

**Version 1.1**

**Date: 2018/04/09**

Abstract

**The objective of this document is to describe the Catalogue Lookup and decision support to staff participating in the sales processes**

Catalogue Lookup

*Decision Support to All Stock Related Aspects*

**Table of Contents**

Document approval and distribution list 2

1. Introduction 3

2. Audience 3

3. Objectives 4

4. Dependencies 4

5. Risks and mitigation 4

6. Requirements overview 4

7. Acceptance 6

# Document approval and distribution list

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

# Introduction

Stock assurance is a requirement, especially at year end where the Organisations stock is to checked for presence.

# Audience

Sales

Purchasing

# Objectives

To optimise the sales process through supporting information exposed against filtering criteria.

Experienced users are able to provide key search terms that expose directly the information being looked for

However, the lookup process allows less experienced users to provide search terms that are less cryptic

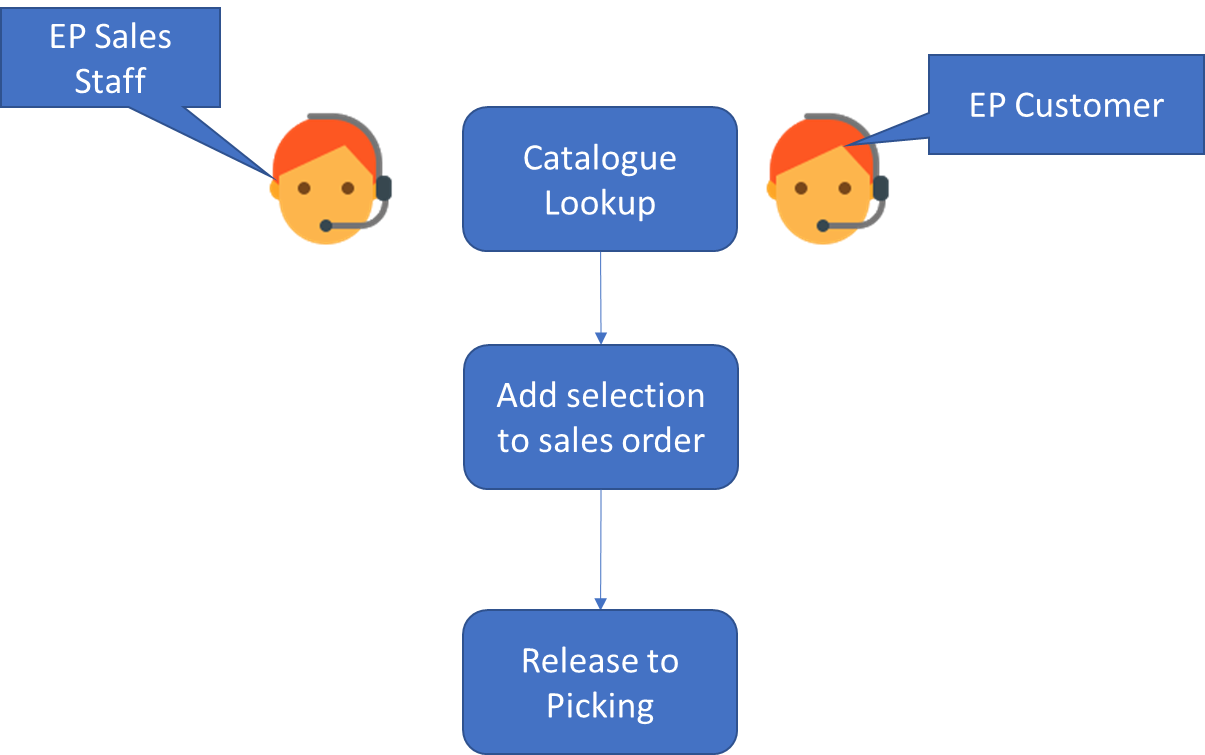
For those with very little experience can use the ***tree search*** method to navigate descending wit ha mouse to derive the information requested by a customer

The catalogue lookup application was designed for deployment across public networks such that qualified customers are able to order product from Engineparts directly and 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

**Diagram 4.1 provides a high-level view of the specific activity**



# Detail description of functionality

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

On logging on the user with a temporary password is required to change it using a character combination that is not overly simplistic and easily hacked.

