FUJ00117539

Document Title:	**LFS E2E Release 1 - Delta HLD**
Document Type:	**Design**
Release:	**S60**
Abstract:	**This is a delta to the High Level Design for the Logistics Feeder Service. This delta represents the changes required to the LFS software as a result of new requirements at E2E Re-Architecture Release 1.**
Document Status:	**APPROVED**
Originator & Dept:	**Pete Jobson– APDU**
Contributors:	**Pete Jobson FEL01**

Reviewed By:

Mandatory Review Authority	**Name**
RASD	Gareth Jenkins(*)
Customer Services	Mik Peach(*)
Development Unit – Design Team	David Johns
	Rex Dixon(*)
	Sud Agrawal
	Roger Barnes
	Trevor Leahy
	Chris Bailey
Development Unit – Development Team	
	Martin McConnel
	Walter Wright
	Matt Arris
Integration & Test	Janusz Holender

Optional Review/Issued for Information

Dave Wilcox	Tony Heaton
Nick Lawman	Roger York(*)
James Stinchcombe	Simon Fawkes
Neil Gormley(*)	Bob Gurney
Phil Boardman	Colin Mills(*)
Helen Pharoah(*)	Pete Ambrose
Andy Scott	(*) Those who returned Comments

Comments By:	**28th January 2004**
Comments To:	**Pete Jobson**
Distribution:	**Reviewers and Approvers**

FUJ00117539

0 DOCUMENT CONTROL

0.1 DOCUMENT HISTORY

Version	Date	Reason
0.1	27/10/03	Initial draft release
0.2a	31/10/03	Addition of LFS Host
0.2	06/11/03	Formal review cycle with LFS Host additions
0.3	N/A	Version skipped due to inconsistencies of document version with the version recorded in PVCS
0.4	26/11/03	Updates due to requirement changes in Pouch Collection suite of functions. Updates to LFS Host to include changes to Maestro scheduling and SLA Metric evaluation Updates following comments
1.0	19/01/04	Changes to Remittance Out modes (Introduction of ROSP, RODP & RISP Modes).

0.2 CHANGES THIS ISSUE

This section lists the major changes to functionality when comparing this document to Version 0.4. Minor corrections to spelling and grammar are not listed here. All changes can be reviewed by selecting 'Tools/Track Changes/Highlight Changes' in the Word™ menu and ensuring that there is a tick against 'Highlight changes on screen'.

1. 5.3.1.3.2.2: Pouch despatch changed from Housekeeping mode to RODP Mode
2. 5.3.1.3.2.4: Remit-Out of cash changed from ROAD Mode to ROSP Mode
3. 5.3.1.3.2.5: Remit-Out reversal of cash changed to RISP Mode

0.3 FORECAST CHANGES

None

0.4 APPROVAL AUTHORITIES

Name	Position	Signature	Date
Gareth Jenkins	E2E Design Authority		
Dave Johns	APDU Design Authority		

FUJ00117539

0.5 ASSOCIATED DOCUMENTS

	Reference	Vers	Date	Title	Source
[1]	LF/DES/002	1.0	05/03/99	Logistics Feeder Service – System Outline Design	Fujitsu
[2]	LF/BRD/001	1.0	20/5/98	LFS Business Requirements Definition	POL/ Fujitsu
[3]	None	1.0	09/04/98	Contract Schedule K01 - Service Levels and Remedies	POL
[4]	None	1.0	09/04/98	Contract Schedule K08 - Service Levels and Remedies	POL
[5]	TD/ARC/001			Technical Environment Description	Fujitsu
[6]	JED/LFS/001 (BP/DES/019)	1.3	11/08/99	Inventory Item Specification	POL (Fujitsu)
[7]	JED/LFS/005 (BP/DES/022)			LFS Bar Codes	POL (Fujitsu)
[8]	JED/LFS/007 (BP/DES/023)			LFS to SAPADS and SAPADS to LFS Application Interface Specification	POL (Fujitsu)
[9]	TD/DES/027			File Transfer Managed Service	Fujitsu
[10]	TSC/AGT/029			High Level Design of Generic Agent Components for Release 2	Fujitsu
[11]	LF/DES/005			Logistics Feeder Service Function Description	Fujitsu
[12]	SD/SPE/016			Horizon OPS Menu Hierarchy	Fujitsu
[13]	SD/DES/005			Horizon OPS Reports and Receipts - Post Office Account Horizon Office Platform Service	Fujitsu
[14]	LF/MAN/001			Logistics Feeder Service Systems Management Support Guide	Fujitsu
[15]	EA/DPR/002	0.3a	24/10/2003	E2E Re-Architecting Release 1 Design Proposal	Fujitsu
[16]	LF/DES/003	2.0	09/11/00	Logistics Feeder Service – High Level design	Fujitsu
[17]	EA/HLD/002	0.1	24/10/2003	Release 1 End-to-End High Level Design	Fujitsu
[18]	AD/DES/039	2.0	24/06/2003	Generic Agent Components for CSR+ HLD	Fujitsu
[19]	LF/IFS/001		26/11/2003	E2E Release 1 – LFS Counter Dialogue Delta – Activity & Screen Flows	Fujitsu

0.6 ABBREVIATIONS AND GLOSSARY

ACC Authorised Collectors Card

ADC Automated Distribution Centre

ADS Automated Distribution Service

FUJ00117539

AIS	Application Interface Specification
APS	Automatic Payment System
BP	Balance Period
CAP	Cash Account Period
EPOSS	Electronic Point of Sale Service
HLD	High Level Design
LFS	Logistics Feeder Service
MIS	Management Information System
ONCH	Overnight Cash Holding
PLO	Planned Order
POL	Post Office Limited
RDDS	Reference Data Distribution System
RDMC	Reference Data Management Centre
RIAD	Remittance In from Automated Distribution System
ROAD	Remittance Out to Automated Distribution System
SAN	Stock Advice Notice (also called SAPADS Advice Notice and Advice Notice)
SAPADS	SAP Advanced Distribution Service
SLA	Service Level Agreement
SOD	System Outline Design
TIP	Transaction Information Processing
TIS	Technical Interface Specification

Agent Chunk	Agent processes will divide the workload for a single process into chunks to provide small recoverable work units. A chunk will be a small subset of the outlets being processed by an Agent Stream and each chunk will be treated as a separate success unit.
Agent Stream	In order to achieve high throughput, multiple instances of an Agent process will be run in parallel. An agent stream can be viewed as multiple instances dynamically sharing the workload to provide a single logical agent. (This is a term introduced by this document and is not in general use within other agent documentation.)

FUJ00117539

BP Rollover	Balance Period Rollover A Post Office may divide a CAP into one or more Balance Periods and each Stock Unit within the Post Office will balance at the end of each balance Period. Typically, the larger Post Offices may treat each day as a balance period. BP Rollover uses the same cash, stamps and stock declaration as Stock Unit Rollover.
Branches	***A Branch is where the counter service for Horizon is available. These are normally the Post Offices but various mobile, in store and temporary Post Offices are included. This term supersedes the term 'Outlets'.***
CAP Rollover	The Post Office year is divided into CAPs (Cash Accounting Periods). At the end of each CAP the Post Master will balance the Accounts for the Post Office and move the Post Office forward into the next CAP. CAP Rollover is the term used for balancing the CAP and moving onto the next CAP.
Cash Declaration	The cash held by each counter is declared by denomination for the Stock Unit at the end of each BP or CAP.
Collection Group	A term used to define a group of Cash Pouches that will be collected by a courier in a single session. This is also know as a collection Sack within this document since the terms 'Collection' and 'Group' have distinctly separate meanings.
Collection Sack	See 'Collection Group' above. These terms are interchangeable and often used together for clarification. Ie: "Collection Group (Sack)".
Host	A host is the Unix/Oracle environment used within the data centre to support an application.
ONCH	Overnight Cash Holding This is a declaration by each counter at the Post Office of the cash denominations in the till at the end of the day.
Outlets	An outlet is where the counter service for Horizon is available. These are normally the Post Offices but various mobile, in store and temporary Post Offices are included. ***Note: This term has now been superseded by the term 'Branches'***.
Pouches	These are 'single use' plastic containers supplied by the POCL couriers used for transferring Cash and Stock between the Central Distribution centre (SAPADS) and the Post Offices. The pouches are not reused and pouches damaged while they are being filled are discarded. Separate pouches are normally used for cash and other stock.

FUJ00117539

Stamp Declaration	The stamps held by each counter are declared by denomination for the Stock Unit at the end each BP of CAP.
Stock Declaration	The Stock held by each counter in a shared Stock Unit is declared at the end of each BP and CAP. Stock Declarations are not required for Individual Stock Units.
Stock Unit	Stock Unit is the identification for the stock allocated to a counter. Stock Units can be either Individual, and the responsibility of a one person, or they may be shared where they can be the responsibility of many people. Shared Stock Units may also be spread across many counters.
Stock Unit Rollover	Prior to the BP or CAP Rollover, each Stock Unit in the Post Office is Rolled over or balanced. For an Individual Stock Unit, this process checks the declarations against balances. For shared Stock Units, the declarations of each counter are accumulated and the totals checked against balances. A successful Stock Unit Rollover results in the Stock Unit moving into the next BP or CAP.

FUJ00117539

0.7 TABLE OF CONTENT

FUJ00117539

FUJ00117539

FUJ00117539

FUJ00117539

FUJ00117539

0.8 LIST OF FIGURES

FUJ00117539

1 INTRODUCTION

This document specifies the high level design of the changes incurred to the Logistics Feeder Service at E2E Re-Architecture Release 1. It identifies the physical design related to the End-to-End Re-Architecting Release 1 Design Proposal[15].

This document relies heavily on the Logistics Feeder Service High Level Design [16]of which it is a Delta. The format and structure of this document will mimic that of document [16]in such a manner that it minimises the effort required to finally update document [16]to a new baseline level once all the changes have been implemented.

As a result, this delta document will contain sections that either *replace* sections of document [16]or are *in-addition* to existing sections in document[16]. These replacement and additional sections will be surrounded by narrative texts that introduce the changes via reference to the Design Proposal[15].

For clarification, this document describes the changes incurred to the LFS Host, Maestro, LFS Branch Systems and LFS Agent Loaders/Harvesters. In addition, it also needs to consider impacts on other Branch/Counter sub-systems (such as the EPOSS Retail Broker).

This document does not consider the impacts incurred on any other part of the Horizon estate as a result of the requirements and design described in[15].

From hereon, various fonts and colour will indicate which parts of the textual content are part of the original descriptions extracted from the LFS HLD[16], are modifications to the HLD or are additional to the HLD.

- Text in Normal font is extracted from the original LFS HLD and is reproduced in this document to give an overall description of those modules that require change. This is included to provide context for the changes described herein.
- ***Text in italics and bold blue colour represents the changes that are necessary to implement the requirements of E2E Release 1.***
- ***Text Highlighted in Yellow is incomplete***

Where document headings exist without textual content, this is simply to indicate that these heading existed within the LFS HLD [16]and that no change is incurred to the original content within those document sections. It is recommended that the original LFS HLD [16]is read alongside this document in order to fully understand the changes described herein.

Note:	*Text that appears within borders summarises requirements, introduces changes or further enhances understanding of the document*

FUJ00117539

2 SCOPE

3 DESIGN PRINCIPLES

Note: These design principles complement or are in addition to the design principles already defined for LFS in [16]

- ***Since the Cash Account is destined to be replaced by a new POL Financial System in future release, any additions made to counter or other systems should not rely on either Cash Account common functionality or Cash Account data or summarisations. However Cash Account bought fwd figures will be required during the initial migration to enable initial Branch cash balances to be derived.***
- ***Apart from the above statement, common functionality should be re-used wherever possible.***
- ***Due regard should be paid to the mechanisms of data storage, retrieval and manipulation at all stages of design and development to ensure efficiency of operation/execution.***

4 REQUIREMENTS

They are summarised below, with detailed functionality being defined in section 5 of this document.

4.1 FUNCTIONAL REQUIREMENTS

4.1.1 Planned Orders

4.1.2 Advice Notices

4.1.3 Daily Cash Statements

Cash Statements are details of Cash denominations by volume and value for an Outlet. LFS Counter produces an overall Cash Statement representing all Stock Units at the Outlet at 'End of Day'. The Cash Statement is based on either Overnight Cash Declarations or Cash Declarations for Stock Unit Rollover. A Null Cash Statement is produced if the declaration for any Stock Units or Drawers is missing.

Note: Additional requirement

As well as delivering declared cash values, the daily cash statement will also contain a calculated Branch cash balance for cash retained in the Branch.

FUJ00117539

4.1.4 Weekly Stock Statements

4.1.5 Pouch Deliveries

The bar codes for SAPADS pouches delivered at the counter are recorded using LFS counter.

Note: Additional requirement

The Pouch Id from the barcode is matched with the Replenishment Delivery Notice and the value of the Pouch is automatically Remitted-in. Stock Pouches do not have an associated Replenishment Delivery Notice and the Remittance-in must be manually performed.

The bar codes for the delivered pouches will be harvested and sent to SAPADS.

4.1.6 Pouch Collections

Volume and value of each item being remitted back to the Automatic Distribution Centre in a pouch are associated with a bar code and noted at the LFS counter. The collection of these pouches is declared using LFS counter and details of the collected pouches and stock items are recorded for Transmission to SAPADS.

The bar codes for collected pouches and stock items associated with each pouch are harvested and sent to SAPADS.

Additional Requirement:

Remittance-out of the actual cash value of cash pouches will be automated at the time of the Pouch Collection.

4.1.7 Pouch Collection Groups

Additional Requirement:

Cash Pouches that have been packed ready for collection must be added to a collection group (synonymous with a postbag or 'sack') before the courier comes to collect. This will simplify the collection process since the pouches and the collection receipts will be prepared in advance. The collection process is therefore simplified to the identification of the sack to be collected.

4.1.8 Replenishment Delivery Notices

Additional Requirement:

Replenishment Delivery Notices are sent to individual Branches and contain details of the volumes of cash denominations that are contained in each Pouch to be delivered by carrier. These notices are used during the delivery (Receipt) of the Cash pouches into the Branch in order to automate the cash remittance-in procedure.

Multiple Replenishment Delivery Notices are sent from SAP ADS on a daily basis.

FUJ00117539

5 SYSTEM COMPONENTS

5.1 LOGICAL ARCHITECTURE

5.2 PHYSICAL DECOMPOSITION

FUJ00117539

5.3 COMPONENT DESIGN

5.3.1.1 LFS Host (Process1)

LFS Host is based on the generic Host architecture. It manages data flows in both directions with SAPADS by accepting inventory data for distribution and sending stock control data collected from outlets. There are ***Five*** elements to LFS Host:

- Planned Orders
- Advice Notices
- Stock Levels
  SAPADS require details of stock, including stamps and cash, held at the Outlet. Cash volumes for each Outlet by denomination are required daily. The data will be calculated at the counter from ONCH declarations at end of day ***and a Branch Balance figure will be automatically calculated to provide a level of confidence in the manually declared sum***. Other Stock volumes are required weekly and the data will be calculated at the counter from a combination of stock declarations and opening balances produced by CAP rollover.
  SOD References: 6.4.6, 6.4.8
- Pouch Details
- Send SLA to MIS.
- ***Replenishment Delivery Notices***
  A file of Replenishment Delivery Notices will be sent by SAPADS on multiple occasions during the day. The file will contain a number of records detailing the cash denominations and values that have been despatched to the Branches. Only basic format and data-type validation is performed on the file contents. The Replenishment Delivery Notices will be sent to the Branch provided the Branch is open or temporarily closed.

FUJ00117539

5.3.1.1.1 Planned Orders (Process 1.1)

5.3.1.1.2 Receive Advice Notices (Process 1.2)

5.3.1.1.3 Replenishment Delivery Notices (Process 1.6)

The stages for processing Replenishment Delivery Notices are:

1. ***Receive Replenishment Delivery Notices***
   Replenishment Delivery Notices file will be created by SAPADS at the LFS/SAPADS boundary and transferred to LFS Host by FTMS. The contents of the file will be validated and transformed for loading onto the Oracle Host database. Any corruption in the file, Data formats or record structures will result in the file being rejected and the process being abandoned.
2. ***Validate Replenishment Delivery Notices***
   FAD codes for valid Replenishment Delivery Notices received from SAPADS will be validated against FAD codes for Outlets from RDMC. Valid Replenishment Delivery Notices will be written to the Agent Table for message generation. Invalid Notices will be written to an Invalid Replenishment Delivery Notices table.
3. ***Send Rejected Replenishment Delivery Notices.***
   Invalid/Rejected Replenishment Delivery Notices will be written to a file and returned to LFS/SAPADS boundary using FTMS.
4. ***Check Replenishment Delivery Notices Load***
   Ensure that all Received Replenishment Delivery Notices have been processed by the Replenishment Delivery Notices loader process.
5. ***Replenishment Delivery Notices End of Day***
   Check on Replenishment Delivery Notices files that have been detected by the LFS Daemon process.

