FUJ00171847

Document Title: OBCS Counter Component Design Report

Document Type: Technical Design

Abstract: This document describes the low-level functionality of the OBCS software, and also the persistent objects and messages used in the Pathway OBCS Implementation.

Status: Approved

Distribution: Steve Warwick

Author: Olu Abatan

Approval Authority: Steve Warwick

Signature/Date:

Comments to: Olu Abatan

Comments by:

FUJ00171847

FUJ00171847

0.1 CONTENTS

0.2 DOCUMENT CONTROL

0.3 DOCUMENT HISTORY

Version	Date	Reason
0.1	13/0197	First Issue
1.0	03/03/97	Second Issue-Changes to attribute grammar
2.0	15/12/97	This document has been administratively baselined in order to bring the document under formal change control

0.4 ASSOCIATED DOCUMENTS

Version	Date	Title
		EPOSS/TD/001 - EPOSS Attribute Grammar

0.5 ABBREVIATIONS

EPOSS	Electronic Point of Sale Service
OBCS	Order Book Control System

FUJ00171847

1. GENERAL

1.1 INTRODUCTION

This document provides a technical overview of the Pathway OBCS software solution. It contains a description of the functionality of OBCS, the persistent objects and messages used in the Pathway OBCS implementation; and also a look at the methods and properties exposed by OBCS classes.

FUJ00171847

1.2 SCOPE

The content of this document is confined to a description of functionality of the methods and properties exposed by classes in the OBCS software. Methods and properties, specifically meant for maintaining encapsulation within OBCS are not considered except in passing. There is also a detailed description of the OBCS attribute grammar catalogue.

FUJ00171847

2. SYSTEM OVERVIEW

The Order Book Control System (OBCS) software is a system which allows for the Receipt, Redirection, Issue and Encashment of valid Order Books. The system also allows encashment of valid non-barcoded books. The word valid in the mention of Order Books in this context, is important; as the system validates each order book against a local or national stop list each time an order book is processed. Validation, however, is not done when redirecting Order Books. OBCS depends on the EPOSS software for monetary transaction entries made to the message store.

Monetary transactions are handled by EPOSS for accounting purposes, because a customer session could consist of more than an OBCS transaction. A session represents a grouping of customer transactions, which usually relate to one customer. All other message store entries that have no monetary value attached to them are made directly to the message store by OBCS.

FUJ00171847

3. SOFTWARE COMPONENTS

There are no external references to OBCS classes apart from the clsOBCS, which is registered with the Riposte OLE client. ClsOBCS.CallInterface method serves as an entry point into OBCS, for the Riposte OLE client and other OLE applications such as EPOSS.

This chapter looks at the classes and methods exposed by OBCS starting with clsOBCS.

3.1 ClsOBCS

As mentioned earlier, this is the class registered with the Riposte OLE Client and is declared as creatable multiuse in the instancing property of the class.

The prominent methods this class expose include:

CallInterface() : This is the entry point for external calls from the Riposte OLE client or other OLE servers that might call OBCS. It consists mainly of a select case statement, which has different case variables to capture the '*cmd'* attribute value from the incoming message. The case statements are usually self-explanatory, but where they are not, comments are included in the code to show what the case statement does. There are various states, which are set by flags; and these states are determined by whether an Order Book being received, redirected, issued, encashed or if an order book is a non-barcoded Order Book.

BuildManualInput(): Manual inputs of order books in OBCS are actually carried out as Barcode scans. The BuildManualInput function builds an input string from the various entries of the Manual input screen. The customer reference number, the additional indicator, the serial number and the CPP number make up the manual input

FUJ00171847

string. This string is then sent to the validation objects as though it came from a barcode scan. However, transactions against this book are still recorded as transactions of an unreadable book.

MenuBtnPress(): This method captures the event that occurs when an OBCS registered button is pressed on the desktop. The registered buttons are the Receive, Redirect, Issue, Unread, Nobarcode and the Manual Input buttons- these captions might vary slightly from the actual captions on the Desktop.

