

**Version 1.1**

**Date: 2018/12/04**

Abstract

**The objective of this document is to describe the application that is used to address customer requirements for stock that Engineparts has not on hand.**

Sales Buyout

*Purchase goods not in stock*

**Table of Contents**

Document approval and distribution list 2

1. Introduction 3

2. Audience 3

3. Objectives 3

4. Business Flow 4

5. Detailed description of functionality 6

6. Dependencies 6

7. Application design philosophy 6

8. Database design philosophy 7

9. Database entities and relationships 8

10. Buy-out to sales-order 8

11. Programs 9

11.1 MS Windows Executables 9

11.2 SQL Stored Procedures 9

12. Acceptance 10

# Document approval and distribution list

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

# Introduction

Often customers need stock that is either not on hand or not a standard stocking item in Engineparts.

To maintain a positive customer experience, ePart has mechanisms to assist in procuring the customer required stock through a controlled process of buying the parts from one or more suppliers.

# Audience

* Sales Staff
* Purchasing
* Developer team

# Objectives

For efficiency, sales staff need to manage the buy-out process on-line in real time; otherwise the process would be very time consuming and error prone with lack of accountability.

A sales person, with customer interaction, identify parts that Engineparts would be unable to supply. For these items, the sales person can:

* Define the debtors account to bill.
* Define suppliers from whom to order the buy-out parts
* Define per supplier which parts are to be bought out

Per supplier a purchase order is created in ePart in anticipation of receiving goods from the supplier.

The option to send the purchase order to each supplier is available but the trend is to call the suppliers with the order details including the purchase order reference.

In the case where a sales person does a cash only buy-out, the sales order is assigned to the sales person buy-out debtors account. This is done to ease the effort to assign ***accountability*** for goods ordered but not sold / collected by the cash-only customer.

Management information is made available to the cataloguing team for analysis, generally around the following:

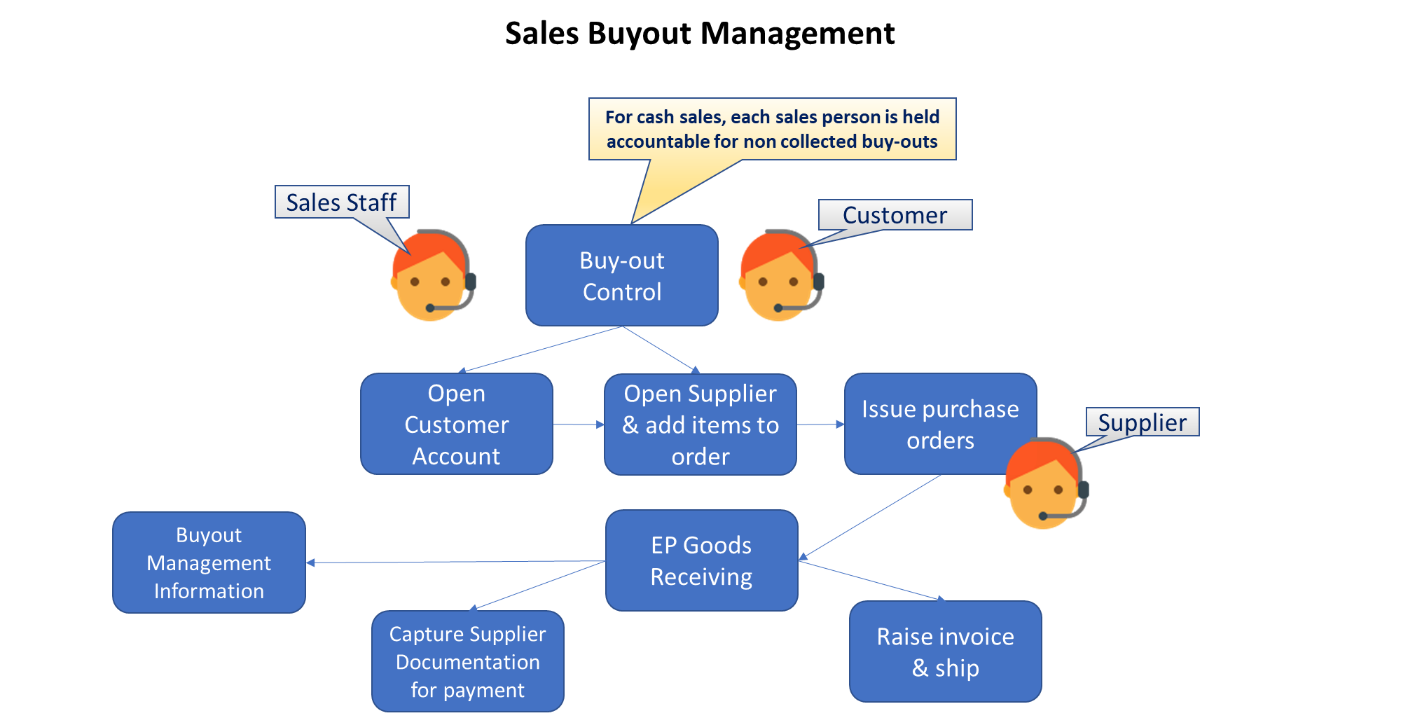
* Can buy-out items become stocking items supported by the sales history?
* Are items being bought-out but returned as incorrect due to identification errors on the ePart catalogue
* Is the reason for buy-out due to lack of stock? Should the supplier order pipeline be modified to ensure consistent availability?
* ……