FUJ00117539

5.3.1.1.3.1 Receive Replenishment Delivery Notices (Process 1.6.1)

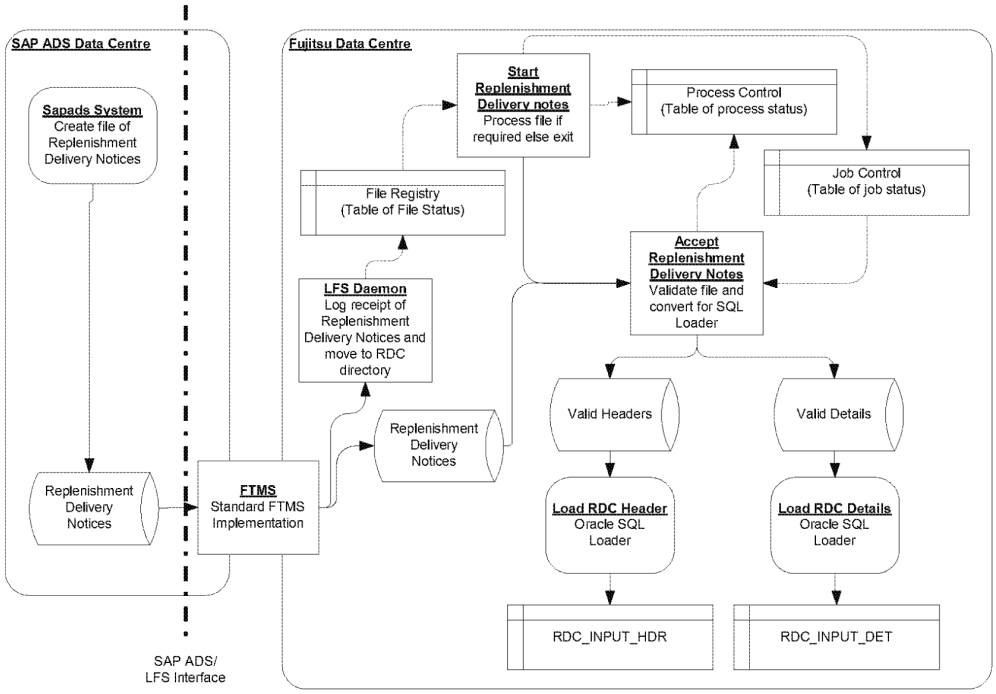

Figure 1 – Receive Replenishment Delivery Process

The aim of this process is to get the Replenishment Delivery Notices supplied by SAPADS onto the Oracle LFS Host database. The overriding assumption is that the file supplied by SAPADS will always be in the correct format (see [8]). There is no comprehensive error reporting functionality built into this process and it will stop and reject the complete file if a corruption is found. The components of this process are shown in Figure 1.

Replenishment Delivery Notices FTMS Link

The SAPADS system will create a Replenishment Delivery Notices file in the client FTMS Host at the SAPADS data centre. FTMS will be configured as follows:

- ***Transfer files from the client FTMS Host in the SAPADS data centre to the Fujitsu FTMS Host at the Fujitsu data centre.***
- ***Secure the transferred file for Archive and Audit.***

A full description of all the parameters required for FTMS will be defined in the FTMS Configuration for LFS and SAPADS.

Replenishment Delivery Notices LFS Daemon

Common process 'LFS Daemon' will detect the Replenishment Delivery Notices file delivered by FTMS. It will log the 'Receipt of the file' in the File Audit Trails table, update the File Registry table for the file and move the file to the Replenishment Delivery Notices Processing Directory.

FUJ00117539

Start Replenishment Delivery Notices (LFSC400)

Maestro will run the 'Start Replenishment Delivery Notices' process according to the schedule. Start Replenishment Delivery Notices will initiate a complete run if there is a Replenishment Delivery Notices file to process otherwise it will end:

- *Check that all previous 'Load Replenishment Delivery Notices' Jobs have been completed. There must be no 'Replenishment Delivery Notices' Jobs with a status of Started.*
- *Find a Replenishment Delivery Notices file in the file Registry that has not been processed.*
  Select row from File_Registry table for a 'Replenishment Delivery Notices' file with null Job_Seq.
  If multiple rows are found then the row with the earliest Inserted_Tsmp should be used.
  Note:
  Need to ensure exclusive access to File Registry table at the start of this process to avoid two processes attempting to use the same file.
- *If there is no file to process then*
  End the Process with a 'Cancel' response
  Exit to Maestro.
- *If the received file is more than 3 days old then*
  Reject the file because we will be delivering Replenishment Delivery Notices that have already been delivered.
  Raise an exception
  Exit to Maestro with a 'Cancel' response.
- *If a File is available for processing then*
  Insert a Job_Control row and associate it with the received file.

Accept Replenishment Delivery Notices (LFSC401)

The Replenishment Delivery Notices file will be pre-processed by the Accept Replenishment Delivery Notices function:

- *Validate the input file:*
  1. *First record in file is a valid File Header record.*
  2. *Following the File Header record, the file will be processed in complete Replenishment Delivery Notices consisting of a Replenishment Delivery Notice Header record followed by Replenishment Delivery Notice Detail records.*
     All records in each Replenishment Delivery Notice must be valid format and data type. (see [8])
  3. *If the file contains no Replenishment Delivery Notices, then complete the process with a success status and exit with a value of '1'. This will cause Maestro to skip any further processing of the file and clean-up by calling End Replenishment Delivery Notices (LFSC099 'RDC')*
  4. *A Replenishment Delivery Notice must not have more than 1000 details.*

FUJ00117539

5. *The last record in the file must be a File Trailer record.*
6. *The counts on the trailer record must be correct.*

- *Every valid Replenishment Delivery Notice Header record will be reformatted and written to the Replenishment Delivery Notices Header pipe for subsequent loading using Oracle SQL loader.*
- *All valid Replenishment Delivery Notice Detail records will be reformatted and written to the Replenishment Delivery Notice Detail pipe for subsequent loading using Oracle SQL loader.*
- *No attempt will be made to checkpoint the input file during the validation process. If there is a failure during the validation process then the pipes created before the failure will be discarded and the complete input file will be reprocessed.*
- *Update the File Registry table row for the Input File to set record count and to set File Status to VALID at the end of the process.*

Load Replenishment Delivery Headers and Details tables. (LFSC402 & LFSC403)

The valid Replenishment Delivery Headers and Details pipes will be loaded in parallel into the RDC_INPUT_HDR and RDC_INPUT_DET tables using Oracle SQL Loader (see 13.2.28 and 13.2.27). There will be no check-pointing and restarting during the loading process. The contents of the tables will be discarded before the load starts so a restart will mean that all rows processed before a failure must be processed again.

Process Schedule

The LFS Daemon will be scheduled to run daily after 'LFS Start of Day' completes. It will be stopped by LFS End of Day.

The 'Start Replenishment Delivery Notices' process will be invoked by Maestro whenever a file is detected as being received by the LFS Daemon and not yet processed by the 'Start Replenishment Delivery Notices' process. This means that a Replenishment Delivery Notices file will be processed shortly after receipt by the LFS Daemon. In practice, the LFS Daemon will execute between the hours of approximately 02:00hrs and 23:59hrs. A full description of the interactions between file receipt and schedule invocation can be found in section 17.7.

FUJ00117539

5.3.1.1.3.2 Validate Replenishment Delivery Notices (Process 1.6.2)

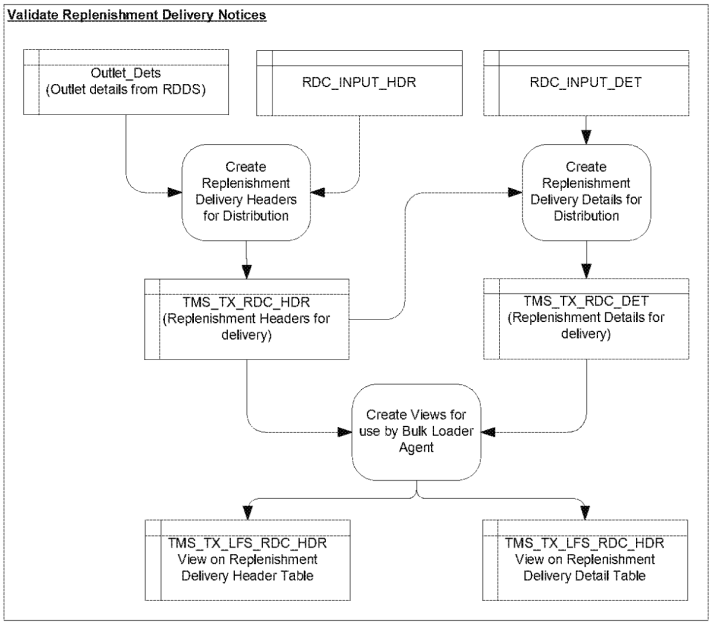

Figure 2 –Validate Replenishment Delivery Notices Process

Note: LFSC404

The aim of this process is to generate tables of Replenishment Delivery Notices for distribution to the Outlets. The overall process for an input file is completed before processing on the next file is started. Therefore, within this process it is reasonable to treat each phase as a complete unit that can be rerun on failure. Each phase will drop the output table and recreate it as a single select statement. There should be no co-ordination problems with this table since the Agent process that uses this table cannot start until this process is completed.

The Valid Replenishment Delivery Headers and Details tables used in these process have a suffix of YYYYMMDD_NNN where YYYYMMDD is formatted from Job_Date and NNN is Job_No from Job_Control table row for the 'Replenishment Delivery Notices' with status of started (see 13.1.10).

For the Replenishment Delivery Notices process in the Job_Control table insert rows in the Table_Registry table for Valid Replenishment Delivery Header and Valid Replenishment Delivery Details table.

The first phase of the process will join the Input Replenishment Delivery Headers and Outlet details from RDMC and create the Valid Replenishment Delivery Headers table for transmission to the Outlet counters where the Outlet is open. The header data is enriched with the performance measurement metric at this point:

- ***Input Tables/Views:***

Description	*Name*	*Table/View*
Input Replenish Dlvy Headers	*RDC_INPUT_HDR*	*Table*
Outlet Details	*Outlet_Det*	*View*
Performance Metric	*RDC_Metric*	*Table*

FUJ00117539

- *Created Table:*
  Agent Transmission Table for Replenish Dlvy Headers *TMS_TX_RDC_HDR….*
- *Join Conditions:*

```
RDC_INPUT_HDR.FAD_Code = Outlet_Dets.Fad_Code and
Outlet_Dets.Automated = 'Y' and
Outlet_Dets.Office_Status < 1 and
(RDC_INPUT_HDR.Received_tsmp -
TRUNC(RDC_INPUT_HDR.Received_tsmp)*86400)
between  rdc_metric.start_time and rdc_metric.end_time
```

The second phase of the process will copy rows from the Input Replenishment Delivery Details table to the Valid Replenishment Delivery Details table for transmission to the Outlet counters where there is a valid header in the Valid Planned Order Headers table:

- *Input Tables/Views:*

Description	*Name*	*Table/View*
Valid Replenish Dlvy Headers	*TMS_TX_RDC_HDR*	*Table*
Input Replenish Dlvy Details	*RDC_INPUT_DET*	*Table*

- *Created Table:*
  Agent Transmission Table for Replenish Dlvy Details *TMS_TX_RDC_DET….*
- *Join Conditions:*

```
RDC_INPUT_DET.FAD_Code = TMS_TX_RDC_HDR.Fad_Code
```

The third phase of the process will create 2 views for use by the Agent Bulk Loader process. The first view will allow the Replenishment Delivery Header table to be Selected and Updated:

- *Input Tables/Views:*

Description	*Name*	*Table/View*
Valid Replenish Dlvy Headers	*TMS_TX_RDC_HDR….*	*Table*

- *Created View:*
  Agent Replenish Dlvy Header View *TMS_TX_LFS_RDC_HDR*
- *Selection Criteria:*

```
SELECT  HDR.Received_Tsmp                   Received_Tsmp,
        HDR.RDC_Inp_Seq                     RDC_Inp_Seq,
        HDR.Pouch_Id                        Pouch_Id,
        HDR.Cash_Centre_Id                  Cash_Centre_Id,
        HDR.Replenishment_Id                Replenishment_Id,
        HDR.Delivery_Date                   Delivery_Date,
        HDR.Agent_FAD_Code                  FAD_Code,
        HDR.Processed_Tsmp                  Processed_Tsmp,
        HDR.Actioned_Ind                    Actioned_Ind
        HDR.Metric                          Metric
From    TMS_TX_RDC_HDR_YYYYMMDD_NNN         HDR
```

Where TMS_TX_RDC_HDR_YYYYMMDD_NNN is the new table.

The second view will allow Replenishment Delivery Details to be selected:

- *Input Tables/Views:*

Description	*Name*	*Table/View*
Valid Replenish Dlvy Details	*TMS_TX_RDC_DET….*	*Table*

FUJ00117539

- *Created View:*
  Agent Replenish Dlvy Details
  TMS_TX_LFS_RDC_DET

- *Selection Criteria:*

```
SELECT  DET.Received_Tsmp                   Received_Tsmp,
        DET.PLO_Inp_Seq                     RDC_Inp_Seq,
        DET.RDC_Line                        RDC_Line,
        DET.Item_Id                         Item_Id,
        DET.Item_Value                      Item_Value
        DET.Agent_FAD_Code                  FAD_Code,
From    TMS_TX_RDC_DET_YYYYMMDD_NNN         DET
```

Where TMS_TX_RDC_DET_YYYYMMDD_NNN is the new table.

Process Schedule

This process should be scheduled to start when the Receive Replenishment Delivery Notices process finishes. The process is not required if there are no Replenishment Delivery Notices to process or if the Receive Replenishment Delivery Notices process is abandoned due to a corrupt input file or if a Replenishment delivery File is empty.

FUJ00117539

5.3.1.1.3.3 Send Rejected Replenishment Delivery Notices (Process 1.6.3)

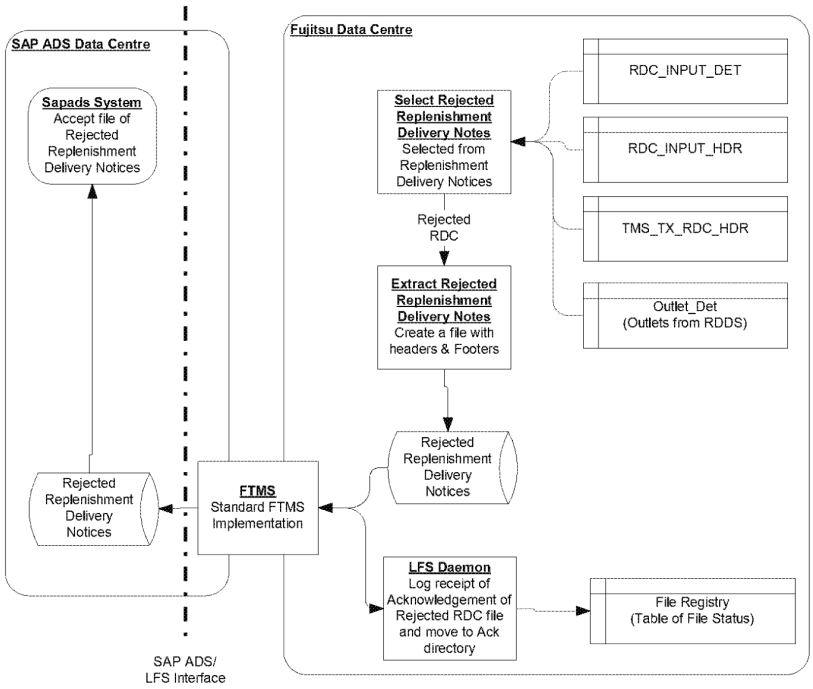

Figure 3 – Send Rejected Replenishment Delivery Notices Process

The aim of this process is to return the rejected Replenishment Delivery Notices on the Oracle Host database to SAPADS. The process, as shown in Figure 3, has three stages:

- ***Extract from the LFS Host Database***
- ***Transferred data by FTMS to the SAPADS Interface boundary.***
- ***Recorded Acknowledgement of Rejected File delivery.***

<u>Extract Rejected Replenishment Delivery Notices (LFSC405)</u>

The extract process will select the data from the Input Replenishment Delivery Header and Detail tables which has not been copied to the Valid Replenishment Delivery Header and Details Tables and output it to the file with file header and trailer (see 14.6):

- ***Insert row for created file in File_Registry Table. File_Date and File_No from Job_Date and Job_No on Job_Control table.***
- ***Create Rejected Replenishment Delivery Notices file.***
- ***Create a File Header record***

FUJ00117539

- *Create rejected Replenishment Delivery Notices by selecting from the Input Replenishment Delivery Headers and Details tables which have not been transferred to the valid Replenishment Delivery tables:*

  Select rows from the input Replenishment Delivery Header table which do not exist in the valid Replenishment Delivery Header table to create the Replenishment Delivery Header records.

  - *Input Tables/Views:*

Description	*Name*	*Table/View*
Input Replenish Dlvy Headers	*RDC_INPUT_HDR*	*Table*
Valid Replenish Dlvy Headers	*TMS_TX_RDC_HDR….*	*Table*
Outlet Details	*Outlet_Det*	*View*

  - *Join Conditions:*

```
RDC_INPUT_HDR.Received_Tsmp=TMS_TX_RDC_HDR.Received_Tsmp(+)
And RDC_INPUT_HDR.RDC_Inp_Seq=TMS_TX_RDC_HDR.RDC_Inp_Seq(+)
And Outlet_Dets.FAD_Code = RDC_INPUT_HDR.FAD_Code (+)
And TMS_TX_RDC_HDR.Received_Tsmp IS NULL
```

- *Write a File Trailer record.*
- *Check to see if there are no contents records (ie: no rejected records). If so, rollback File Registry and Job Control table updates and move the file to the 'unwanted' filename "Empty_Rejects_File" to avoid cleaning-up.*
- *Update File_Status on the File_Registry table for the Rejected Replenishment Delivery Notices file to indicate that it has been sent to SAPADs*
- *Raise an exception if the total of loaded Replenishment Delivery Notices and Rejected Replenishment Delivery Notices is not equal to the Accepted Replenishment Delivery Notices.*

Rejected Replenishment Delivery Notices FTMS Link

FTMS will be configured as follows:

- *Transfer files from the FTMS Host in the Pathway data centre to the client FTMS Host at the SAPADS data centre.*
- *Secure the file to be transferred for Archive and Audit.*

A full description of all the parameters required for FTMS will be defined in the FTMS Configuration for LFS and SAPADS.

Rejected Replenishment Delivery Notices LFS Daemon

Common process 'LFS Daemon' will detect the Acknowledgement for the Rejected Replenishment Delivery Notices file delivered by FTMS. It will log the 'Receipt of the file' in the File Audit Trails table, update the File Registry table for the file and move the file to the Acknowledgement file Processing Directory. (See 5.3.1.1.7).