The user code is aligned with an account receivable (debtors) account with:

* 1. A trading credit limit.

Additional sales will be blocked should this limit be exceeded. However, senior management are permitted to override this temporarily.

The process with Sage (X3) ***describe the X3 processes such that a technical person can interpret***

* 1. A trading term (30, 60 days etc).
  2. A discount code that the system uses to calculate a selling price.
  3. Essential customer detail is exposed

Delivery address

Special actions such as ***Must Provide Purchasing Reference.***

Telephone

Contact details

***The discount code methodology is described in PROVIDE A LINK TO THE RELEVANT DOCUMENT***

# Dependencies

|  |  |  |
| --- | --- | --- |
| # | Description | Action / By whom |
| 1 | Accounts receivable |  |
| 2 | Pricing |  |
| 3 | Catalogue maintenance |  |
| 4 | Discount structures |  |
| 5 | Warehouse activities |  |
| 6 | Sales order |  |

# Design philosophy

The design philosophy closely adheres to the manner in which the ePart was incarnated by splitting the entire application into 3 basic components:

* 1. Presentation – this is done in Builder c++ with limited if any engagement of business logic
  2. Business logic – this is done using MSSQL stored procedures
  3. Data persistence – the fact that business logic is contained in Stored Procedures makes the persistence integral part of the business logic process.

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.

# Database design approach

Essentially the industry that Engnieparts participates in has a very strong engineering participation where assembly, sub-assembly and finite parts are strongly represented.

Consequently, ePart catalogue follows this discipline in its relationship design structure.as a “inverted tree” construct.

**Diagram 8.1 provides a high-level view of the *core* catalogue Entity Relationship as deployed**



This ***self-referencing*** relationship provides for unrestricted levels of relationships to be assembled in an ***assembly*** to ***Sub-Assembly*** without limits. The ***Part(s)*** relationship provides the ***item level component stocking attributes*** as depicted in the following diagram:

**Diagram 8.2 provides a view of how the relationship is used to depict a easily viewed construct**

The assembly / sub-assembly explosion indicates how elements with related attributes can be viewed with the highlighted entry indicating the actual stock item.

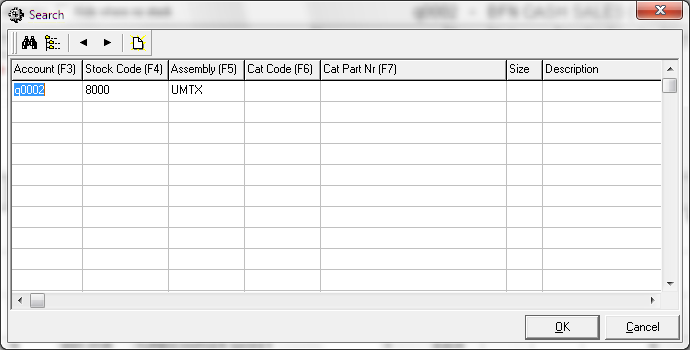
By design the solution provides for the linking of assemblies to stocking items even when the stock item is made up of sub-assemblies i.e. a turbo charger can be sold as a complete unit yet some parts making up the turbo charger can be sold separately.