# Business Flow

In support of the business objectives, the business requirements are to institute end to end control over the buy-out process:

* The customer is identified as an existing customer using an acceptable debtors account code.
* As part of the process, the standard credit limit checks etc are applied before the buy-out can be commenced
* The items to be bought out are associated against an approved supplier with whom Engineparts has an agreement if bought on account.
* Should the supplier be cash-only to Engineparts, the sales person needs to arrange for payment with the Engineparts accounting department.
* Engineparts receiving on receipt of the supplier goods apply the standard receiving disciplines to ensure that the goods received are the goods ordered by item, quality and quantity.
* Wrong parts are returned
* Over supply is returned
* Short supply advisories are notified
* Goods accepted as OK are binned according to the buy-out binning rules
* Sales order(s) released are passed on to picking for customer fulfilment
* Picking rules apply that ensure that the buy-out goods are associated with appropriate sales order and once picked and check, the relevant invoicing done, followed by despatching and delivery.
* Should the receiving customer decline the goods bought-out there is a business decision that needs to be adhered to:
  + Parts bought-out may not be a product that Engineparts would naturally stock thus not to be brought into the formal stock holding
  + If not part of the normal stock holding an attempt must be made to return to supplier.
  + If the supplier is not will to honour the returns request, the goods are sent back to the Engineparts customer as non-returnable.
  + Otherwise, if affordable, Engineparts could decide to provide a RFC to the customer and book it into stock as a non-standard purchase.