Process Schedule

This process should be scheduled to start when the Validate Replenishment Delivery Notices process finishes. The process is not required if there are no

FUJ00117539

Replenishment Delivery Notices to process or if the Receive Replenishment Delivery Notices process is abandoned due to a corrupt input file.

FTMS will be configured to send the file when it is available in the sending directory.

The LFS Daemon will detect the Acknowledgement file from FTMS. The Replenishment Delivery Notices End of Day process will check that all files have been acknowledged and raise an alert for each file that has not been delivered within 24 hours.

FUJ00117539

5.3.1.1.3.4 Check Replenishment Delivery Notices Load (Process 1.6.4)

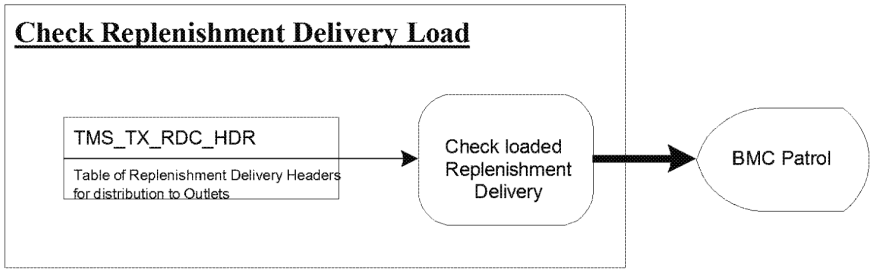

Figure 4 – Check Replenishment Delivery Notices Load Process

Note: Module LFCS406

The aim of this process is to ensure that the Replenishment Delivery Notices Loader has processed all rows in the Replenishment Delivery Notices table. This process will exit with error status to ensure that action is taken to process the missed data if the Replenishment Delivery Notices loader has not processed all received Replenishment Delivery Notices.

- ***Create an exception if there are any rows in the TMS_TX_RDC_HDR… table where ACTIONED_IND is set to 'N'.***
  The rows in the table at initialised with ACTIONED_IND 'N' by the Validate Replenishment Delivery Notices process and they should be updated by the Loader process to Null if the message was created successfully or 'F' if the message creation failed.
  Note:
  This process will not raise exceptions for Replenishment Delivery Notices that have failed to load because the exception should have been raised by the Replenishment Delivery Notices Loader already.

Process Schedule

This process should be scheduled to start when the Replenishment Delivery Notices Loader (Process 2.1) finishes successfully.

FUJ00117539

5.3.1.1.3.5 Replenishment Delivery Notices End of Day (Process 1.1.5)

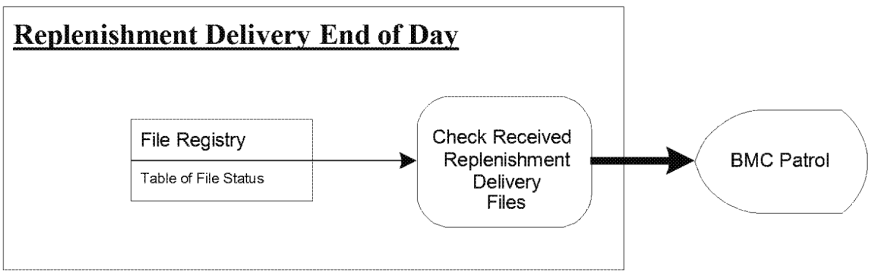

Figure 5 – Replenishment Delivery End of Day

This process will check the Replenishment Delivery Notices files that have been detected by the LFS Daemon and create alerts if necessary.

- ***Check the File Registry table for a Replenishment Delivery Notices file with the current Business date. Sixteen must be found on a weekday and Saturday, eight must be found on a Sunday. If this is not the case then raise an Alert. (LFSC407)***
- ***Check the File Registry table for every Rejected Replenishment Delivery Notices file that has not be acknowledged for over 24 hours. Raise an Alert for each such file. (LFSC408)***

Process Schedule

This process will execute after the LFS_Daemon has been stopped at 23:59. This is 2 hours after the last RDC file should have been received for the day and therefore all files for the day should have been both received and processed.

5.3.1.1.4 Stock Levels (Process 1.3)

FUJ00117539

5.3.1.1.4.1 Send Daily Cash Statements (Process 1.3.1)

Changes are required to this process (LFSC301) in order to ensure that daily tables and views are created with additional attributes as specified in section 13.2.41. This requires update of constant C_HDR_TABLE in LFSC301.h.

Changes are also required to LFSC304.pc and LFS304.h to additionally send the additional two fields for Generated Cash position to SAP ADS. These two additional fields will be turned-on via a soft switch held in the Application Parameters table 'R1_SAPADS_MIGRATE'. The additional fields will be passed to SAPADS only when the value of this parameter is 'ON'. See Section 13.3.1.

Migration: Old CashHeader tables need to be converted to the new structure before running the new schedule.

5.3.1.1.5 Pouch Details (Process 1.4)

5.3.1.1.6 Send SLA to MIS (Process 1.5)

5.3.1.1.7 Host Common Processes

Updates are required to lfscommon.h and common.h to #define error messages and other constants

FUJ00117539

5.3.1.1.7.1 LFS Daemon

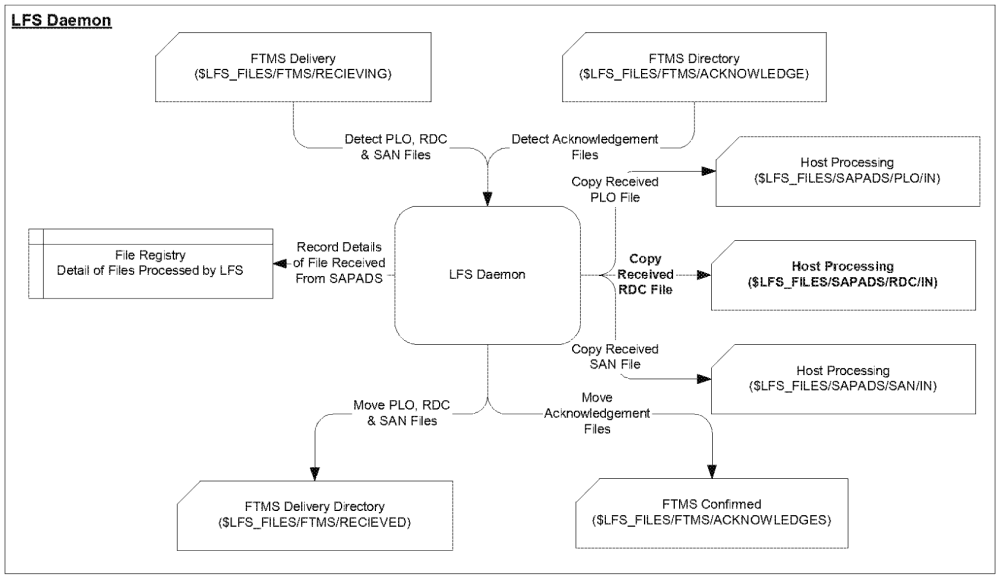

Figure 6 – Receive SAPADS File Process

This process will monitor the directories where the files are delivered by FTMS. Files sent by SAPADS will be prepared for subsequent processing. Acknowledgement files will be logged.

SAPADS Files

The process for Planned Orders, Replenishment Delivery Notices and Advice Notices files received from SAPADS.

- Log receipt of the file in the File_Registry.
  Duplicate file name then raise an exception.
- Copy the Received file to Host Processing Directory.

  File Type = 'PLO' then Copy the file to the Planned Orders Input Directory
  ($LFS_FILES/SAPADS/PLO/IN)

  File Type = 'SAN' then Copy the file to the Advice Notices Input Directory
  ($LFS_FILES/SAPADS/SAN/IN)

  File Type = 'RDC' then Copy the file to the Replenishment Delivery Notices Input Directory
  ($LFS_FILES/SAPADS/RDC/IN)

  Unknown File Type then raise an exception by inserting a row in the exception table.
- Move the Received file to Received Directory to inform FTMS that the file has been processed. FTMS will tidy the Received Directory.

FUJ00117539

Acknowledgement Files

FTMS generates an Acknowledgement file on the remote server when it delivers a file. This file contains details of the delivery and it is transferred back to the `Acknowledge` directory on the Host. The Host process must read the file and process the delivery details if the transfer was successful or raise an exception if the transfer has failed.

File Name

The name of the Acknowledgement file will be the name of the original file including the original extension with a '.ack.' or '.nak' extension added on. The '.ack' extension signifies a successful delivery and a '.nak' extension signifies a failure in the delivery process.

Example:

Sent file lf19990417001.plo
Successful Acknowledgement file lf19990417001.plo.ack
Failure Acknowledgement file lf19990417001.plo.nak

Contentsof a successful Acknowledgement file.

```
DeliveredServiceID= FTMS-CMS-DLR-TARGET PostProcess
DeliveredStatus= Success
FailureReason=
OriginalFileName= 100 SERVICES CHANGE.ctl
OriginalPath= C:\SEQUENT\DLRIN
OriginalMachineID= MILTONE
OriginalFileSize= 19968
OriginalCreationDate= 09/09/1999-14:32:56
OriginalCheckSum= 353084000
DeliveredFileName= 100 SERVICES CHANGE.ctl
DeliveredPath= C:\PATHWAY\CMS\RECEIVE
DeliveredMachineID= MILTONE
DeliveredFileSize= 19968
DeliveredDateTime= 09/09/1999-14:33:29
DeliveredCheckSum= 353084000
```

Contentsof a failure Acknowledgement file.

```
DeliveredServiceID= FTMS-CMS-DLR-TARGET Verify
DeliveredStatus= Failure
FailureReason= 54: FTMS-CMS-DLR-TARGET Verify:
Cryptography error during Verify of file C:\FTMS\FTMS-
CMS-DLR-TARGET\Verify\In\19990917131325.one9byte.crd:
0x8701001f: Check value did not match
OriginalFileName= one9byte.crd
OriginalPath= C:\SEQUENT\DLRIN
OriginalMachineID= MILTONE
```

FUJ00117539


```
OriginalFileSize= 1
OriginalCreationDate= 17/09/1999-12:44:14
OriginalCheckSum= 2746444292
DeliveredFileName= 19990917131325.one9byte.crd
DeliveredPath= C:\FTMS\FTMS-CMS-DLR-TARGET\Verify\In
DeliveredMachineID= MILTONE
DeliveredFileSize= 350
DeliveredDateTime= 17/09/1999-13:13:47
DeliveredCheckSum=
```

Processing

- Find the row in the File_Registry for the file sent to SAPADS.
  The name of the Acknowledgement file without the '.ack' extension will identify the File_Registry row.

  If the row is not found or the Status of the file is not sent then raise an exception and ignore the file.

  Acknowledgements are received for the following files:
  Rejected Planned Orders
  Rejected Advice Notices
  Rejected Replenishment Delivery Notices
  Cash Statements
  Stock Statements
  Pouch Delivery Details
  Pouch Collection Details
  Control File

- If a record is found with `DeliveredStatus= Success`
  then the file was delivered successfully.
  - A record must exist with `DeliveredDateTime= DD/MM/CCYY-hh:mm:ss`
    where `DD/MM/CCYY` and `hh:mm:ss` is the date and time respectively when the file was created on the remote server.
    If the record is not found then raise an exception and ignore the file.
  - Log the Delivery Date/Time for the file on the File_Registry row using the `DeliveredDateTime`.

- If a record is not found with `DeliveredStatus= Success`
  then FTMS failed to deliver the file.
  - A record must exist with `FailureReason = aaaaaaa.....aaaaaaa`
    where `aaaaaaa.....aaaaaaa` is reason for the failure.
    If the record is not found then raise an exception and ignore the file.

FUJ00117539

- Raise an exception for the failed FTMS transfer quoting the failure reason.
  The File_Registry entry will not be updated and the process will not be closed because it may be possible to get FTMS to transfer the file after fixing the reason for the failure.
  The process must be abandoned by operations if they cannot get the file transferred otherwise LFS will report an 'Acknowledgement Not Received' exception for the file.

- Move the Acknowledgement file to the `Acknowledged` directory to inform FTMS that the file has been processed. FTMS will tidy the `Acknowledged` directory.

Outstanding Files

The LFS Daemon will check which SAP ADS files are outstanding for processing and create a list of these files in the $LFS_FILES/SAPADS directory. This list is used by Maestro for scheduling the processing of such files. Refer to section 17.7 for further details of the file format.

5.3.1.1.7.2 Start of Day

5.3.1.1.7.3 End of Day

Create Control File (LFSC701)

This process needs enhancing to include Replenishment Delivery Notices files as valid file types for inclusion in the control file.

FUJ00117539

5.3.1.1.7.4 Tidy Application Tables

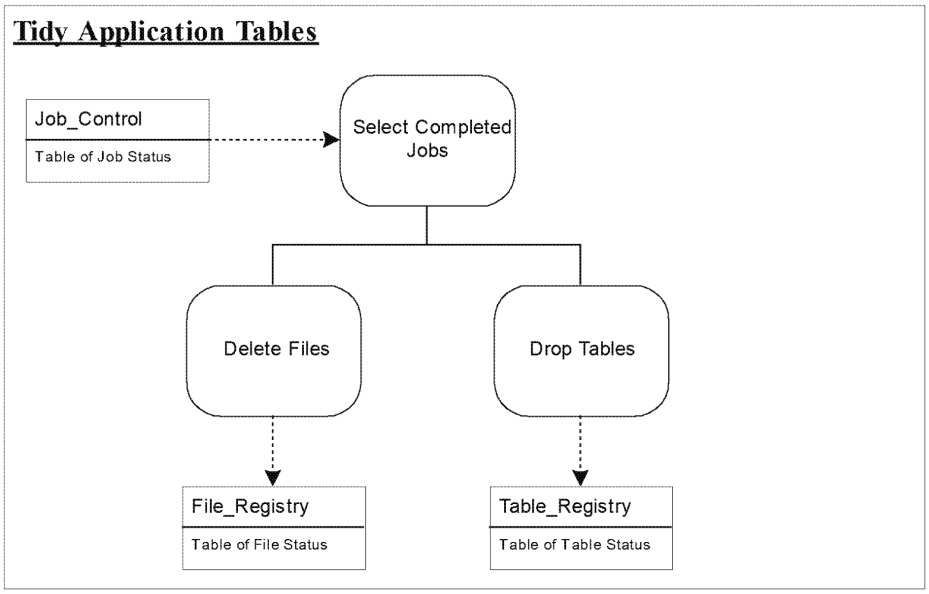

Figure 7 - Tidy Application Tables

This process will Tidy the Database Tablespaces by dropping tables and delete files in local directories that are no longer required. It will use an application parameter to determine when the tables and files can be removed.

Select completed Jobs

Select Retention Period from the APPLICATION_PARAMETER table for tidying tables. Select all Jobs that are marked as completed that are older than the Retention Period. All tables associated with the selected jobs can be dropped. All files associated with the selected jobs can be dropped.

Delete Files

For each completed job select all rows for Files in the File_Rigistry. Mark each row as deleted and delete the file associated with the table.

The following files are deleted:

- Received file (Planned Orders, ***Replenishment Delivery Notices*** and Advice Notices).
- Rejection Files (Planned Orders, ***Replenishment Delivery Notices*** and Advice Notices)
- Sent Files (Pouch Deliveries / Collections, Cash / Stock Statements and Control files).
- Acknowledgement files (for all sent and rejection files).

FUJ00117539

Drop Tables

For each completed job select all rows for tables in the Table_Rigistry. Mark each row as dropped and drop the table associated with the table.

The following tables are dropped:

- SENT_MIS_DATA_YYYYMMDD_NNN
- TMS_RX_CASH_DET_YYYYMMDD_NNN
- TMS_RX_CASH_HDR_YYYYMMDD_NNN
- TMS_RX_PCOL_DET_YYYYMMDD_NNN
- TMS_RX_PCOL_HDR_YYYYMMDD_NNN
- TMS_RX_PDEL_DET_YYYYMMDD_NNN
- TMS_RX_PDEL_HDR_YYYYMMDD_NNN
- TMS_RX_STOCK_DET_YYYYMMDD_NNN
- TMS_RX_STOCK_HDR_YYYYMMDD_NNN
- TMS_TX_PLO_DET_YYYYMMDD_NNN
- TMS_TX_PLO_HDR_YYYYMMDD_NNN
- TMS_TX_SAN_DET_YYYYMMDD_NNN
- TMS_TX_SAN_HDR_YYYYMMDD_NNN
- ***TMS_TX_RDC_DET_YYYYMMDD_NNN***
- ***TMS_TX_RDC_HDR_YYYYMMDD_NNN***

FUJ00117539

5.3.1.2 LFS Agent (Process 2)

All messages created for LFS will include a WAIndex container with a LFSFlag attribute. A local counter index will be created on this attribute and all retrieval of LFS messages can use the Index to retrieve the messages quickly and efficiently. The Index will not cause any performance problems on the counter because there will be relatively few LFS messages per Outlet. The value associated with the LFSFlag attribute will further subdivide the messages by message type:

PO	Planned Order
AN	Advice Notice
PD	Pouch Delivery
PC	Pouch Collection
CM	Cash Marker
CS	Cash Statement
SK	Stock Marker
SS	Stock Statement
RO	Remittance Out
DN	Non-Value Stock Declaration
CN	Non-Value Stock Confirmation
RP	Remittance Pouch
EP	Empty Pouch
PI	Print Initialisation

Note: Additional Messages

RD	***Replenishment Delivery***
PR	***Pouch Reversal (reverse Remittance-Out)***
SA	***Collection Sack (Collection Group)***

LFS will require ***four*** Agent processes:

- Planned Orders Loader
- Advice Notices Loader

Note: Additional Agent

- ***Replenishment Delivery Loader***

  A separate Agent Loader process will load Replenishment Delivery Notices in the same manner as the Planned Orders Loader. The structure of the messages being loaded will be the same as the Planned Orders, however the actual Riposte Attribute Grammar will be different. A separate loader will be used due to the possible differences in Service Levels and frequency of Loader operation.