StopFound(): This function makes entries into the message store, if a stop or a recall is found against a book. This method is called when a book is issued, encashed or received.

ConfirmNotDuplicate(): This method compares the previously read order book with order book , currently being read and returns a duplicate warning message, if the order books are similar and the states in which the order books were read are identical.

ProcessRequest(): This method encapsulates most of the process carried out on an order book, and is called from callInterface method of the clsOBCS class.

PerformLookUp(): This method is called to check if the order book is a local or foreign book. A book is a foreign book if the customer reference number on the book is not recognised at a local post-office. PerformLookup also calls the LocalStopFound() method-if the order book is a local one, or the RemoteStopFound() method if, otherwise. Both methods belong to the clsBook class and determine if there is a stop or recall against the book. If the order book is a local one, then the ***QueryOutcome*** method is called. The ***QueryOutcome*** method carries out various functions on the book depending on whether or the book was Manual Entry and also if a stop or recall was found against the book.

RegisterControls( ): This method registers the buttons used for OBCS on the Desktop.

FUJ00171847

ProcessManualTokenValidation(): This method calls the validation object with the built input string for a manual entry.

PanelDef(): This method holds the defination for the customised panel used in the OBCS issue, receive and redirect menus.

The three other classes used in OBCS are used mainly for the purpose of encapsulation , and are not called outside OBCS. They are:

- ClsBook
- ClsDisplay
- ClsMessage

3.2 ClsBook

The class contains methods that relate to information held on an order book. This class contains the following methods:

CheckCRNFound(): This method checks whether or not an order book is recognised at a local Post-office by checking for the customer reference number from the local message store. This method is called by the performLookUp method of the clsOBCS class.

CovertBarCode(): This method gets the various components from the barcode string. That is, the customer reference number, the additional indicator, the serial number and the CPP number. The ConvertBarCode method does a validation test on each of these entries by calling the appropriate validation methods. The validation methods for the barcode string are ***ValidCRN*** for validating customer reference number, ***ValidAddInd*** for validation of the additional indicator entry, ***ValidIOP*** for validation of the serial number and ***ValidSysInd*** for the validation of the CPP number. Even though a book passes the initial validation, there is still a need to make sure the book details adheres to the business rules

FUJ00171847

LocalStopFound(): This method actually does the checking of the message store to find out if a stop or a recall exist for a book, recognised as local order book.

RemoteStopFound(): This method first of all sets up a filter for any response the counter might receive in the local message store the with the attribute '<Data.TranType:OBCSLookupResult>'. A message is then sent to the agent with the attribute '<Data.TranType:OBCSLookup>' , together with the details of the order book. This message is wrapped up with create priority message' method of rclient. Attribute values to force an ISDN connection and to indicate how long the connection is to remain up are specified with create priority message. *The OBCS_LOADER_QUERY_STOPS* agent queries the Oracle database for any stops against the book.

If the ISDN line is up and running, an *OBCSLookUpResult* entry is made by the agent into the local message store, then the message store is updated with a Ninoenabled record. If there are any stops or recalls against the book the local message store is also updated with the stops or recalls.

When the RemoteStopFound method sets up a filter for a counter, the result of the query is sent to the clsOBCS.CallInterface with the *cmd* value of NotifyApplication'. This is processed in the case 'NotifyApplication' statement of the clsOBCS class. If for some reason the ISDN line is unavailable the same NotifyApplication case statement captures a time-out attribute then calls the clsMessage.Display() to display the appropriate message.

3.3 ClsScript

This class contains methods that are used for the display of OBCS scripts, the manual input script, the custom panel used by the receive, redirect and issue screens; and the script used for encashment. The methods for the clsScript include:

ManualInputScript(): This method displays the script used for manual entry.

FUJ00171847

UpdateCounterInfo() This method updates the counters on the screen for Scanned, Impound and Unreadable books when a receiving or redirecting order books.