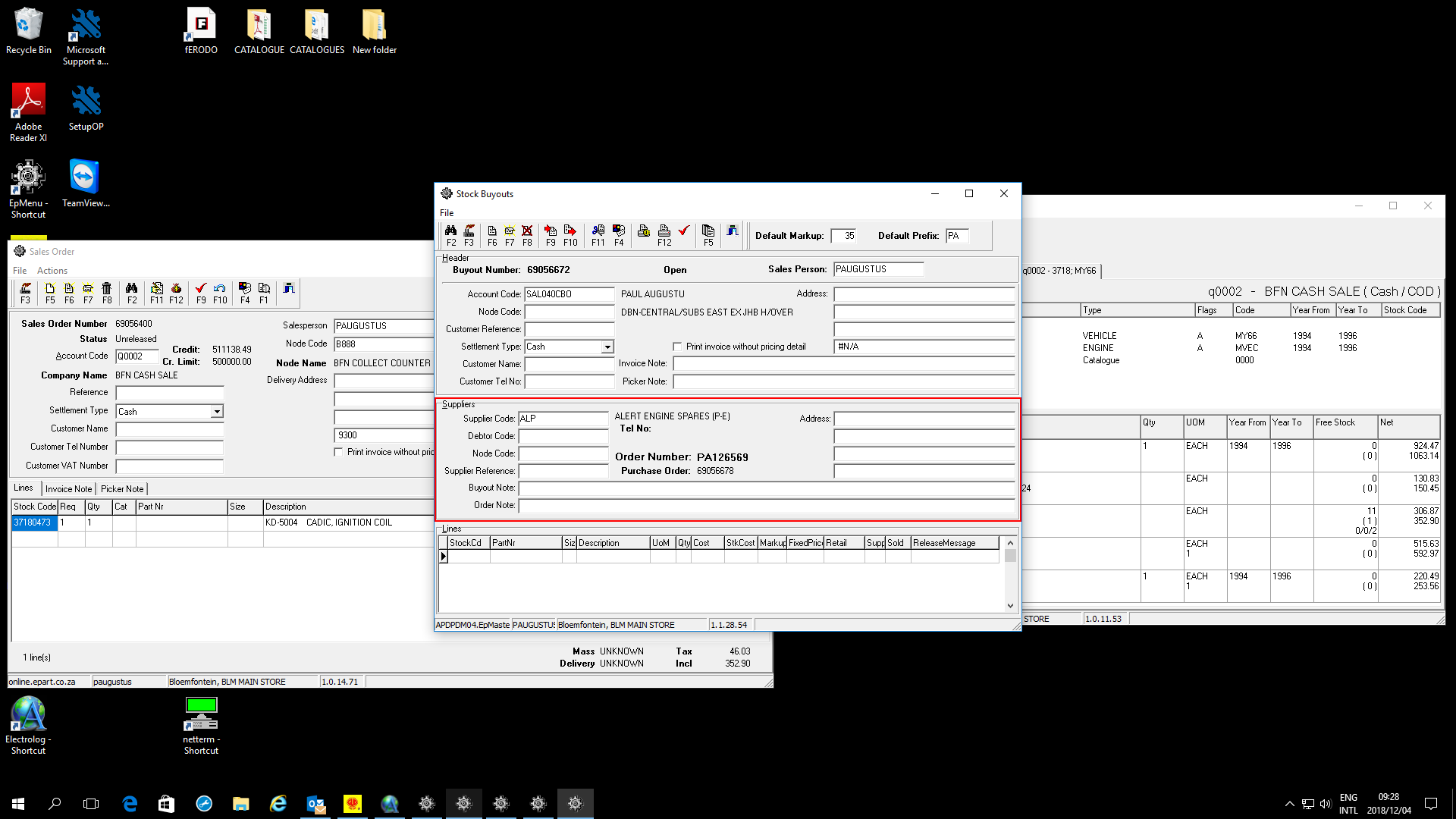
**Diagram 4.1 is a high-level presentation of the bigger business / operational flow and needs to be expanded on:**



The following diagram is a high-level presentation of the bigger business / operational flow and needs to be

# Detailed description of functionality

The following illustration provides a view of the sales person display unit with the buyout application view in the foreground



This function is specifically useful in that sales staff are able to maintain a good relationship with customers by providing a service for items either not stocked by Engineparts or where Engineparts might be sold-out.

Beneficially, buy-outs are reviewed by the purchasing department to determine if the buy-out product should become a stocking item or to chase suppliers for delivery of new stock.

# Dependencies

* Accounts Receivable
* Accounts payable
* Pricing
* Cataloguing
* Warehousing
* Purchasing
* Master Data to / 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.