- LFS Harvester
  Updated to include the harvest of a 'Generated Cash Position' attribute.

FUJ00117539

5.3.1.2.1 Planned Orders Loader (Process 2.1)

5.3.1.2.2 Advice Notices Loader (Process 2.2)

5.3.1.2.3 Replenishment Delivery Loader (Process 2.4)

Note: Additional Agent that almost exactly replicates the Planned Orders Agent

This is an implementation of a 'Bulk Loader' Agent which will read data from the Valid Replenishment Delivery Notice Tables, `TMS_TX_RDPC_HDR_YYYYMMDD_NNN` and `TMS_TX_RDPC_DET_YYYYMMDD_NNN`, and create one message for each Replenishment Notice (Refer to [18] for generic aspects of Bulk Loaders). The Agent process will use views, `TMS_TX_LFS_RDPC_HDR` and `TMS_TX_LFS_RDPC_DET`, to provide independence from the physical implementation of the tables. Each view will only include the columns of interest to the Agent process.

Processing Rules:

- ***Process Management***
  The valid Replenishment Notices will be processed by multiple instances of the Replenishment Delivery Agent process. Each instance of the process will examine the Agent Work table for Replenishment Notices, `TMS_AWT_LFS_RDPC_HDR`, to determine the subset of the outlets to process. Each row in the table will specify a range of Outlets to be processed and the instance will process Replenishment Notices for that range of Outlets. When each instance completes its allocation of Outlets then it marks the row in the Agent Wait table as processed.
  If an instance fails to complete due to a failure then a subsequent instance can process the data. This recovery situation can result in duplicate messages being generated however, this is not visible at the counter and has no ill effects on counter functionality. This has been accepted as a rare failure condition and the additional complexity to remove duplicate Replenishment Notices at the counter has not been included (The Counter will simply use the first occurrence - In particular, any such duplicates should be identical)
- ***Data Selection:***
  Select data from `TMS_TX_LFS_RDPC_HDR` for a range of Group_Ids in Received_Tsmp, RDPC_Inp_Seq order.
  Select data from `TMS_TX_LFS_RDPC_DET` for a range of Group_Ids in Received_Tsmp, RDPC_Inp_Seq, RDPC_Line order.
- ***Data Update:***
  Update all rows selected from the `TMS_TX_LFS_RDPC_HDR` table setting Actioned_Ind to Null and Process_Tsmp to sysdate. The updating of the time stamp is no longer required for SLA calculations but it will remain in the process because it could be useful for problem resolution.
- ***Data Processing:***
  logically, each process will retrieve the Outlet range allocated from the `TMS_AWT_LFS_RDPC_HDR`. For the Range it will select all Replenishment Notice header records from the `TMS_TX_LFS_RDPC_HDR` and all the details

FUJ00117539

lines for the range of Replenishment Notice Header records from the `TMS_TX_LFS_RDPC_DET` *table. The data must be selected in sequence from both tables to ensure that each Replenishment Notice Header and its details are retrieved in the same order for message creation.*
Physically, the data is selected from the `TMS_TX_LFS_RDPC_HDR` *table in blocks of an optimal number of rows. For each Header row retrieved from the* `TMS_TX_LFS_RDPC_HDR` *the detail records will be retrieved from the* `TMS_TX_LFS_RDPC_DET` *table.*
I.E. There will be one select on the `TMS_TX_LFS_RDPC_DET` *table for each valid Replenishment Notice. When all rows from the Header block have been processed then the success unit will be committed before selecting the next block of rows from the header table. It is hoped that all header rows for a process will be retrieved in the first block.*
Note:
The updating for each retrieved block of data is performed immediately after the fetch. This means that the Processed_Tsmp will be set to the time before any messages are written to the message store although the update will only be committed to the database after the last message has been written to the message store.

- *Message creation:*
  One Replenishment Notice message must be created for each row selected from the `TMS_TX_LFS_RDPC_HDR` *table. The message will be constructed using a row from the* `TMS_TX_LFS_RDPC_HDR` *table and all associated rows in* `TMS_TX_LFS_RDPC_DET` *table with the same Received_Tsmp and RDPC_Inp_Seq. There may be between 1 and 1000 lines for each Replenishment Notice but the actual number is unlikely to exceed 30 lines for a Replenishment Notice because there are less than 30 Cash Product Lines supported by SAPADs. Approximately 4 to 8 lines for a Replenishment Notice is expected.*

  There will be a minimum of one detail line for every Replenishment Notice. If the generated message is too large for Riposte then the Data.Body container should be converted into an attachment and stored with the message (See 16.7).

- *Exception Handling:*
  All exceptions detected by the Agent Loader will be reported via Tivoli. Where the exception is related to a row then the Header row is marked as failed and an exception is raised via the TMS_Exceptions table. Any such failure should be non-fatal (ie RC1) and the agent carries on processing other rows

Process Schedule

This process will be scheduled via Maestro. It will be scheduled to start when the Validate Replenishment Notices process, see 5.3.1.1.3.2, ends successfully. It may run more than once in a day if more than one non-empty file is received from SAPADS. It will be cancelled in the Replenishment Notices Maestro schedule if no file has been received, if the whole file is rejected, all data in the file is rejected or

FUJ00117539

the file is logically empty. In practice it will probably be invoked up to about 15 times per day.

It is expected that Operational and Service Level Agreements will be defined at some point in the future. A soft mechanism is employed such that a measurement metric may be described in meta-data in a manner that allows the metric to vary depending on the time of day of receipt of the RDC file.

5.3.1.2.4 LFS Harvester (Process 2.3)

This is an implementation of an 'Interactive Harvester' Agent that will accept all LFS Data generated at the counter and store on the Host Tables. To provide some independence from the Physical implementation the Agent process will access the Host tables via views defined in the TMS Agent Interface user on the Host database.

Changes are required to harvest an additional attribute within the Daily Cash Statement Header record. This attribute will be the Generated Cash value as documented in the message structure (see section 15.9)

FUJ00117539

5.3.1.3 LFS Counter (Process 3)

The counter changes will be implemented in the same style as the existing applications and the generic changes such as menus and buttons will be implemented via changes to their associated documentation. All new messages, collections and objects will be detailed with the appropriate function. Changes to existing data structure will be noted with the functions associated with the function.

Three groups of functions are required to support LFS:

1. LFS On-Line View Functions

2. LFS On-Line Update Functions

Note: Change Pouch delivery to include Rem-in. Add function to reverse the remit-out of Cash. Add function to compile a 'Collection Group'

There are ***seven*** Counter functions that can be divided into two groups.

There are ***five*** functions that will capture details of Pouches delivered and collected.

The Pouch delivery function uses the bar code ***to automate the subsequent Remittance In from SAPADS function via reference to a previously received Replenishment Delivery Notice.***

The Pouch collection functionality, represented by the other ***five*** functions, satisfies the requirement to:

- Capture the details of the ***Cash/S***tock being returned in a Pouch
- ***Provides the ability to reverse (unpack) a Cash Pouch***
- ***Allows packed Pouches to be related together as a Collection Group***
- ***Records when Collection Groups are collected by couriers***

The other two functions satisfy the requirements for processing Non-Value Stock. One function will allow Non-Value Stock to be declared at Stock Unit level and the second function will allow the office manager to view a summary of the total Non-Value Stock declaration of the office and confirm that it is correct.

3. LFS Off-Line Functions.
   Two LFS off-line functions are required to gather data from the Riposte Message store and produce Cash Statements and Stock Statements. Daily Cash Statements will allow SAPADS to monitor the cash holding at Outlets and minimise the amount of working Cash held by Outlets. Stock statements, to be produced weekly, will allow stock balances to be monitored and replenished.

FUJ00117539

5.3.1.3.1 LFS On-Line View Functions (Process 3.1)

5.3.1.3.1.1 View Planned Orders (Process 3.1.1)

5.3.1.3.1.2 View Advice Notices (Process 3.1.2)

FUJ00117539

5.3.1.3.2 LFS On-Line Update Functions (Process 3.2)

The On-Line update functions implemented to support LFS are:

1. Confirm Pouch Delivery.
   This function will allow one or more bar codes for Pouches that have been delivered by couriers to be scanned. On confirmation of session end, two copies of Pouch Delivery notes will be printed and details of the delivered Pouches will be recorded in a message for subsequent harvesting and transmission to SAPADS.

Note: Additional Functionality

   The Pouch delivery function will locate the Replenishment Delivery Notice associated with all Cash deliveries. This will contain details of the contents of cash pouches and enable an automated Remittance-in of the Cash. Remittance Receipts will be automated for Cash Pouches.

2. ***Pouch Despatch***

Note: New Function

   This function allows a Collection Group of Pouches to be marked as having been collected by Courier. A financial transaction will be performed at the end of each Cash Pouch collection that will transfer the value of the cash from a Cash-in-Pouches product to the Cash In-Transit (Out) product hence removing the cash from the overall Branch liability. No financial transactions will be performed for Stock Pouches. Additionally, details of the Collected Pouches will be recorded in a message for subsequent harvesting and transmission to SAPADS. A Liability Transfer Receipt will be printed at the end of the session.

3. ***Prepare Collection***

Note: New Function

   Pouches must be gathered together into Collection Groups before the courier arrives to collect. The Pouch Id of each Pouch belonging to a Collection Group will be scanned or manually entered. The declaration of Group contents will be recorded as a Message for use later during the Pouch Despatch process. Collection receipts will be printed at the end of the process.

4. Confirm Remittance Out Pouch.
   The settlement process for the EPOSS Remittance Out functions for Automatic Distribution Centre will call this function. A Collection Pouch bar code will be scanned and associated with SessionId for the Remittance Out Session. These details will be used in the subsequent transmission when the Pouch is collected.

Note: Additional Functionality

   The Remittance-Out of cash will settle to a 'Cash-in-Pouches' product in such a manner as to retain the liability of the cash within the Branch.

FUJ00117539

5. Reverse Remittance-Out Pouch.

Note:	*New Function*

The manual reversal of remittance in/out of Cash will be disabled because cash pouch contents cannot be modified without destroying the Pouch and individual items within a pouch should not therefore be reversed. This function allows the Remittance-out to be reversed in its entirety. Each original Remittance transaction is located and reversed. An additional message 'Reverse Pouch' is written to indicate that the Pouch cannot be re-used or collected.

FUJ00117539

5.3.1.3.2.1 Confirm Pouch Delivery (Process 3.1.3)

Note: Revised Process

Counter Staff will confirm Delivery of Pouches by a courier by reading the bar codes on each pouch. The bar codes for the pouches in a Delivery session will be printed on two copies of the Delivery note. Both copies will be signed, by the counter staff, and the function will create a message to inform SAPADS of the Delivery being completed.

Where the Pouch Id indicates that the pouch contains Cash, the system will match the Pouch Id from the scanned/entered barcode with the Pouch Id contained in the Replenishment Delivery messages.

If a Replenishment Delivery message cannot be found for a cash Pouch Id that has been received, then the Postmaster will be presented with a prompt to enter the total cash value received for that pouch.

On completion, a Pouch Delivery Message will be written to the message-store for each scanned Pouch (as per current process).

In addition, where the delivered Pouches contain Cash, the cash value, deduced from the Replenishment Delivery message, is used to generate an automatic Remittance-In transaction for each Pouch. If the Replenishment Delivery message had not been found, then the manually entered total cash value will be used instead. (RIAD for Product 1). The Pouch Id will accompany the Cash Remittance-In Settlement transaction as a new attribute:

```
EPOSSTransaction.BlackBoxData.PouchId
```

Once the Auto-Remittance-In function is implemented, the existing Remittance-In ADC function must be modified such that it disallows the manual remit-in of cash.

FUJ00117539

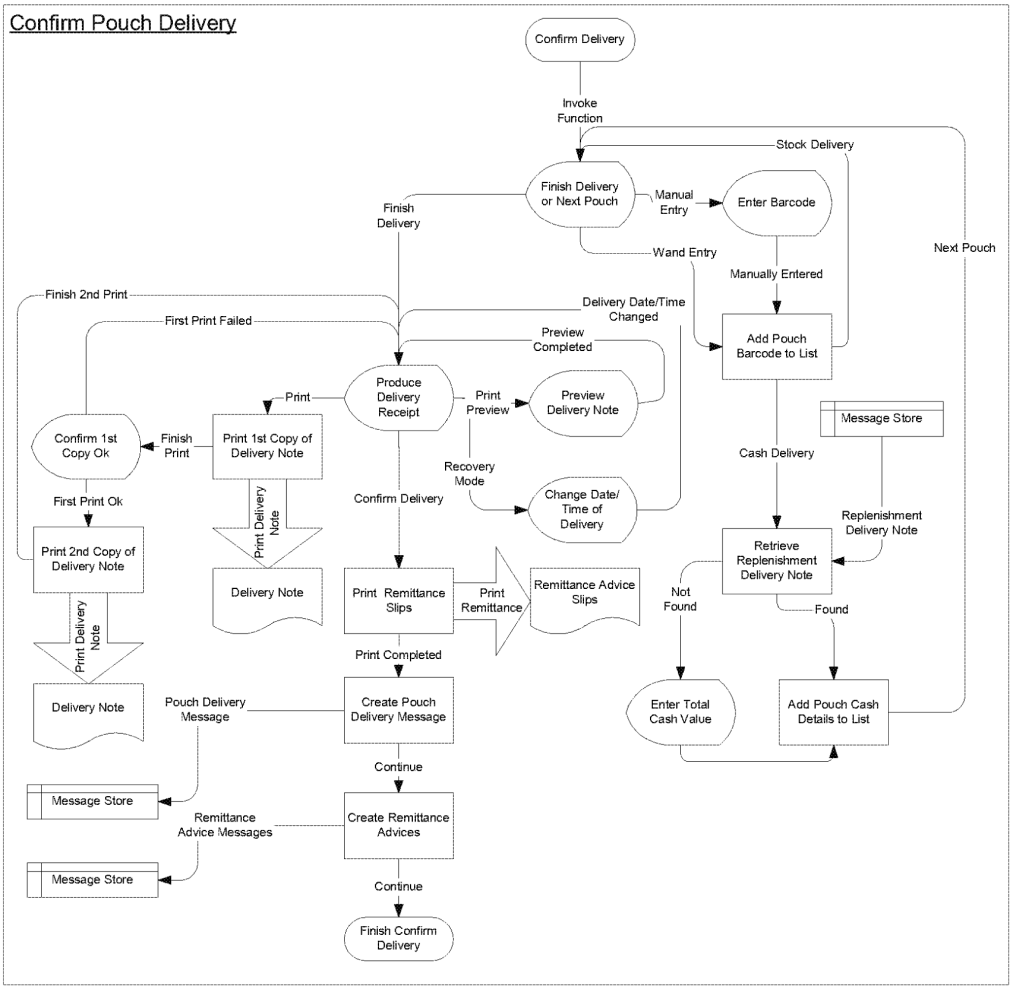

Figure 8 – Data and Process flow for Confirm Pouch Delivery

Trigger

- Menu Button (Pouch Delivery)

Processing

- Set context for reading Pouch bar codes and wait for bar code to be read. The panel should have button to allow manual input of Barcode and a complete button to indicate that the delivery is complete.
- If a Barcode is read using the wand then add the Barcode to the list of pouches and increment the count on the context panel indicating the number of pouches that have been included in the delivery.
- If Manual Input is requested then display calculator panel. Add the entered Barcode to the list of pouches and increment the count on the context panel indicating the number of pouches that have been included in the delivery.

FUJ00117539

- Bar Code validation should be as specified in [7]. Only Delivery Barcodes will be accepted. ***Stock can be differentiated from Notes and Coin Pouches via reference to the barcode***.
- The input of a duplicate bar code within the outlet will be rejected. The duplicate validation will not be comprehensive. It simply ensures that no confusion can occur within the Outlet with the existence of duplicate bar codes (pouch Ids). It will be possible to enter via keyboard or wand the same bar code at multiple outlets and to enter the same bar code at an Outlet after the original message has expired and it has been archived. ***Duplicate Cash-Pouch barcodes will be allowed during the migration period whereby Pouches that were received pre-migration may be received again Post-migration so that the value of the Cash Pouch may be remitted-in to the Branch. Pre-Migration Pouches can be identified by the lack of a Data.Manual attribute on the associated Pouch Delivery message.***
- ***If the Barcode indicates that the Pouch contains Cash then retrieve the associated Replenishment Delivery Message from the message store. (WAIndex.LFSFlag:RD) with matching PouchId. If not found, then display the calculator panel for entry of the Total Pouch Cash Value. The calculator panel will require the value to be entered twice to ensure that erroneous input is not accepted.***
- If delivery session ended via the complete button without any Pouch Declarations then Display an Error panel with options to return to Barcode scanning or to abandon the session. If the session is being abandoned then return to menu without creating a Delivery message otherwise wait for bar code to be read.
- When the Delivery session is ended using the complete button then present a Report panel to allow the delivery note to be printed or previewed. The panel should include a Complete button.
- If Print is selected then print the first copy of the Delivery Note and display a panel allowing the user an option to retry the print or continue with the print of the second copy. If retry is selected then return to the Report panel. If the first print is confirmed as Ok then print the second copy and return to the Report panel. The intermediate panel allows the user to take the first copy off the printer before the second copy is printed.
- If the complete button is used before the Delivery Note has been printed or previewed then display an error panel with options to return to the Report panel or abandon the Delivery Pouch session.
- The Change Date/Time button will be used in recovery mode to record when the delivery actually took place. The last entered date/time will be recorded as TransactionDate and TransactionTime attributes in the Pouch Delivery message and in all associated Cash Remittance messages.
  Note:
  The date and time will be displayed and entered in local time but it must be

FUJ00117539

stored as GMT. All dates/times in LFS messages should be in GMT and they should be displayed in local time.