DisplayInfo(): This method displays the scripts for encashment, impounds, recalls and issues

3.4 ClsMessages

The last class used by OBCS is the ClsMessages, this class contains just one method.

Display(): This method displays all the message on OBCS scripts that are picked up from the message store. Quite a number of messages are still hard coded. Messages are displayed with buttons, which call the clsOBCS.CallInterface() with the *cmd* values of 'Message', 'Message1', ConfirmEncash, ProcessDup and InvalidManual.

FUJ00171847

4. OBCS TRANSACTIONS

This chapter shows in pictorial terms what happens when an order book is scanned. The whole of this chapter concentrates on flow charts that show the interactions of the classes and methods when an Order Book is scanned, in different transaction states. The methods and classes referred to in the flow charts were fully discussed in the previous chapter.

Fig 4.1 illustrates the classes and methods called, when an order book is scanned for encashment. The shaded part of the boxes displays the class name while the bottom half displays the name of the actual method called within the class. The boxes with broken lines show processes that are executed outside OBCS. The parameters for the methods are not shown in the diagram, except for the CallInterface method of the clsOBCS class.

When an order book is scanned, the input string is passed to the validation object. If the book passes the validation test, then OBCS is called by EPOSSImpulse through the CallInterface of the clsOBCS class with the ***'cmd'*** value of 'IOEvent'. EPOSSImpulse is one of the ole applications used by EPOSS. '***cmd'*** is one of the attributes Riposte uses to when propagating from one application to another.

Fig 4.2 Shows the classes and methods called when an order book is scanned; while receiving, redirecting or issuing an Order Book.

Fig 4.3 Shows the classes and methods called when scanning a foreign order book.

FUJ00171847

Fig 4.1 Flow diagram for encashing a local order book.

Fig 4.2 Flow diagram for Receiving, Redirecting or Issuing an order book

FUJ00171847

Fig 4.3 Flow diagram for a encashing a foreign Transaction

FUJ00171847

5. Object Attribute Grammar

This section details the required attribute grammar structures used by the OBCS system. These structures take the form of messages placed in the message store and

FUJ00171847

are used by the OBCS Counter System to record transaction outcomes and to communicate with the Correspondence Server.

Fig 5.1 shows the various functions that can be carried out on an order book.

Fig 5.1

Figure 5.1 shows the various data paths that result in an entry into the message store, an order book can be Received, Redirected, Issued or Encashed, there are also nonbarcoded books which require a different sort of processing. Chapters 3 & 4 discuss the process shown in Figure 5.1 in detail. This chapter focuses on the entries made into the message store for each of these processes.
Entries made into the message store are made in numeric values, which corresponds to various states and Results defined below

Current definition for states in OBCS:

Receive	1
Redirect	2

FUJ00171847

Issue (Hanover)	3
Encash	4
NoBarcode	5

Current definition for Results in OBCS:

OK	1
IMPOUND	2
UNREAD	3
INVALID	4

5.1 Collection Objects

Local Nino Collection

This collection object is committed to the journal the first time an Order Book is presented at a Post Office. It registers that this is the nominated Post Office for the given Customer Reference Number. An exception to this case, would be if the customer has been to the post-office with another Order Book with the same Customer Reference number; i.e the beneficiary has more than one type of benefit.

Collection:NINOEnable	Collection Name
ObjectName:	Customer Reference Number
Data:	Core Data
CustId:	Customer Reference Number

5.2 Transaction Record

For OBCS transactions that have no monetary value associated with them, e.g. Receive, Redirect or Issue transactions, OBCS will be responsible for writing the transaction to the journal. These transaction will be in the following format:

FUJ00171847

Application	AppName ie. OBCS
Data:	
State:	Transaction state
IOP:	IOP Number
Result:	Transaction status result
EntryMethod	Indicates method of transaction
ForeignIndicator	Indicates foreign/local transaction
ForeignEnquiryTime	Shows the time for a foreign transaction
ForeignEnquiryTimeOut	Indicates whether or not a timeout ocurred with an attempted foreign transaction
Vouchers	Number of vouchers in a transaction

