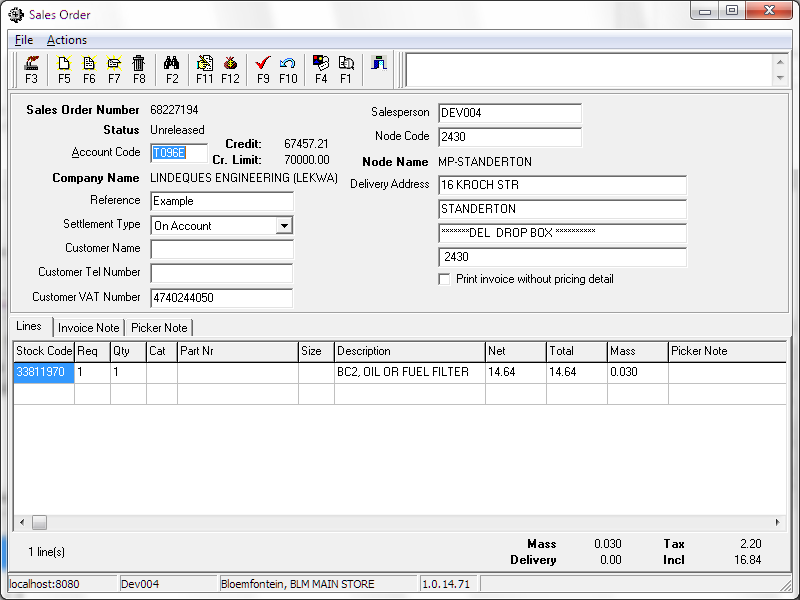
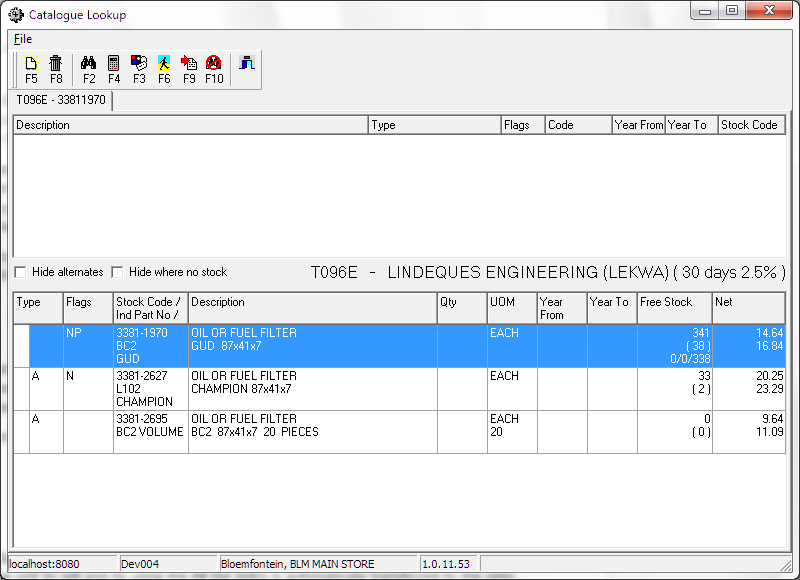
The sales order application uses an object oriented approach to the data and a three tier approach to communicate with the database server.

**<This section will be replaced with a link to the database design philosophy document>**

# Database entities and relationships

# Catalogue lookup to sales-order

The catalogue lookup application operates independently from the sales order program. However, there is a path of communication from the catalogue lookup to the sales order as depicted in the following diagram:



Once the decision has been made as to which item to sell, pressing the ***F9*** key (or clicking this button) is a shortcut to automatically place the item onto the sales order.

# Programs

# MS Windows Executables

|  |  |
| --- | --- |
| **Name** | **Description** |
| salOrder.exe | Used to capture the customer’s details on the sales order and release it to picking. |

# SQL Stored Procedures

|  |  |
| --- | --- |
| **Name** | **Description** |
| salSalesOrderLineRefresh | Returns all fields of a sales order line. |
| salSalesOrderLineFieldChange | Verifies, applies and cascades changes to a sales order line. |
| salSalesOrderLineHomeCreate | Adds a line to a sales order. |
| salSalesOrderLineHomeDelete | Removes a line from a sales order. |
| salSalesOrderLineHomeFindByCtrlRef | Finds all the lines of a sales order. |
| salSalesOrderRefresh | Returns all fields of a sales order header. |
| salSalesOrderFieldChange | Verifies, applies and cascades changes to a sales order header. |
| salSalesOrderGetTotals | Calculate the total sales price and VAT of a sales order. |
| salSalesOrderRelease | Verifies whether the sales order can be released, and splits it into multiple orders if it has more than 10 lines. Then passes them on to salSalesOrderReleasePhase2 and salTranCopyToStkTran. |
| salSalesOrderReleasePhase2 | Does further verification, prepares the picking job details and then marks the sales order as ‘released’, to indicate that it is ready for picking. |
| salTranCopyToStkTran | Copies the delivery address and customer VAT number to stkDeliverDet, where the invoicing expects to find it. |
| salSalesOrderRecall | If the order is not yet on a picking job, reverses the release. In the case where the order was split, this only recalls one part. |
| salSalesOrderHomeCreate | Creates a new sales order. |
| salSalesOrderHomeDelete | Deletes a sales order and all its lines. |
| salSalesOrderFindOpenOrders | Finds sales orders that have not yet been released and either have less than 10 lines or are older than 25 minutes. |
| salOrderSearchAccCode | Finds a debtor account code by the account’s name. |
| salOrderSearchSalesperson | Searches by name among sales people to whose orders the current user has access. |
| salOrderSearchNodeCode | Finds a node by its name. |
| drDebtorRefresh | Looks up a debtor account. |
| drDebtorAddressFindByAccCodeAndType | Finds of a debtor’s addresses of one type. In this case used to find all the delivery addresses. |
| drDebtorAddressRefresh | Looks up the details of each address returned by the search. |
| catBaseItemRefresh | Looks up a base item. Used here to determine whether an item’s cost price can be overridden. |

# 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 ………………………………………………………………………………