- ***Following Delivery Note printing, print any remaining Cash Remit-in receipts (these will be for those cash pouches where the contents were deduced from the Replenishment Delivery Notice).***
- On completion create a Pouch Delivery message ***(with additional attribute(s) Data.Items.Item.Manual indicating whether The Cash value was derived from a Replenishment Delivery Notice or whether the Cash value was entered manually). Note: If the Pouch Id is a duplicate (due to the Migration issue noted earlier), then refrain from writing a duplicate Pouch Delivery message.***
- ***Create a Cash Remittance Transaction for each cash Pouch received. Balance the transactions to the Cash RIAD settlement product. One Settlement transaction is required for each Pouch since it must carry the Pouch Id within the*** `EPOSSTransaction.BlackBoxData.PouchId` ***message attribute.***
- Check that all Barcodes entered are unique before committing the message. There is a possibility that another counter at the Outlet may have been used to enter one or more of the Barcodes while the Barcodes were being scanned and printed. This final check will minimise the possibility of duplicate Barcodes being used within an Outlet.

Note:	***The Cash settlement product is different to the normal RIAD Settlement product. Details of how settlement products may be configured by Product/Mode are described in [18]***
	Also, Existing Reversal of a Cash RIAD transaction will be prohibited.

FUJ00117539

5.3.1.3.2.2 Pouch Despatch (Process 3.1.*4*)

A new button will invoke the Pouch Despatch function.

Where collection is by a Carrier who should have a valid ACC Card, then the ACC Barcode number will be checked for validity following a barcode scan or manual entry. Since the courier type for a Branch is fixed, the branch will either; always expect a valid ACC card to be produced or will never expect an ACC card to be produced. The system will determine whether to ask for an ACC card via the presence (or absence) of a dummy non-core Product definition.

Note:	***New reference data item. The 'ACC' product needs to be defined and distributed to correct Branches.***

On completion, the function will create a message to inform SAPADS of the collection being completed

In addition, a Remit-Out Despatch Pouches (RODP) transaction will be written for the value of each Cash Pouch. The …….

- ***Product being transacted is the Settlement Product that was used during the Confirm Remittance-Out process (Section 5.3.1.3.2.4).***
- ***Value is the value of the original Remittance-Out transaction.***
- ***Settlement Product is a Cash 'RODP' product (Cash in Transit)***

The Settlement transaction will contain an additional attribute:

`EPOSSTransaction.BlackBoxData.PouchId`

This attribute will be used by POL FS for reconciliation.

Finally, a Liability Transfer Receipt will be produced (Refer to [19]).

FUJ00117539

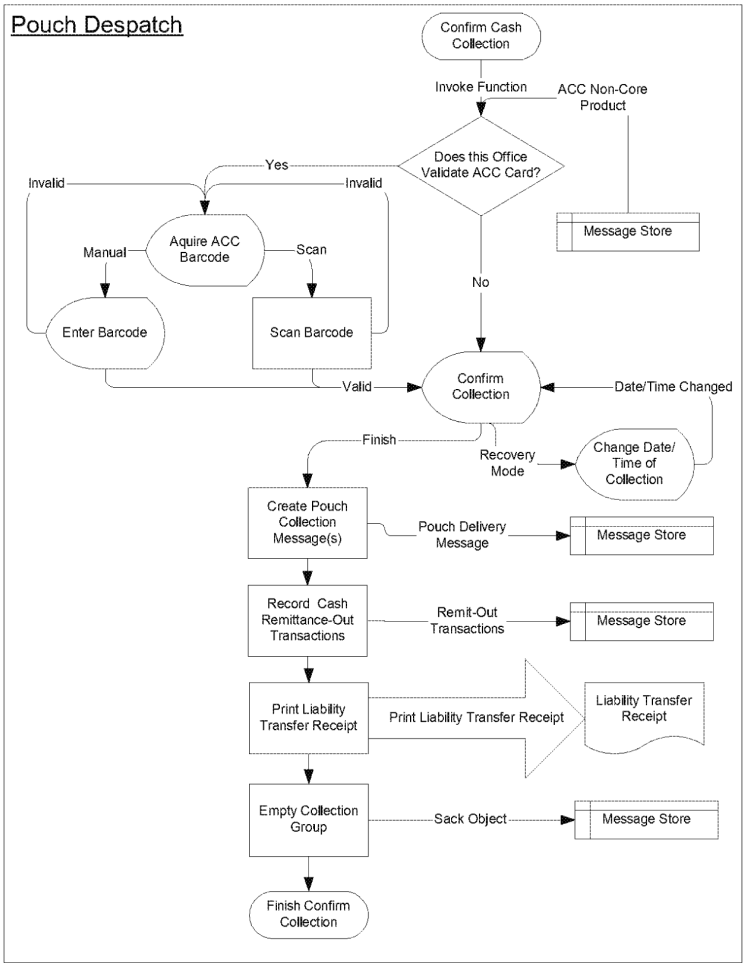

Figure 9 – Data and Process flow for Confirm Cash Pouch Collection

Trigger

- Menu Button (***Pouch Despatch***)

Processing

- ***Determine whether the Branch should verify the ACC Card carried by the courier by looking-up the dummy non-core 'ACC Product'. If product does not exist, set context for reading ACC bar code and wait for bar code to be read. The panel should have button to allow manual input of Barcode. Verify the barcode and allow re-try if invalid.***
- ***Bar Code validation should be as specified in [7]. Only Valid ACC Barcodes will be accepted.***
- The Change Date/Time button will be used in recovery mode to record when the Collection actually took place. The last entered date/time will be recorded as TransactionDate and TransactionTime attributes in the Pouch Collection message.

FUJ00117539

Note:
The date and time will be displayed and entered in local time but it must be stored as GMT. All dates/times in LFS messages should be in GMT and they should be displayed in local time.

- On completion Retrieve the latest version of the LFSSack message WAIndex.LFSFlag:SA.
- Create a Pouch Collection message.

  For each Pouch ***within the sack***, locate the LFS Remittance Out message (WAIndex.LFSFlag:RO) with a matching *PouchId* (bar code).

  Create Pouch message using Remittance Transactions identified by the LFS Remittance Out message.

  Create a Collection message identifying all pouches in the Collection ***Group***.

  Generate a new transaction for each Cash Pouch in the Collection Group. The Product being transacted is the Settlement Product that was used in original Remittance-out Transaction for each Pouch. The value for the new transaction is the same value that was settled in the original Remittance-out. Transact the Product in Housekeeping mode and settle to the Product as directed by Reference Data (This will be the new Cash in Transit product).
- ***Generate a Liability Transfer Receipt (Refer to [19]).***
- ***Write a new LFSSack message containing no Pouch Ids. This new message will have an incremented version number***

(See Pseudo code below for detailed processing)

Processing Logic

The processing rules above simplify the process for generating Pouch and Collection messages. The logic presented below provides the detailed processing.

All Remittance transactions associated with each pouch are read using the session Identified for the LFS Remittance Out message and a Pouch Collection message is created. After all pouches in the Collection Group have been processed a Collection message is created. It is created last to ensure that the LFS Harvester Agent will only receive a Collection message after all the pouch messages associated with the collection are on the correspondence server. This relies on Riposte continuing to replicate all messages from a single counter in the creation sequence.

As each remittance Cash Transaction is handled, accumulate the value. At the end of processing each Pouch, create a Housekeeping transaction for the Product used to settle the original Remittance-out transaction. The value of the transaction is the same as the value of the original Remittance-out settlement and the new transaction should be settled to the Product dictated by Reference Data.

Data Definitions:

FUJ00117539

Coll_Mgs	Table of Pouch Details for Collection. Details of each entry: PchId barcode PchSess SessionId for Pouch Remittance Out Session Pmess Message Number for Pouch Message RemId Counter for Remittance Transactions RemCnt Number of Remittance items for Pouch ***SettleProd*** ***The product used to settle the original remittance session.*** ***SettleValue*** ***The value that was settled in the original remittance transaction.***
Pch_Coll	Table of Pouchs for Collection. Details of each entry: Pch_Msg Table of Remittance Items for the Pouch
Pch_Coll.PchMsg	Table of Remittance Items for the Pouch. Details of each entry: RemNum Message Number of Remittance Out Transaction PNo Product Number from Remittance Out Transaction PQty Quantity from Remittance Out Transaction Pval Value from Remittance Out Transaction

Process Pouches:

```
Initialise Coll_Mgs(PchId, PchSess, Pmess, RemId, RemCnt)
Initialise Pch_Coll(Pch_Mgs)
PchCnt = 0
For each Pouch in the Collection (barcode)
    PchCnt = PchCnt + 1
    Initialise Pch_Coll.Pch_Mgs(PchCnt)(RemNum, PNo, PQty, PVal)
    Coll_Msg.PchId(PchCnt) = Pouch Id (barcode)
    Read LFS Remittance Out Message <WAIndex.LFSFlag:RO>
                              with Matching Pouch Id (barcode)
    If Message not found then
       If Cash Pouch
         Create Exception and Exit
       Else
         Coll_Msg.PchSess(PchCnt) = zero
         Coll_Msg.RemId(PchCnt) = zero
         Coll_Msg.RemCnt(PchCnt) = 1
         Set Pch_Coll.Pch_Mgs(PchCnt)
           RemNum(1) = zero
           PNo(1) = zero
           PQty(1) = zero
           PVal (1) = zero
       End if
    else
       Coll_Msg.PchSess(PchCnt) = TxnData.SessionId
       Coll_Msg.RemId(PchCnt) = <Id:> of LFS Rem Out message
       Read EPOSS Remittance Out Messages
                              with Matching TxnData.SessionId
       If there are no EPOSS Remittance Out messages
                              with Matching TxnData.SessionId then
         Log an exception                    (No Products remitted out)
         Coll_Msg.RemCnt(PchCnt) = 1.
         Set Pch_Coll.Pch_Mgs(PchCnt)
           RemNum(1) = <Num:> of LFS Remmittance Out message
           PNo(1) = zero
           PQty(1) = zero
           PVal (1) = zero
       Fi
       If there are no EPOSS Remittance Out messages
                with matching TxnData.SessionId and
                     EPOSSTransaction.INVI attribute then
```

FUJ00117539


```
        Log an exception                    (No Inventory Products remitted out)
        Coll_Msg.RemCnt(PchCnt) = 1.
        Set Pch_Coll.Pch_Mgs(PchCnt)
          RemNum(1) = <Num:> of LFS Remmittance Out message
          PNo(1) = zero
          PQty(1) = zero
          PVal (1) = zero
      Fi
      Select all messages
             with matching TxnData.SessionId and
                  EPOSSTransaction.INVI attribute
      PCount = zero
      For every Remittance Message Selected
        If EPOSSTransaction.INVI product does NOT exist then
          Coll_Msg.SettleProd(PchCnt) = EPOSSTransaction.ProductNo
          Coll_Msg.SettleValue(PchCnt) = EPOSSTransaction.Debit/100
        Else
          PListPtr = 1
          Pfound = False
          If Unit Price for EPOSSTransaction.INVI.ProductNo != zero
          then
            Until  PListPtr > Pcount or
                     Pfound
              If a Pch_Coll.Pch_Mgs(PchCnt).PNo(PListPtr) =
                           EPOSSTransaction.INVI.ProductNo
              Then
                   Pch_Coll.Pch_Mgs(PchCnt).PQty(PListPtr) =
                           Pch_Coll.Pch_Mgs(PchCnt).PQty(PListPtr)
                           + EPOSSTransaction.INVI.Qty
                   Pch_Coll.Pch_Mgs(PchCnt).PVal(PListPtr) =
                           Pch_Coll.Pch_Mgs(PchCnt).PVal(PListPtr)
                           + EPOSSTransaction.Credit
                   Pfound = True
              Else
                   PlistPtr = PlistPtr + 1
              Fi
            Repeat
          Fi
          If not Pfound then
            Pcount = Pcount + 1
            Pch_Coll.Pch_Mgs(PchCnt).RemNum(PCount) =
                           <Num:> of EPOSS Remittance Out message
            Pch_Coll.Pch_Mgs(PchCnt).PNo(PCount) =
                           EPOSSTransaction.INVI.ProductNo
            Pch_Coll.Pch_Mgs(PchCnt).PQty(PCount) =
                           EPOSSTransaction.INVI.Qty
            Pch_Coll.Pch_Mgs(PchCnt).PVal(PCount) =
                           EPOSSTransaction.Credit
          Fi
        Fi
      Repeat
    Fi
Repeat


Print Pouch Collection Receipt (Print * after barcode if no SessionId present)
```

Note:
The process must be able to cater for empty Collection Groups (Sacks)

FUJ00117539

This will result in each PouchId (barcode) being associated with the Remittance Session Transaction Id if the Remittance Session was completed before the Collection.

Message Generation:

Create the Pouch Messages

```
PmgsCnt = 1
Until PmsgCnt > PchCnt Do
    If Coll_Msg.PchSess(PchCnt) = zero then
      Create an Empty Pouch Message with Coll_Msg.PchId(PmsgCnt)
      Coll_Msg.RemId(PmsgCnt) = <Id:> of Empty Pouch message
      Pch_Coll.Pch_Mgs(PmsgCnt).RemNum(1) =
                                <Num:> of Empty Pouch message
    Fi
    Create a <Data.Items:> container for a Pouch Message
      RemPtr = 1
      Until RemPtr > Coll_Msg.RemCnt(PmsgCnt) Do
                                Create next <Items.Item> container
        RemSessNum  = Pch_Coll.Pch_Mgs(PmsgCnt).RemNum(RemPtr)
        ProductNo = Pch_Coll.Pch_Mgs(PmsgCnt).PNo(RemPtr)
        Qty         = Pch_Coll.Pch_Mgs(PmsgCnt).PQty(RemPtr)
        Value       = Pch_Coll.Pch_Mgs(PmsgCnt).PVal(RemPtr)
        RemPtr    = RemPtr + 1
      Repeat
    Create a Pouch Message with
                  PouchId =    Coll_Msg.PchId(PmsgCnt)
                  RemNodeId =  Coll_Msg.RemId(PmsgCnt)
                  RemItemCnt = Coll_Msg.RemCnt(PmsgCnt)
                  Data.Items container      Already created above
    Coll_Msg.Pmess(PmsgCnt) = <Num:> of Pouch Message
    PmsgCnt = PmsgCnt + 1
Repeat
Create a Collection message with
      PouchItemCnt = PchCnt
      An Items.Item container for each element of Coll_Mgs
        Pouch_Id = Coll_Mgs.PchId
        PouchMessNum = Coll_Msg.Pmess
```

Create the Remittance Messages

Note: New section of code to generate the transactions to rebalance the cash from the 'cash-in-pouches' account.

```
/* New code to reverse transfer to Cash-in-Pouches product and to
settle to RODP settlement account */
PmgsCnt = 1
TotalCashValue = 0
Until PmsgCnt > PchCnt Do
    Create new transaction with ..
         Value = Coll_Msg.SettleProd(PmgsCnt)
         Product = Coll_Msg.SettleValue(PmgsCnt)
         Txn Mode = 'RODP'

    Settle to normal settlement product for this Product/Mode
    combination.  The settlement transaction must have
    EPOSSTransaction.BlackBoxData.PouchId = PouchId

    PmsgCnt = PmsgCnt + 1
```

FUJ00117539

Repeat

Write an empty LFSSack message with an incremented version number.

FUJ00117539

5.3.1.3.2.3 ***Prepare Collection*** (Process 3.1.*?*)

> ***Note: New Process. This will be based loosely on the existing Confirm Pouch Collection function since it replicates much of the user interface.***

The Prepare Collection function will be invoked by a new button.

Counter Staff will confirm ***the inclusion*** of Pouches in ***a Collection Group (Sack)*** by reading the bar codes on each pouch. The bar codes for the pouches in a Collection ***Group*** will be printed on two copies of the Collection note. Both copies will be signed, by the courier and by counter staff ***when the Courier comes to collect the Collection Group (Sack).***

On completion of the Collection Group creation, after all Pouches have been scanned, a Sack Collection message will be updated to include the scanned Pouch Ids.