5.2.1 Receive

When an Order book is first received at the post-office the clerk is expected to do a receipt of the order book on the system. The system checks for the customer reference number and if it finds it then it is a local / recognised book. The system then goes ahead to make an entry in the message store. With Result = OK and State = Receive
A typical entry for receive would look like this:

<Message:<GroupId:197997><Id:1><Num:892><Date:06-Mar-1997><Time:10:14:38><User:ADM><Application:OBCS><TranType:Admin><Data:<**State:1**><IOP_ident:ZX999058B 02154><ForeignIndicator:0><ForeignEnquiryTime:><ForeignEnquiryTimeOut:1><Vouchers:0><EntryMethod:0><**Result:1**>><CRC:D46EB953>>

With <Result:1> being the equivalent of <Result:OK> and <State:1> being the equivalent of <State:Receive>
If the customer reference is not recognised then the system tries to do a foreign transaction. This should not normally happen under Receive mode, because a book not belonging to the post-office should be redirected except where the customer is receiving a benefit for the very first time.

FUJ00171847

5.2.2 Issue

When an Order book is handed over to the beneficiary by the post-office; the book is checked for any stops or recalls that may have been recorded against the book. If there are no stops or recalls, the system then goes ahead to make an entry in the message store. With Result = OK and State = Issue
A typical entry for receive would look like this:

<Message:<GroupId:197997><Id:1><Num:894><Date:06-Mar-1997><Time:10:15:07><User:ADM><Application:OBCS><TranType:Admin><Data:<**State:3**><IOP_ident:ZX999044D 02154><ForeignIndicator:0><ForeignEnquiryTime:><ForeignEnquiryTimeOut:><Vouchers:0><EntryMethod:0><**Result:1**>><CRC:C31D14A0>>

With <Result:1> being the equivalent of <Result:OK> and <State:3> being the equivalent of <State:Issue>
If the customer reference is not recognised then the system tries to do a foreign transaction. This should not normally happen under Issue mode, unless the book was not received-in by the clerk when the book was received at the post-office.

5.2.3 Redirect:

When an Order book is sent to the wrong post-office, the clerk should redirect the order book. The entry made into the message store would look like this:

<Message:<GroupId:197997><Id:1><Num:893><Date:06-Mar-1997><Time:10:14:53><User:ADM><Application:OBCS><TranType:Admin><Data:<**State:2**><IOP_ident:ZX999063C 02154><Vouchers:0><EntryMethod:0><Result:1>><CRC:A57C950F>>

The book is not checked for stops or recalls. The Entry <State:2> being equivalent to <State:Redirect>

5.2.4 Encash

FUJ00171847

For encashment transactions, OBCS will need to record the number of vouchers to be encashed together with the value of each voucher. EPOSS will be responsible for writing the transactions to the message store. An EPOSS transaction has two parts, namely core EPOSS attributes and Additional Data attributes. The Additional Data information will contain the OBCS specific details, namely, Customer Reference Number, Additional Indicator, IOP etc. An EPOSS transaction has the following format:

<Message:<GroupId:197997><Id:1><Num:896><Date:06-Mar-1997><Time:10:15:35><User:ADM><Drawer:ADM><EPOSSTransaction:<Ref:0100><SU:><CAP:><BP:><SessID:><TxnID:0><Pr
oductNo:884><Qty:-1><PVer:0><SaleValue:-0.45><EntryMethod:0><CrossReference:><StartDate:06-Mar-1997><StartTime:10:15:26><STF:4><EndDate:06-Mar-1997><EndTime:10:15:26><ETF:5><LkdTxns:0><BlackBoxData:<ForeignIndicator:1><ForeignEnquiryTime:00571000><ForeignEnquiryTimeOut:0><M:>><AdditionalData:<Vouchers:1><State:4><Result:1><IOP_ident:ZX999062B 02154>><PVer:0>><ServiceType:O><Debit:45><TranType:S><PM:<L1:510><L2:495><L3:3005><L4:3016><L5:3017>><SM:<L1:><L2:><L3:3163><L4:3168><L5:3172>><CRC:E5BA3350>>