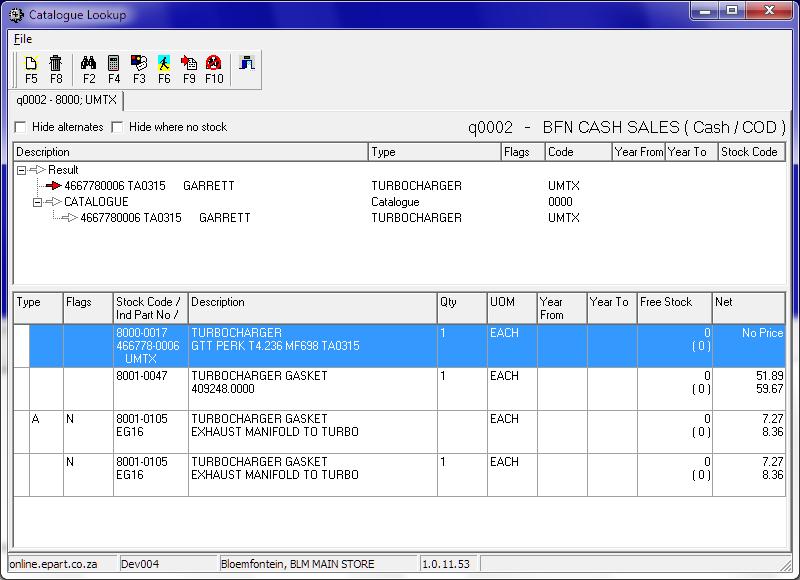
***Note that there are more control attributes such as assembly type, code and others. Refer to the full entity relationship chart for more details***

# The advanced searching approaches

In addition to the parts explosion (inverted tree) format of searching, there is the advanced searching capability and is depicted in the following diagram:

**Diagram 9.1: A high-level view of advanced search capabilities:**



**Diagram 9.2: The search results:**

Some of the search criteria can expose the concept of ***KITS*** being for servicing or engine overhaul and several others as well. This allows for opportunistic selling by the sales staff enhancing the customer experience.

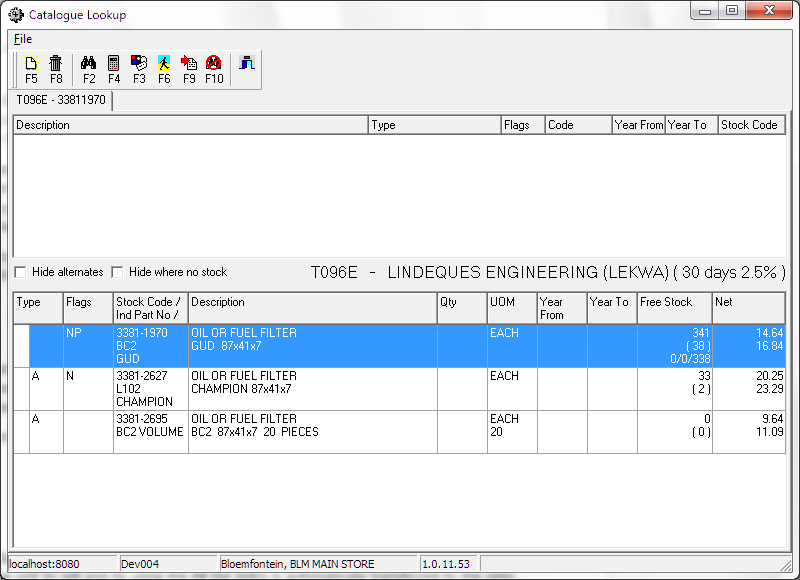
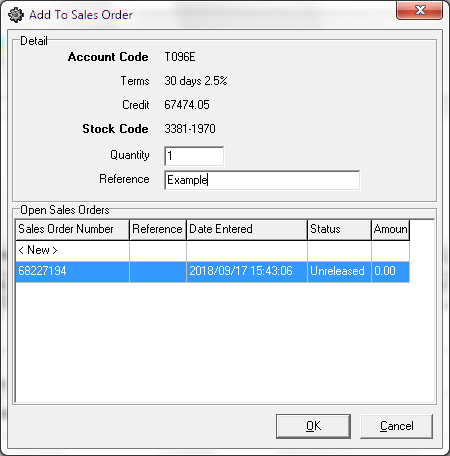
The catalogue lookup allows for the searching using industry part numbers or supplier part numbers for the same stocking unit. This is for instance a piston ring can have multiple industry part numbers depending on which manufacturer distributer is called off by the customer.