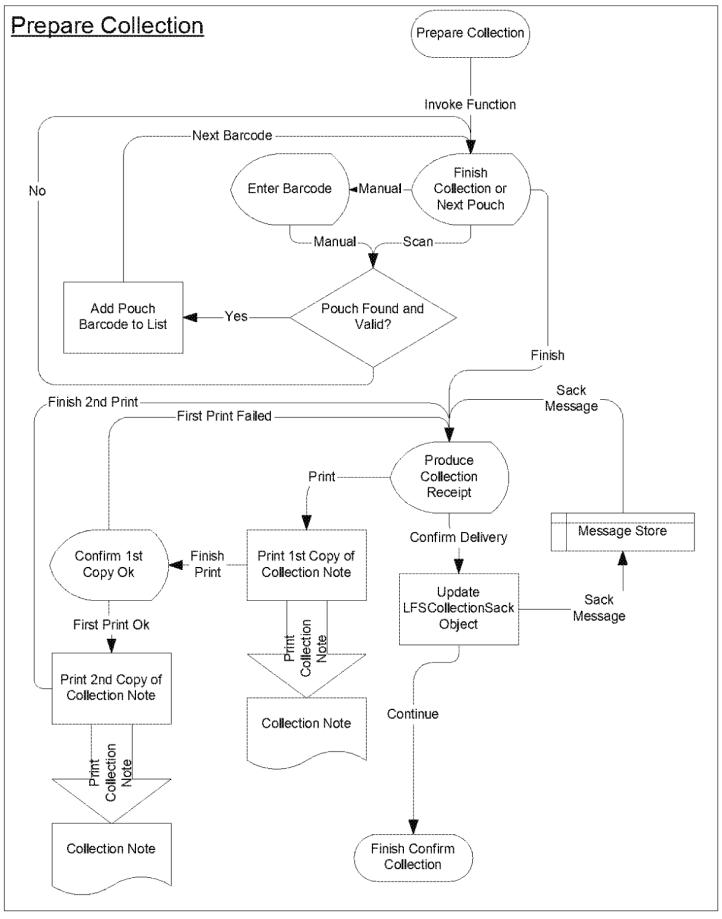

Figure 10 – Data and Process flow for Confirm Collection Group

FUJ00117539

Trigger

- Menu Button (***Prepare Collection***)

Processing

- Set context for reading Pouch bar codes and wait for bar code to be read. The panel should have button to allow manual input of Barcode and a complete button to indicate that the Collection is complete.
- If a Barcode is read using the wand then add the Barcode to the list of pouches and increment the count on the context panel indicating the number of pouches that have been included in the Collection.
- If Manual Input is requested then display calculator panel. Add the entered Barcode to the list of pouches and increment the count on the context panel indicating the number of pouches that have been included in the Collection.
- Bar Code validation should be as specified in [7]. Only Collection Barcodes will be accepted.
- ***Retrieve the LFS Remittance-Out Message (WAIndex.LFSFlag:RO) with a matching PouchId (bar code). If NOT found, then reject Barcode.***
- ***Retrieve the LFS Reverse Pouch Message (WAIndex.LFSFlag:RP) with a matching PouchId (bar code). If found, then reject Barcode.***
- ***Retrieve LFS Collection Sack object 'LFSSack' to ensure that the Pouch Id does not already exist within the Collection Group.***
- If the ***Prepare Collection*** session ended via the complete button without any Pouch Declarations then Display an Error panel with options to return to Barcode scanning or to abandon the session.
  If the session is being abandoned then return to menu without ***updating the*** Collection ***Sack*** message otherwise wait for bar code to be read.
- When the ***Prepare Collection*** session is ended using the complete button then present a Report panel to allow the Collection note to be printed or previewed.
- If Print is selected then print the first copy of the Collection Note and display a panel allowing the user an option to retry the print or continue with the print of the second copy. If retry is selected then return to the Report panel. If the first print is confirmed as Ok then print the second copy and return to the Report panel. The intermediate panel allows the user to take the first copy off the printer before the second copy is printed. ***The Collection Note must contain all Pouches scanned in the current session plus all pouches already prepared for collection (as indicated by the Pouch Ids in the LFSSack message WAIndex.LFSFlag.SA message with highest version number).***

FUJ00117539

- If the complete button is used before the Collection Note has been printed or previewed then display an error panel with options to return to the Report panel or abandon the Collection Pouch session.
- On completion create/update ***the LFSSack message that contains the Pouch Id of all entered and valid Pouches the version number within the message must be incremented.***
- Check that all Barcodes entered are unique before committing the message. There is a possibility that another counter at the Outlet may have been used to enter one or more of the Barcodes while the Barcodes were being scanned and printed. This final check will minimise the possibility of duplicate Barcodes being used within an Outlet.

FUJ00117539

5.3.1.3.2.4 Confirm Remittance Out (Process 3.1.5)

Note: Modified Process

This process will capture sufficient details to allow the contents of a collection pouch to be read by recording SessionId for a remittance out session.

This function will be invoked by the Out-ADC button within the menu 'Transactions/Remittances'. The menu structure for Remit-Out ADC should be modified such that there are only three buttons; Stock, Notes and Coins. Choosing any one will disable the other two thereby inhibiting mixed pouches of stock/cash. The choice of any button will lead to a sub-menu or pick-list so that the remittance-out items may be selected.

The transaction will be performed in mode 'ROSP' for pouches containing cash and mode 'ROAD' for pouches containing Stock.

Following entry of the remittance items, and acceptance of the remittance stack, control will be passed to the LFS function where the Pouch Id must be scanned or manually entered. Validation will ensure that the Pouch type matches the type of stock that is being packed into it.

The EPOSS Transaction will be settled to the normal settlement product(s) as dictated by ModeParameters and by CofAProductModes collections.

An LFS Remit Out message is then written to the Stack.

This stack will be committed to the message store as a single unit and all messages being written as part of the Remit Out transaction will have the same TxnData.SessionId attribute. This is used by the Agent Harvester to retrieve details of remit out session.

FUJ00117539

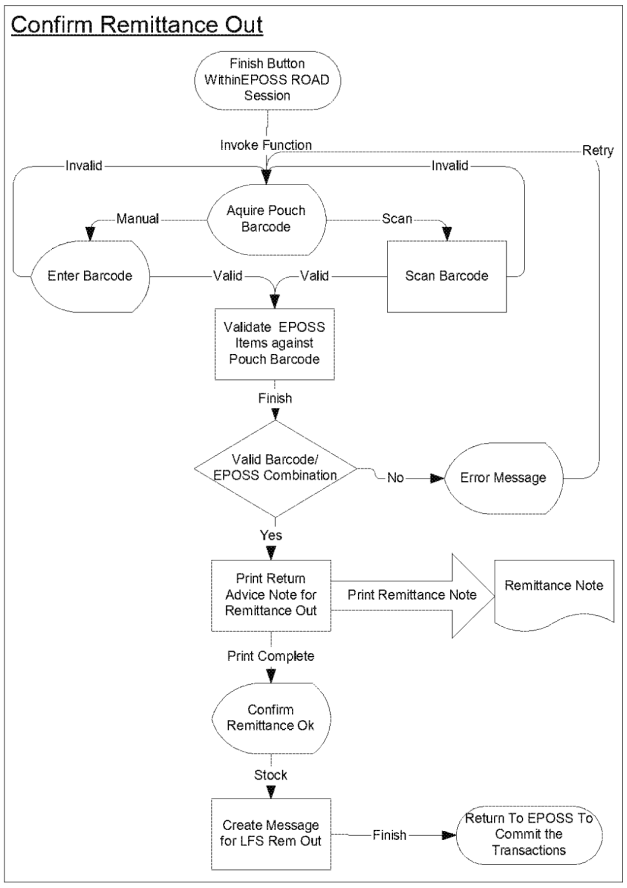

Figure 11 – Data and Process flow for Confirm Remittance Out

Trigger

- ***Invoked by Rem Out ADC Button (and EPOSS call following settlement of the ROAD/ROSP Session).***

Processing

- Request and Enter/Wand in the Pouch bar code
- Bar Code validation should be as specified in [7]. Only Collection Barcodes will be accepted. ***In addition, further checks will ensure that Stock, Notes and Coins are all packed into Pouches that correctly reflect their contents – full validation rules are described in [7]. These rules are also outlined in section 15.22.2 for clarity***.
- The input of a duplicate bar code within the outlet will be rejected. The duplicate validation will not be comprehensive. It simply ensures that no confusion can occur within the Outlet with the existence of duplicate bar codes (pouch Ids). It will be possible to enter via keyboard or wand the same bar code at multiple outlets and to enter the same bar code at an Outlet after the original message has expired and it has been archived

FUJ00117539

- Print Stock Return Note for all Inventory Items on stack.
  The Stock Return Note is included in the pouch with the stock.
  The items on the stack must be printed in the groups as specified via Reference Data. See 15.21.
- Add LFS Remittance Out message to Retail Broker stack. This will be committed to the Message store when the Remittance Out messages are committed. If the Remittance Out process is abandoned then the message will not be written to the message store.
- Return to the settlement process ***that will write the transactions to the message store.***

Note:	***The Remittance Advice slip should be enhanced to include the Pouch Id***

FUJ00117539

5.3.1.3.2.5 Reverse Remittance Out

Note: This is a new LFS function

Once a Pouch has been packed and the Remittance slip printed, the Remittance Slip will be placed in the Pouch and the Pouch will be sealed. If the Branch wishes to modify the contents of the Pouch, then the Pouch must be opened (effectively destroying the pouch).

Modifying pouch contents therefore requires the destruction of the Pouch and the packing of a new Pouch. This function reverses the transactions performed when the Pouch was originally packed thereby 'destroying' the Pouch.

This function will request that the barcode of the Pouch to be destroyed is scanned. If this proves to be a Pouch that has been packed and which has not yet been dispatched then, following confirmation, the transactions performed at the time of packing the Pouch will be reversed.

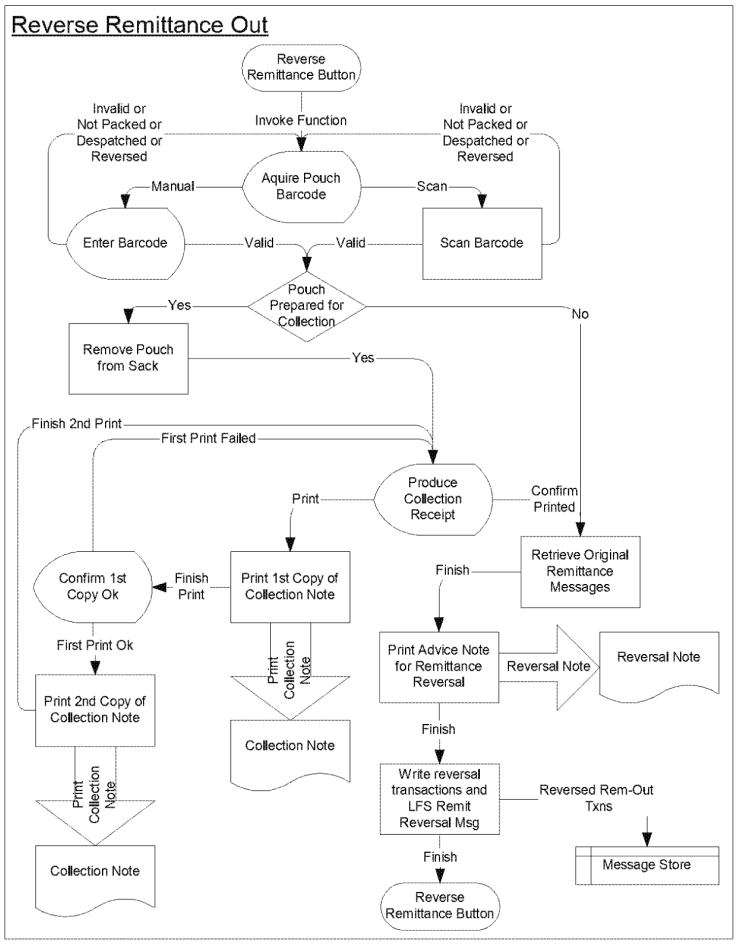

Figure 12 – Reverse Remittance Out

FUJ00117539

Trigger

- *Menu Button (Reverse Remittance-Out)*

Processing

- *Request and Enter/Wand in the Pouch bar code*
- *Bar Code validation should be as specified in [7]. Only Collection Barcodes will be accepted. Only Cash Barcodes will be accepted.*
- *Validate that the barcode exists as a packed Pouch. LFS Message (WAIndex.LFSFlag:RO) with matching PouchId (bar code). Error and return to barcode scan if NOT exists.*
- *Validate that the pouch has not been collected. LFS Message (WAIndex.LFSFlag:PC) with matching PouchId (bar code). Error and return to barcode scan if exists.*
- *Validate that the pouch has not been reversed. LFS Message (WAIndex.LFSFlag:PR) with matching PouchId (bar code). Error and return to barcode scan if exists.*
- *Read LFSSack message (WAIndex.LFSFlag.SA with highest version number).*
- *Check contents of Sack message for PouchId matching the Reversal Pouch Id. If found, present a Report panel to allow the Collection note to be re-printed or previewed. If Print is selected then print the first copy of the Collection Note and display a panel allowing the user an option to retry the print or continue with the print of the second copy. If retry is selected then return to the Report panel. If the first print is confirmed as Ok then print the second copy and return to the Report panel. The intermediate panel allows the user to take the first copy off the printer before the second copy is printed. The Collection Note must contain all Pouches prepared for collection as indicated by LFSSack message (less the pouch being reversed).*
- *Remove the Pouch from the LFS Sack message then increment the Version and write the message back to the message-store.*
- *Print Stock Return Reversal Note for all Inventory Items contained within the original Remittance out transaction (as identified by LFS Remit-Out message). The items on the stack must be printed in the groups as specified via Reference Data. See 15.21*
- *Write reversal transactions to the message-store (exactly the same as the original transactions with the accounting sense reversed and with the Stock Unit changed to be the current Stock Unit). The transaction will be performed in 'RISP' mode. Append a new LFS Reverse Pouch message (WAIndex.LFSFlag:PR) with the same Session Id to indicate that this is a reversal.*

FUJ00117539

5.3.1.3.3 LFS Off-Line Function (Process 3.3)

LFS Off-Line functions will capture data that is required periodically without user intervention. These functions will summarise data already captured by EPOSS transactions without having a major impact on EPOSS functionality or the counter. Two Off-Line processes and one common process are required:

1. Create Daily Cash Statements.
   Overnight Cash declarations are made everyday for all counters that have been used. These will be consolidated into a single cash statement for the Outlet and sent to SAPADS. On days where there is a cash declaration and an ONCH declaration then the latest data will be used to create the Cash Statement.

Note: Changes Required

In addition to the declared values, derived values will be generated for the Total Branch Balance.

SOD References: 6.4.5, 7.4.3, 7.10

FUJ00117539

5.3.1.3.3.1 Create Daily Cash Statement

A Cash statement contains the quantities and total values of each cash item (Denomination) for an Outlet. Since sales information is not recorded at this level, the ONCH and cash declarations will be used. ONCH declarations are made everyday for every counter that has been used on the day. Individual stock units are checked to see if the total value of all the declarations matches the calculated balance of cash. A discrepancy is recorded if the balance is not achieved. The same checking process takes place at Cash declaration stage to get confidence in the amount of cash being reported. The same check is not possible for ONCH or Cash declarations for shared stock Units. The total for all cash declaration of a shared stock unit is checked when the Stock Unit is rolled over every week prior to CAP rollover. For daily Cash Statement purposes, there is an assumption that all declarations for a shared declaration are complete provided there is one declaration for the Stock Unit. The process can only check for a single declaration existing because there is no accounting at counter level within EPOSS. There is no indication of the number of shares that have been used within a shared Stock Unit.

The ONCH and Cash declaration processes are primarily intended to provide a check on the Cash Balance for the Stock Unit and they continue to record the transactions against the Cash Product (ProductNo:1). The denomination product will be recorded in the <INVI:……> container with INVI.ProductNo as the Product number of the denomination and INVI.Qty as the quantity of the Product. EPOSS will calculate the quantity using the value entered for the denomination and the Fixed Price of the product. Accumulating the quantity and value for each denomination for the Outlet will generate Cash Statement.

Processing

- If an Outlet has No Stock Units or there have been no transactions, declarations or rollovers at the Outlet then the Outlet should be ignored. No Cash Statement will be generated until the first transaction has been made at an Outlet.
- Read End of Day markers for yesterday.
  This will allow all searches to be restricted to a smaller number of messages.
- Every Stock Unit that has been used for Transactions during the day must make a cash or ONCH declaration. Outlets will be reported as 'Incomplete ONCH' if any Stock Units that have been used during the day have failed to make a declaration.
  This may occur for several days until the Outlet has to end its CAP. All Stock Units must declare cash as part of the Stock Unit balancing process prior to CAP Rollover. Therefore, a Cash declaration will be forced on the counter that is causing the Incomplete ONCH.
- If a Stock Unit has not been used since the last cash or ONCH declaration then the last valid cash or ONCH declaration for the Stock Unit will be

FUJ00117539

used.
I.e. The latest declaration will be rolled forward for inactive Stock Units.

Note:
Shared Stock Units will be treated the same as Individual Stock Units in that a single declaration for a Shared Stock Unit will be deemed sufficient as a complete declaration for that Stock Unit.

- The latest declaration for each Stock Unit will be used for the Cash Statement. There will be no differentiation between Cash declarations for Stock Unit Rollover and ONCH declarations. The aim is to provide the latest details for each Drawer / Stock Unit. This may result in a mixture of Stock Unit Cash declarations and ONCH declarations being used in a Cash Statement where an ONCH declaration is made after the Stock Unit Cash declaration.

  Note:
  Stock Unit Balance Period rollovers and CAP rollovers have no impact on the production of a cash statement. If cash declarations are made for Stock Unit prior to a rollover then the declaration will be used even if the rollover has not been completed provided it is the latest declaration.

- Declaration Ids for Shared Stock Units will not be checked. Provided there is one declaration of a Stock Unit then the Stock Unit will be accepted for accumulation into the Cash Statement.

- All Declaration Ids used in cash declarations for a Stock Unit during the day will be accumulated to create a single cash declaration for Stock Unit. This assumes that the same Declaration Id will be used for declaring cash before a Stock Unit Rollover and for ONCH Declarations.

  Note:
  This can result in a mixture of Cash Declarations for Stock Unit Rollover and ONCH Declarations being used to create an accumulated cash declaration for a Shared Stock Unit. If different Drawer Ids are used for Cash and ONCH declarations then the overall total used in the cash statement for a share Stock Unit could be doubled.

- The date and time used for the Last ONCH Timestamp will be taken from the latest declaration used for the Cash Statement. This may be a Cash declaration for a Stock Unit Rollover or it may be an ONCH declaration for an Individual Stock Unit or an Identity within a Shared Stock Unit.

- Nil Cash Declarations will be accepted as a valid declaration for a Stock Unit. It may result in a Cash Statement with no details if all declarations at the outlet are nil. In such a case the Cash Statement for the Outlet will be created with a single item where the ProductNo, Qty and Value will be zero and the LastONCHDate and LastONCHTime will be taken from the EPOSS Declaration Trailer message.

- Ensure that duplicate declarations are not included. Each declaration within a Stock Unit is identified by a Declaration Id. EPOSS allows

FUJ00117539

multiple declarations to be made for the same Declaration Id because it may be necessary to use a counter after an ONCH or Cash declaration has been made before End of Day for an ONCH declaration or Stock Unit rollover for a cash declaration. This would require a fresh declaration to record the cash position at End of Day or Stock Unit Rollover respectively. Therefore, only the latest declaration for each declaration Id within a stock unit must be used (See selection of Declaration Id in next process step).