EPOSSTransaction:	EPOSS transaction details
SU	Stock Unit
CAPBP:	
SessID:	Identifier for this session
TxnID:	Identifier for this transaction
ProductNo:	EPOSS ProductNo
RetailPrice:	Retail price of product
Qty:	
Pver:	
SaleValue:	Actual Sale Value
VAT:	VAT on this sale
EntryMethod:	MagCard, Keyboard etc
CrossReference:	Cross Reference to previous transaction
StartDate:	Date this transaction commenced

FUJ00171847

StartTime:	Date and time this transaction commenced
EndDate:	Date this transaction completed
EndTime:	Date and time this transaction completed
LkdTxns:	Logically linked transactions
BlackBoxData:	Data provided by extended applications
M:	TransactionMode
AdditionalData:	Specifics of OBCS transaction
Vouchers:	Unique transaction per counter position
State:	OBCS state
Result:	Order book status
IOP_ident:	Order book details
ServiceType:	
PM:	
Debit:	Transaction value in pence

5.2.5 Stop Collection

This collection object defines a stop against a given order book. Stop records will be distributed to Post Offices that have a Local Nino record for the given Customer Reference Number.

Collection:OBCSStop	Collection Name
ObjectName:	Customer Reference Number
Application: OBCS	Application Name
Data:	Core Data
AddInd:	Additional Indicator -
IOP:	IOP Number
SysInd:	System number
ActionType:	Instruction to Clerk

The ActionType field determines the instruction displayed to the clerk. Currently, there are three valid action types.

FUJ00171847

1 = Impound with Encashment (Recall)

2 = Impound without Encashment (Stop)

3=Purge stop/Recall

This is an example of the entry made into the message store by the OBCS_STOP_DISTRIB agent for a book with a stop against it, notice the action type attribute in bold. For a recall, the action type would be a value, two and the recall date attribute would contain the recall date for the Order Book.

<Message:<GroupId:444444><Id:1><Num:5361><Date:24-Dec-1996><Time:09:22:28><Collection:OBCSStops><ObjectName:TX328870D 16051><Data:<CustId:TX328870D><AddInd: ><CPPNo:019><IOPNo:16>**<ActionType:2><RecallDate:><Type:Enabled**>><Version:1><CRC:995D9FE7>>

If an error is made in loading down stops from the Benefit agency, and a stop or recall has been issued wrongly. A purge would be sent down from the correspondence server, this purge has the same format as a distributed stop. However, the Type attribute would be set to '*disabled*'.

If there is a stop against the book and the clerk tries to Receive Issue or Encash the book then an entry would be made into the message with the attribute similar to the example below. <Result:2> indicate that there is a stop against the book.

<Message:<GroupId:444444><Id:1><Num:5446><Date:07-Jan-1997><Time:18:33:02><User:ADMIN><Application:OBCS><Data:<State:1><IOP_ident:TX328870D 16051><Vouchers:0>**<Result:2**>><CRC:631FF18A>>

5.3 Message Formats

The following sections describe the attribute grammar format of messages written to the Message store to enable the processing of OBCS transactions.

5.3.1 Query Record

In cases where a LocalNino object is not found in the local Message store, it will be

FUJ00171847

necessary to lookup the central stop list to determine the status of an order book. Writing a message with a TranType of OBCSStopLookup together with the order book details does this. The message will have the following format:

Data:	Core data of message
TranType:OBCSStopLookup	Specifices lookup type
CustId:	CustomerRef Number
AddInd:	AdditionalIndicator -
IOP:	IOP Number
SysInd:	System Indicator