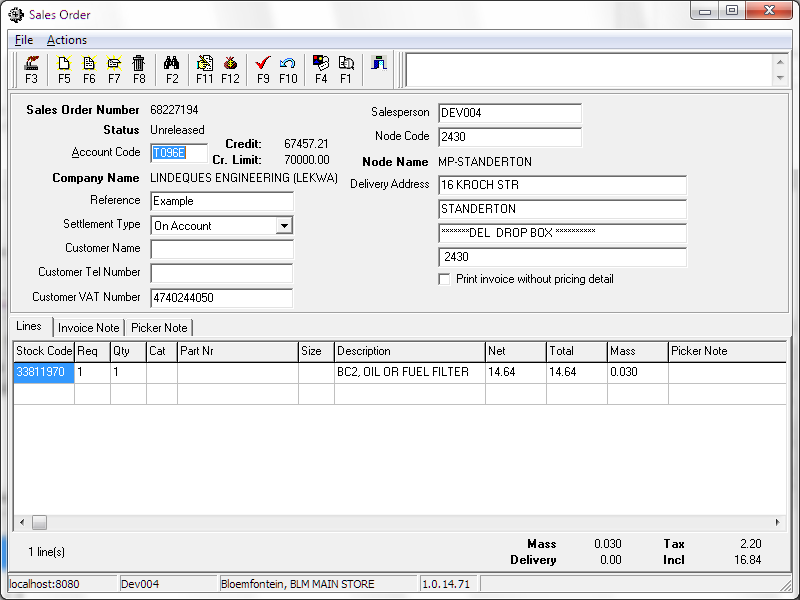
To enhance customer experience, the system allows for supersessions, alternates and discontinuations.

# 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

Diagram 10.2 diagram provides a view of how the lookup interacts with the sales order





The effective optimisation is that from the search and filter results set, a choice is made of the stocking unit to sell and by using the ***F9*** the entry is automatically transferred to the sales order.

# Database entities and relationships

The full catalogue related database diagram provides a view of the various participating relationships. Notably is the simplicity of the database tables participating in the core structure.

Many of the additional tables are there for control and optimisation purposes.

A specific reference is made to the following tables:

* “Bridging table” – this is often used in data mining structures where the volume of data requires a specialised table of this kind
* BOM level – this table is a control table to prevent illogical parent child linking i.e. linking an engine to a water-pump rather than linking a water-pump to an engine
* ….
* ….
* ….
* …

The catalogue support system provides for a highly efficient source of information, provided the underlying data is accurately maintained and kept up to date; failing which will result in poor customer experience.

# Programs

# MS Windows Executables

|  |  |
| --- | --- |
| **Name** | **Description** |
| catLookup2.exe | Used to look up which parts the customer wants to purchase, and add them to the sales order. |
| salOrder.exe | Used to capture the customer’s details on the sales order and release it to picking. |

# SQL Stored Procedures

# catLookup2.exe

|  |  |
| --- | --- |
| **Name** | **Description** |
| catCatQueryResultHomeFindByQueryNo2 | Searches for items matching the criteria, to populate the item grid. |
| catCatQueryBomNodeHomeFindByQueryNo | Finds the section of the BoM above and below the search results. |
| salAddToSalesOrder | Automatically adds an item to an existing or new sales order. |
| catLogLostSaleManually | Records a formal lost sale, in the form of a failed-search-type informal lost sales marked as “manual”. |

# salOrder.exe

|  |  |
| --- | --- |
| **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 the phase 2 procedure. |
| salSalesOrderReleasePhase2 |  |
| 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 ………………………………………………………………………………