- The SaleValue for Cash declarations is specified in pounds and pence (llll.pp) and it is always negative. It must be converted to a positive number of pence. The value for each item in the message must be in pence.
- Summarise cash or ONCH declarations into a Cash Statement.

```
For Each Stock Unit (SU)
    Read all Cash and ONCH declarations for the day
    If there is no Cash or ONCH declaration for the Stock Unit then
        If Transactions have been created during the day for the SU then
            ONCH_Incomplete = 'Y'
            Create Cash statement message for the Outlet
            End process
        Else
            Find the last ONCH or Cash Declaration for the SU.
            If Any transactions have been created since the Declaration then
                ONCH_Incomplete = 'Y'
                Create Cash statement message for the Outlet
                End process
            Else
                Use the last Declaration for the SU.
            Fi
        Fi
    Fi
    If the latest declaration is ONCH then
        Use ONCH Declarations for SU Cash Statement
    Else
        Use Cash Declarations for SU Cash Statement
    Fi
    Select the latest Declaration Id for the Declaration Type.
    For each declaration within Declaration Id
        If declaration latest for outlet then
            LastONCHDate = Date for declaration message
            LastONCHTime = Time for declaration message
        Fi
        Accumulate Quantity and Value of denomination for each declaration
        into outlet totals
    Repeat
Repeat
```

Note:	***Additional attributes that record the derived cash position for an Outlet need to be populated at E2E Release 1***

An automatically derived Branch cash balance needs to be attached to each Cash Statement passed back to SAPADS. This is the total cash liability of the Branch.

FUJ00117539

The Generated Cash Balance will be calculated by the EOD Branch Transaction Summarisation process described in [17]. The results of the summarisation will be stored as Local Persistent objects Collection CofABalance with Rdata.Data.EODDate equal to today's date. All objects with will be selected and the values for Rdata.Data.Balance will be summed. If no records are found for the current date, then the EPOSS EOD process is assumed to not have run successfully and the Balance will be deemed 'Not Available'.

Create a Cash Statement message for the accumulated figures. ***If the Generated Cash Balance was available, then create the message with attribute Data.GeneratedCash set to the calculated value. Otherwise do not include this attribute in the message***.

Process Schedule

This process should run every day at 19:05 on one counter at each Outlet. The generated Cash Statement will be harvested by the Agent and delivered to SAPADS by 22:00. If the Cash Statement is delayed then it will be in the second delivery before 24:00 to meet the secondary SLA time. There would be no point in generating a Cash Statement after 23:15 because the data could not be sent to SAPADS in time for their process.

5.3.2 EPOSS Changes

5.3.2.1 Declarations (Process 1)

FUJ00117539

5.3.2.2 Remittances (Process 2)

NOTE:
Remittance-Out ADC transactions must not be reversible. A new function described in section 5.3.1.3.2.5 will perform the reversal of the remit-out function. The removal of manual remittance functionality must be soft-launched at the same time as the soft-launch of the remittance functionality.

5.3.2.2.1 Remit in from Auto Distribution Centre (Process 2.1)

This will be a 'Standard' Remittance In function based on a new transaction Mode 'RIAD'. ***Manual Remittance-in will be disabled for the Cash product by the removal of the appropriate button.***

5.3.2.2.2 Remit Out to Auto Distribution Centre (Process 2.2)

FUJ00117539

5.3.2.3 Stock Unit Balancing (process 3)

5.3.2.4 Office CAP (process 4)

5.3.2.5 Counter Scheduler (process 5)

FUJ00117539

5.3.3 Reference Data Service Enhancement

Various changes are required to reference data at the Counters to support the changes described within this document.

All changes to reference data are described in [17] with the exception of the new ACC Card token definition that is described in Section 15.22.3

FUJ00117539

5.3.4 MIS Enhancement

The LFS Host monitors the date and time at which incoming files are received onto the host system from SAPADS. These times are included in messages created by the LFS Agent Loaders as the timestamp when the data was received by LFS. When the messages are replicated to the counter the SLA Agent will be invoked and it will generate a SLA message containing the time the data was received and the time it reached the counter. These messages will be harvested by the Data Warehouse Harvester Agents and passed to the Data Warehouse for subsequent SLA analysis / reporting.

A new type of message, the 'Replenishment Delivery' will be delivered to the counters. This needs to be separately identified with the creation of a new Confirmation Agent 'METRIC' attribute.

New standing data will be required within the Data Warehouse to ensure that the new delivery performance measures are reportable.

It is assumed at this time that the measurement of SLA's surrounding the delivery of the new Replenishment Delivery messages is not outside the bounds of the existing measurement algorithm (ie: no code changes are required to the Data Warehouse).

However, it appears that the bulk of the Replenishment Delivery notices should reach the Counters before start of trading. Following this, further Replenishment Delivery notices may arrive on an hourly basis to be delivered shortly thereafter. If two separate measurements are required:

- ***Bulk delivery with cut-off time and fixed time for delivery***
- ***Interactive-type delivery where message must arrive 'nn' minutes following receipt***

Therefore two separate metrics will need to be used by the LFS Replenishment delivery Loader.

L007 Files received between 16:00:01 and 06:00:00(Cut-Off) will be delivered to the counters by 08:00:00 on the day of Cut-Off

L008 Files received between 06:00:01 and 16:00:00 will be delivered within 2 hours of time of receipt of the file

The times and the metrics used are soft-configured in table RDC_METRIC.

Performance Measure	*Target Measure*	*% Measurement*
L007 – Cut-Off 06:00	*08:00 Day A*	*70%*
	08:00 Day B	*90%*
	08:00 Day C	*95%*
L008 – Cut-Off 0 hours	*2 Hours from Receipt*	*50%*
	4 Hours from Receipt	*60%*
	8 Hours from Receipt	*80%*

FUJ00117539

5.3.5 Migration

The migration implications are considered separately in the Release 1 End-to-End High Level Design document [17]

6 NETWORKING SERVICES

7 PLATFORMS

8 SYSTEM MANAGEMENT

9 APPLICATION DEVELOPMENT

10 SYSTEM QUALITIES

11 SOLUTION IMPLEMENTATION STRATEGY

12 COSTS, RISKS AND TIMESCALES

12.1 COSTS

12.2 RISKS

12.3 TIMESCALES

It is assumed that the changes described within this document will be implemented within the S60 Timescales.

FUJ00117539

13 APPENDIX 1 – LFS SCHEMA

13.1.1 Planned Orders

13.1.2 Advice Notices

13.1.3 Replenishment Delivery Notices

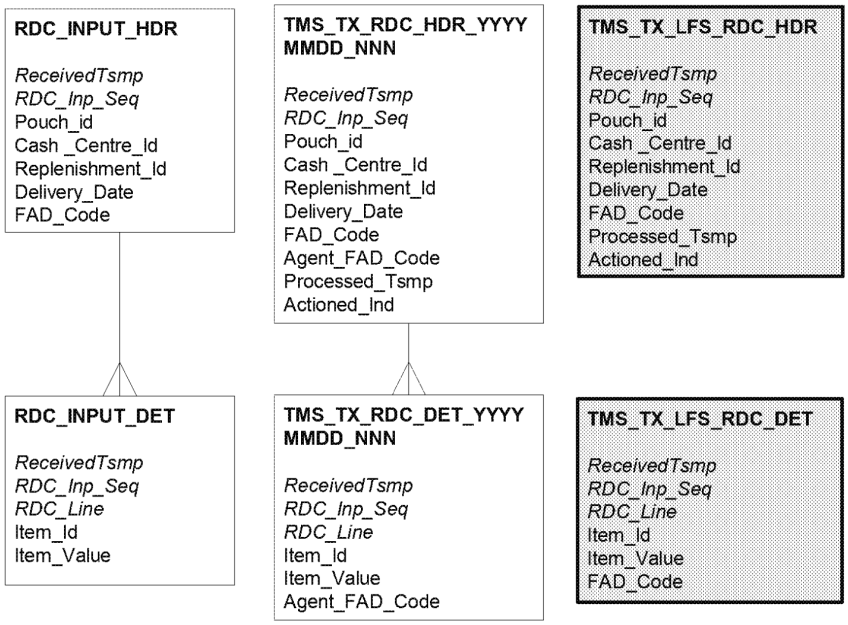

Figure 13 – Replenishment Delivery Notices Tables/Views

FUJ00117539

13.1.4 Cash Statements

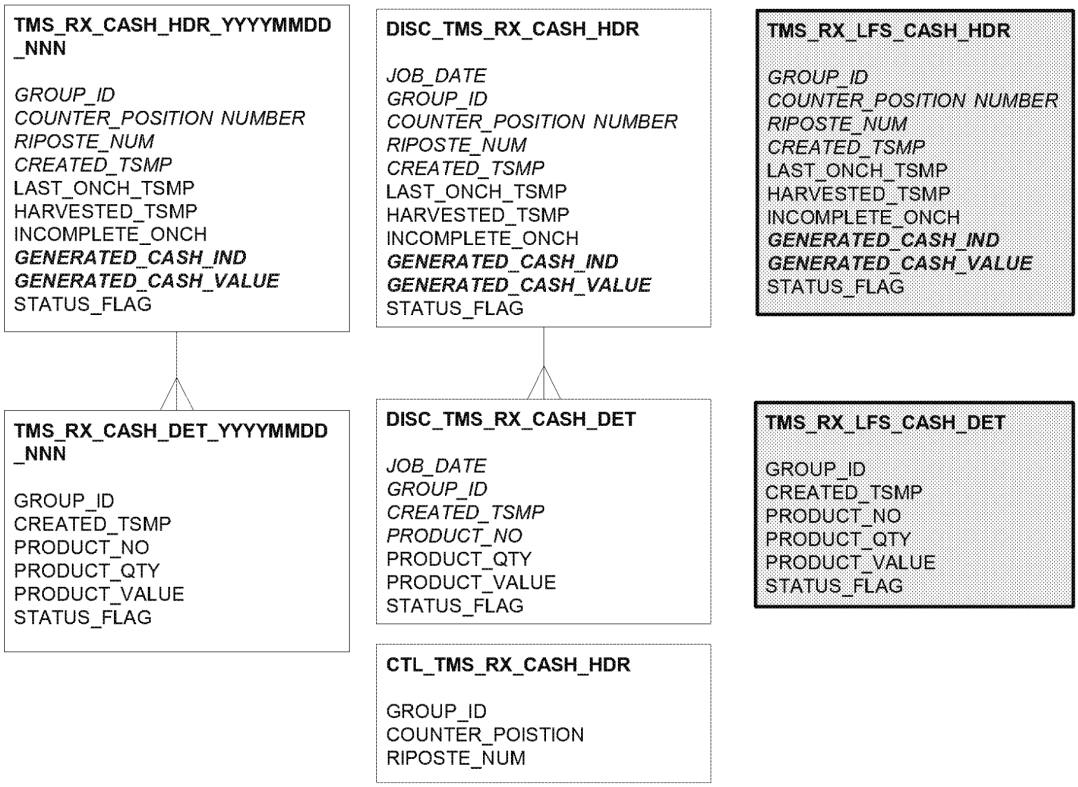

Figure 14 – Cash Statement Tables/Views

FUJ00117539

13.1.5 Stock Statements

13.1.6 Pouch Deliveries

13.1.7 Pouch Collections

13.1.8 MIS

13.1.9 Outlet Details

13.1.10 Control and System Tables

13.1.11 Audit and Archive Tables

13.1.12 Agent Tables

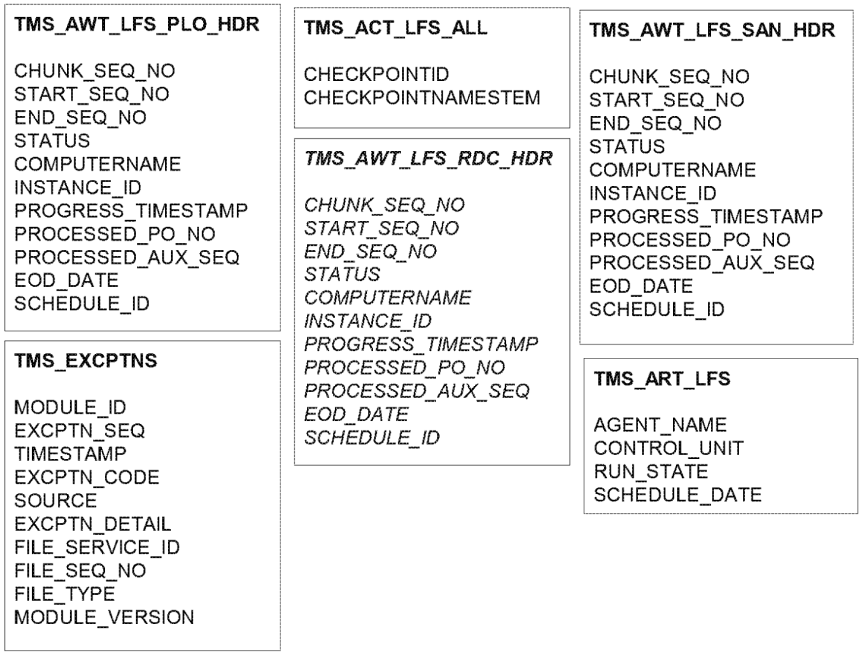

Figure 15 - Audit and Archive Tables

FUJ00117539

13.2 LFS TABLES

13.2.1 APPLICATION_PARAMETERS

13.2.2 ARCHIVED_TABLES

13.2.3 ARCHIVE_EVENTS

13.2.4 CTL_TMS_RX_CASH_HDR

13.2.5 CTL_TMS_RX_PDEL_HDR

13.2.6 CTL_TMS_RX_STOCK_HDR

13.2.7 DISC_TMS_RX_CASH_DET

13.2.8 DISC_TMS_RX_CASH_HDR

Table Type: Heap (normal)

Copies of duplicate header rows for Cash Statement messages that are discarded. The LFS Harvester can harvest the same message several times in recovery and fail-over scenarios. The host processes detect the duplicate rows and transfer second and subsequent row with the same message identifier to the discard table. This table is archived and tidied.

Column	Type	Length	Nulls	Description
JOB_DATE	DATE	7	N	
GROUP_ID	NUMBER	6,0	N	
COUNTER_POSITION	NUMBER	2,0	N	
RIPOSTE_NUM	NUMBER	10,0	N	
CREATED_TSMP	DATE	7	N	
LAST_ONCH_TSMP	DATE	7	Y	
HARVESTED_TSMP	DATE	7	N	
INCOMPLETE_ONCH	VARCHAR2	1	Y	
GENERATED_CASH_IND	***VARCHAR2***	***1***	***Y***	
GENERATED_CASH_VALUE	***NUMBER***	***12,0***	***Y***	
STATUS_FLAG	VARCHAR2	4	Y	

FUJ00117539

13.2.9 DISC_TMS_RX_PCOL_DET

13.2.10 DISC_TMS_RX_PCOL_HDR

13.2.11 DISC_TMS_RX_PDEL_DET

13.2.12 DISC_TMS_RX_PDEL_HDR

13.2.13 DISC_TMS_RX_STOCK_DET

13.2.14 DISC_TMS_RX_STOCK_HDR

13.2.15 EXCEPTION_CODES

13.2.16 EXCPTNS

13.2.17 FILE_AUDIT_TRAILS

13.2.18 FILE_REGISTRY

13.2.19 JOB_CONTROL

13.2.20 LFS_PRODUCT_TRANSLATIONS

13.2.21 OUTLET_DET_19990101_NNN

13.2.22 PLO_INPUT_DET

13.2.23 PLO_INPUT_HDR

13.2.24 PROCESS

13.2.25 PROCESS_AUDIT_TRAILS

13.2.26 PROCESS_CONTROL

13.2.27 RDC_INPUT_DET

Table Type: Heap (normal)

This table has one row for every line of Replenishment Delivery Detail record. The key is derived from the owning RDC_INPUT_HDR table. Uniqueness is guaranteed by the addition of RDC_Line that is generated by LFS. This ensures record uniqueness even if there is a duplicate ITEM_ID within any single Pouch.

Column	Type	Length	Nulls	Description
RECEIVED_TSMP	DATE	7	N	Prime Key From owning RDC_INPUT_HDR row
RDC_INP_SEQ	NUMBER	5,0	N	Prime Key From owning RDC_INPUT_HDR row
RDC_LINE	NUMBER	4,0	N	Prime Key Sequence Number, starting at 1,within

FUJ00117539

				each Received_tsmp and rdc_inp_seq, and incremented by 1 for each Item_Id
ITEM_ID	NUMBER	10	N	identifies an inventory item within a pouch.
ITEM_VALUE	NUMBER	11	N	Indicates the value of the inventory item in the Pouch. The value is in pence (£*100).

Volumes:

Approximately 15 rows per Replenishment Delivery in input file
(Note: Maximum = 1000 rows per Replenishment Delivery)

Average data size = 11,000 Replenishment Delivery x 15 rows x 37 bytes = 6.5Mb

13.2.28 RDC_INPUT_HDR

Table Type: Heap (normal)

This table has one row for every Replenishment Delivery Pouch. The Coins or Notes associated with the Replenishment Delivery Pouch are in the RDC_INPUT_DET table

Theoretically, we could use the SAPADS generated Pouch_Id as a unique key for our tables. In practice this would tie us to their processes and risks for the management of the data item. A safer option has been taken to generate unique keys for LFS tables within the confines of the system.

RDC_Inp_Seq is required because there is no guarantee that duplicate Pouch_Ids for the same office will not be in a single file.