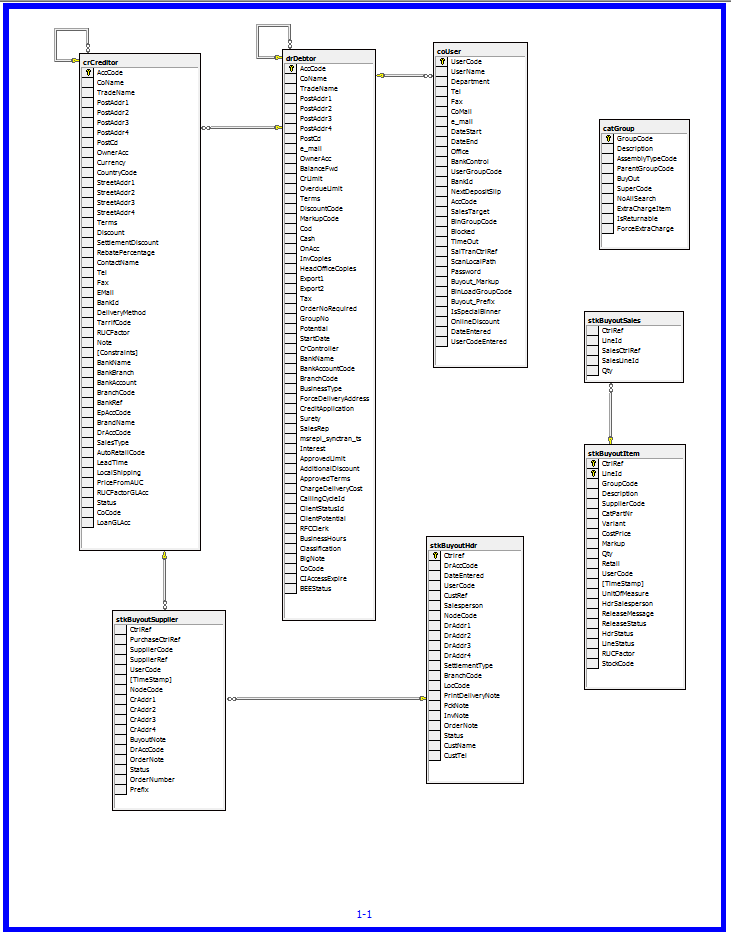
This modular approach should increase the longevity of the product by allowing developers to replace the business logic and presentation independently of one another.

# Database design philosophy

The buy-out application uses an object-oriented approach to the data and a three tier approach for presentation, business logic and database persistency.

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

# Database entities and relationships



# Buy-out to sales-order

The buy-out application operates in conjunction with sales order and purchase order programs.

The buy-out outcome generates the dependent sales order and purchase order(s) and can be accessed independently through the respective user interfaces.

This is made possible due to the application design that uses the same set of database tables and common business logic.

# Programs

Notably, is the naming convention where the 1st 2 / 3 letters denote the sub-system i.e. ‘dr’ for debtors, ‘stk’ for stock management etc.

The stored procedures used here represent the ***business logic and data persistence*** of the solution whereas the ***presentation logic*** only performs visual functions.

The presentation logic used the Borland C++ Builder development framework; originally used for its use in tertiary education centres i.e. UOFS. However, the framework was swapped in favour of MS Visual Studio with C# and subsequently Borland as a company has ceased business, making the C++ Builder framework obsolete

***Importantly, the Borland C++ framework MUST be replaced over the long term with potentially re-developing the user interfaces in MS C# in and amongst many other UI opportunities such as web-based development.***

The effort to replace the presentation logic is near enough about 10% of the total code base; consequently represents a smaller percentage of effort.

# MS Windows Executables

|  |  |
| --- | --- |
| **Name** | **Description** |
| StkBuyout.exe | Compile name to include into EpMenu |

# SQL Stored Procedures

|  |  |
| --- | --- |
| **Name** | **Description** |
| StkBuyoutCreat |  |
| StkBuyoutDelete |  |
| DrSearchAddress |  |
| StkBuyoutNextLineId |  |
| StkBuyoutLoad |  |
| StkBuyoutCoverInvoice |  |
| CoGetNextCrtlRef |  |
| DrCheckCrLimit |  |
| coRptLog |  |
| StkBuyoutQtySold |  |
| StkBuyoutRelease |  |
| StkBuyoutDocumentLookup |  |
| StkBuyoutOpen |  |
| StkBuyoutNextOrderNumber |  |
| StkBuyoutValidateAccount |  |
| StkBuyoutValidateStockCode |  |
| StkBuyoutSupplierDelete |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 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 ………………………………………………………………………………