Column	Type	Length	Nulls	Description
RECEIVED_TSMP	DATE	7	N	Prime Key Date/Time the file of Replenishment Delivery Notices were received.
RDC_INP_SEQ	NUMBER	5,0	N	Prime Key Sequence Number, starting at 1,within RECEIVED_TSMP incremented by 1 for each Replenishment Delivery Header
POUCH_ID	NUMBER	12	N	SAPADS generated unique Pouch Id.
CASH_CENTRE_ID	VARCHAR2	3	N	Cash Centre from where the cash is sourced.
REPLENISHMENT_ID	NUMBER	8	N	The unique identifier of a replenishment delivery.
DELIVERY_DATE	DATE		N	Date when the replenishment is to be delivered to the Branch.
FAD_CODE	VARCHAR2	7	N	Format 999999x where x is a numeric digit0 to 9 or 'X'.

Volumes:

Approximately 11,000 rows per input file processed (64,000 rows per week)

Average data size = 11,000 x 42 bytes = Less than 1Mb

13.2.29 RDC_METRIC

Table Type: Heap (normal)

This table contains static information related to the 'Metric' that is used for the measurement of performance of delivery of Replenishment delivery Notices to counter positions. The Metric to be used is dependent on the time of delivery of the 'RDC' file to the LFS Host.

This table contains a time-range (measured in number of seconds since midnight) and an associated 'Metric'. The time of delivery of the RDC file to the LFS host is compared with the time-range and the associated 'Metric' is placed in the TMS_TX_RDC_HDR_YYYYMMDD_NNN table for Agent consumption.

The time ranges are inclusive and must be consecutive without gaps.

FUJ00117539

Column	Type	Length	Nulls	Description
START_TIME	NUMBER	5	N	Number of seconds from midnight that the associated Metric becomes valid
END_TIME	NUMBER	5	N	Number of seconds from midnight that the associated Metric stops becoming valid
METRIC	VARCHAR2	4	N	Performance Measure metric.

Volumes:

Approximately 3 rows

13.2.30 SAN_INPUT_DET

13.2.31 SAN_INPUT_HDR

13.2.32 SENT_MIS_DATA_YYYYMMDD_NNN

13.2.33 TABLE_REGISTRY

13.2.34 TMS_ACT_LFS_ALL

13.2.35 TMS_ART_LFS

13.2.36 TMS_AWT_LFS_PLO_HDR

13.2.37 TMS_AWT_LFS_RDC_HDR

Table Type: Heap (normal)

Table used by the Agent Replenishment Delivery loader to multistream the process.

Column	Type	Length	Nulls	Description
CHUNK_SEQ_NO	NUMBER	10,0	N	
START_SEQ_NO	VARCHAR2	7	Y	
END_SEQ_NO	VARCHAR2	7	Y	
STATUS	VARCHAR2	1	N	
COMPUTERNAME	VARCHAR2	15	Y	
INSTANCE_ID	VARCHAR2	3	Y	
PROGRESS_TIMESTAMP	DATE	7	Y	
PROCESSED_PO_NO	VARCHAR2	7	Y	
PROCESSED_AUX_SEQ	VARCHAR2	20	Y	
EOD_DATE	DATE	7	Y	Maestro schedule's business day
SCHEDULE_ID	NUMBER	5,0	Y	Generated by agent_schedule_work to distinguish between different Maestro schedules for an agent on a business day

Sequences for table TMS AWT LFS RDC HDR

Sequence	Parameters
Tms_Seq_Awt_Lfs_Rdc_Hdr	Increment by 1 Cache Size 20

FUJ00117539

13.2.38 TMS_AWT_LFS_SAN_HDR

13.2.39 TMS_EXCPTNS

13.2.40 TMS_RX_CASH_DET_YYYYMMDD_NNN

13.2.41 TMS_RX_CASH_HDR_YYYYMMDD_NNN

Table Type: Heap (normal)

This table is used to create an entry for every Outlet that has supplied a Cash Statement. It provides the header information for each Daily cash statement record in the file returned to SAPADS.

One of these tables will be created when the off-load starts. The table will be dated using LFS_Business_Date and the will be numbered sequentially 001, 002, ….. within LFS_Business_Date. Agent Harvester will use a view, TMS_RX_CASH_HDR to access the table. This will provide a fixed interface for the Agent processes.

Column	Type	Length	Nulls	Description
GROUP_ID	NUMBER	6,0	N	Prime Key FAD Code for the Outlet.
COUNTER_POSITION	NUMBER	2,0	N	Counter Identifier within Outlet (FAD Code).
RIPOSTE_NUM	NUMBER	10,0	N	Message number for Counter.
CREATED_TSMP	DATE	7	N	Prime Key Date and time the Daily Cash Statement Declaration was created at the counter.
LAST_ONCH_TSMP	DATE	7	Y	Date and Time of Last ONCH declaration used for Cash Statement.
HARVESTED_TSMP	DATE	7	N	Date and Time the Daily Cash Statementwas harvested
INCOMPLETE_ONCH	VARCHAR2	1	Y	'Y' or 'N', 'Y' = All ONCH declarations for Outlet not available at end of day.
GENERATED_CASH_IND	***VARCHAR2***	***1***	***Y***	***Indicating whether the LFS EOD process was able to determine the Branch Cash Balance***
GENERATED_CASH_VALUE	***NUMBER***	***12,0***	***Y***	***The automatically calculated Branch Cash Balance - this may be null if EPOSS EOD failed to execute.***
STATUS_FLAG	VARCHAR2	4	Y	Indicates the status of the row for duplicate / recovery processing.

Table used in following views/procedures

Type	
VIEW	TMS_RX_LFS_CASH_HDR

Note:

One row per Daily Cash Statement identifying the Outlet and Data created/harvested details. The Incomplete ONCH indicator reports that the Outlets Daily Cash Statement could not be produced because all ONCH declarations had not been made by end of day. There is no indication of Outlets that have failed to create Daily Cash Statements before the file was created.

Volumes:

Max Data size = 20,000 x 30 bytes = Less than 1 Mb

FUJ00117539

13.2.42 TMS_RX_PCOL_DET_YYYYMMDD_NNN

13.2.43 TMS_RX_PCOL_HDR_YYYYMMDD_NNN

13.2.44 TMS_RX_PDEL_DET_YYYYMMDD_NNN

13.2.45 TMS_RX_PDEL_HDR_YYYYMMDD_NNN

13.2.46 TMS_RX_STOCK_DET_YYYYMMDD_NNN

13.2.47 TMS_RX_STOCK_HDR_YYYYMMDD_NNN

13.2.48 TMS_TX_PLO_DET_YYYYMMDD_NNN

13.2.49 TMS_TX_PLO_HDR_YYYYMMDD_NNN

13.2.50 TMS_TX_RDC_DET_YYYYMMDD_NNN

Table Type: Heap (normal)

This table is used by the LFS Agent, Load Replenishment Delivery Notices, to create the of Replenishment Delivery messages for Counters. The YYYYMMDD_NNN suffix is generated from the Job_Date and Job_Seq_No columns of Job_Control table. They represent the date and file number of the input file.

Column	Type	Length	Nulls	Description
RECEIVED_TSMP	DATE	7	N	Prime Key From owning RDC_INPUT_HDR row.
RDC_INP_SEQ	NUMBER	5,0	N	Prime Key From owning RDC_INPUT_HDR row.
RDC_LINE	NUMBER	4,0	N	Prime Key Sequence Number, starting at 1,within each Received_tsmp & RDC_INP_SEQ, and incremented by 1 for each ITEM_Id. RDC_LINE from RDC_INPUT_DET.
ITEM_ID	NUMBER	10	N	identifies an inventory item within a pouch (from RDC_INPUT_DET).
ITEM_VALUE	NUMBER	11	N	Indicates the value of the inventory item in the Pouch. The value is in pence (£*100). (from RDC_INPUT_DET).
AGENT_FAD_CODE	CHAR	7	N	First six characters of FAD_Code from Replenishment Delivery Header. Denormalised from owning RDC_INPUT_HDR row.

Table used in following views/procedures

Type	
VIEW	TMS_TX_LFS_RDC_DET

Volumes:

As per RDC_INPUT_DET because there are expected to be very few rejections.

Average data size = 11,000 Replenishment delivery Notices x 15 rows x 44 bytes = 7.5Mb

13.2.51 TMS_TX_RDC_HDR_YYYYMMDD_NNN

Table Type: Heap (normal)

This table is used by the LFS Agent, Load Replenishment Delivery Notices, to create the header attributes of Replenishment Delivery messages for Counters. The YYYYMMDD_NNN suffix is

FUJ00117539

generated from the Job_Date and Job_Seq_No columns of Job_Control table. They represent the date and file number of the input file.

Column	Type	Length	Nulls	Description
RECEIVED_TSMP	DATE	7	N	Prime Key Date/Time the file of Replenishment delivery notices were received. RECEIVED_TSMP from RDC_INPUT_HDR
RDC_INP_SEQ	NUMBER	5,0	N	Prime Key Sequence Number, starting at 1,within RECEIVED_TSMP incremented by 1 for each Replenishment Delivery Header. RDC_INP_SEQ from RDC_INPUT_HDR.
POUCH_ID	NUMBER	12	N	SAPADS generated unique Pouch Id. POUCH_ID from RDC_INPUT_HDR
CASH_CENTRE_ID	VARCHAR2	3	N	Cash Centre from where the cash is sourced. CASH_CENTRE_ID from RDC_INPUT_HDR
REPLENISHMENT_ID	NUMBER	8	N	The unique identifier of a replenishment delivery. REPLENISHMENT_ID from RDC_INPUT_HDR
DELIVERY_DATE	DATE		N	Date when the replenishment is to be delivered to the Branch. DELIVERY_DATE from RDC_INPUT_HDR
FAD_CODE	VARCHAR2	7	N	Format 999999x where x is a numeric digit0 to 9 or 'X' FAD_CODE from RDC_INPUT_HDR
AGENT_FAD_CODE	CHAR	7	N	First six characters of FAD_Code from Replenishment Delivery Header.
PROCESSED_TSMP	DATE	7	Y	Date/Time when the Message was created. This will be the SYSDATE when the chunk containing the row was retrieved for processing by the Agent Loader.
ACTIONED_IND	VARCHAR2	1	Y	Status for row used by Agent process. N = New (Not Processed) Null = Processed Ok. F = Processed Failed
METRIC	VARCHAR2	4	N	Performance Measure metric.

Table used in following views/procedures

Type	
VIEW	TMS_TX_LFS_RDC_HDR

Note:

The failed rows which are set with Actioned_Ind = 'F' are not handled by any automated process. They represent an exception that will have been raised by the Agent process via Tivoli. The Indicator allow the row(s) to be selected easily during investigation.

Volumes:

As per RDC_INPUT_HDR because there are expected to be very few rejections.

Average data size = 11,000 x 64 bytes = Less than 1Mb

13.2.52 TMS_TX_SAN_DET_YYYYMMDD_NNN

13.2.53 TMS_TX_SAN_HDR_YYYYMMDD_NNN

FUJ00117539

13.3 INITIAL LOAD.

The following tables will be populated during database build.

13.3.1 Application Parameters

Name	Type	Value	Description
R1_SAPADS_MIGRATE	*T*	*OFF*	*Indicating whether we have migrated to the E2E Release 1 SAP ADS Cash Statement Interface format. Initially set OFF (Not Migrated)*
RDC_FILE_AGE	*N*	*3*	*Reject a replenishment delivery file if it is more than 3 days old*

13.3.2 Job_Control

A dummy record is created for every Job Type as a primer for establishing the completion of the previous job. These records are never deleted and would be used if all Jobs are 'Tidied'. I.e. The initial situation could occur during live running if all Jobs of one or more types are tidied. These permanent 'Dummy' Job control records would allow new jobs to start.

The following rows are created:

JOB_SEQ	JOB_NAME
21	*RDC*
22	*RDC_EOD*

13.3.3 Archived_Tables

13.3.4 Table_Registry

A row in the Table_Registry table with a date of 19990101000000 for the following tables:

Table_Name	Table_Seq	Table_Status
Tms_Tx_RDC_Det	*001*	*COMPTE*
Tms_Tx_RDC_Hdr	*001*	*COMPTE*

These entries will set the database to a running state. The processes that receive data from SAPADS will see the previous jobs as complete and the processes that are harvesting / sending data will have tables that can be used on day one.

13.3.5 RDC_Metric

This table determines which metric will be used dependent on the time off arrival of the RDC file.

FUJ00117539

Start_Time	End_Time	Metric	Description
1	*21600*	*L007*	*Between 00:00:01 and 06:00:00*
21601	*57600*	*L008*	*Between 06:00:01 and 16:00:00*
57601	*86400*	*L007*	*Between 16:00:01 and 24:00:00*

13.3.6 LFS_PRODUCT_TRANSLATIONS

14 APPENDIX 2 – LFS FILES

All files processed within LFS are defined within the Application Interface Specification [8]. The following sections define the processing / validation rules for all files and records.

14.1 PLANNED ORDERS FILE

14.2 REJECTED PLANNED ORDERS FILE.

14.3 ADVICE NOTICES FILE

14.4 REJECTED ADVICE NOTICES FILE.

14.5 REPLENISHMENT DELIVERY NOTICES FILE

There will normally be sixteen Replenishment Delivery files per day (and eight on Sunday) The RDC EOD Function will police this (see section 5.3.1.1.3.5).

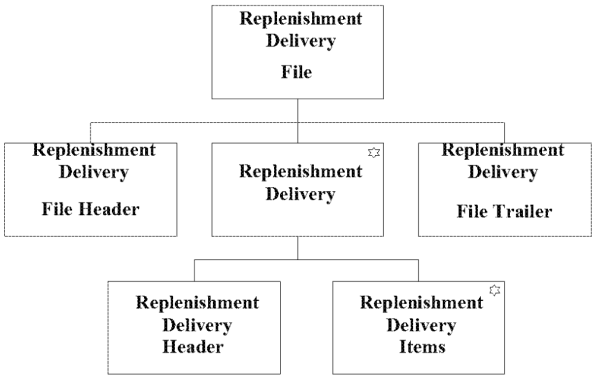

Figure 16 - Structure of Replenishment Delivery File

FUJ00117539

File size expected to be 4.2Mb

14.5.1 Replenishment Delivery File Header record

Field	*Format*	*Description / Rules*
Record_Id	*Char(5)*	*= 'RDFH1'*
Source	*Char(6)*	*= 'SAPADS'*
FileName	*Varchar(32)*	*Must be the same as the terminal name of the Physical file.* *Format of the name: saccyymmddsss.rdc* *Where ccyymmdd is the date for the file and sss is the sequence number within the day starting at 001.* *File name will be in lower case. Example sa19990501001.rdc*

One record per file to verify that the correct file is being processed.

14.5.2 Replenishment Delivery Header record

Field	*Format*	*Description / Rules*
Record_Id	*Char(5)*	*= 'RDCR1'*
FAD_Code	*Char(7)*	*Format 999999x where x is a numeric digit 0 to 9 or 'X'*
Replenishment_Delivery_Id	*Number(8)*	*Uniquely identifies a Replenishment Delivery from a Cash Centre. Not Validated*
Delivery_Date	*Date*	*Date of Replenishment Delivery – validated only as being a valid date format. No range validation required.*
Pouch_Id	*Number(12)*	*No validation performed*
Cash_Centre_Id	*Char(3)*	*No validation performed*

One or more Replenishment Delivery headers per file.

One record per Replenishment Delivery Pouch. There will be 64,000 Replenishment Delivery Notices per week with an average of 11,000 Replenishment Delivery Notices per day.

Volumes:

Approximately 11,000 records per day processed (64,000 per week)

Maximum data size = 11,000 x 43 bytes = Less than 1Mb

14.5.3 Replenishment Delivery Detail record

Field	*Format*	*Description / Rules*
Record_Id	*Char(5)*	*= 'RDDR1'* *If the record type cannot be recognised then it will be assumed to be an invalid detail line.*
Item_Id	*Number(10)*	*Material Id of Cash Denomination*
Item_Value	*Number(11)*	*Value of cash denomination (in pence)*

FUJ00117539

One to 1000 records per Replenishment Delivery Note. The AIS [8] states that there is a maximum of 15 items. The average number of items is also 15. No validation will be performed on this and there will be no limits enforced.

Volumes:

Approximately 15 records per Replenishment Delivery Notice
Average data size = 11,000 x 15 x 26 = 4Mb

14.5.4 Replenishment Delivery File Trailer record

Field	Format	Description / Rules
Record_Id	Char(5)	= 'RDFT1'
Record Count	Number (7)	Number of records in the file inclusive of the File header and trailer records. With leading zeros.

Last record in each file to verify the complete file has been processed.

14.6 REJECTED REPLENISHMENT DELIVERY FILE.

A Rejected Replenishment Delivery file will only be created if one or more Replenishment Delivery Notices are rejected because the target Outlet is not open or temporarily closed. If no Replenishment Delivery Notices are rejected then no file will be sent to SAPADS.

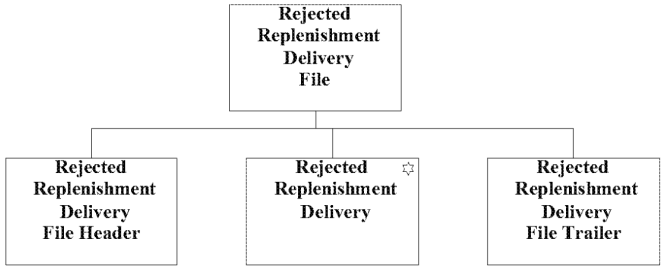

Figure 17 - Structure of Rejected Replenishment Delivery File

The expectancy is that there will be very few rejections for each input file. There may be a few Outlets (less than 100) that may be in different states on SAPADS and Horizon. Therefore, the file requirements for this file will be modest at less than 1Mb per file.

14.6.1 Rejected Replenishment Delivery File Header record

Field	Format	Description / Rules
Record_Id	Char(5)	= 'RDFH1'
Source	Char(7)	= 'PATHWAY'
FileName	Char(32)	Must be the same as the terminal name of the Physical file. Format of the name: lfccyymmddsss.rdc

FUJ00117539

Field	Format	Description / Rules
		Where ccyymmdd is the date for the file and sss is the sequence number within the day starting at 001.

One record per file to verify that the correct file is being processed.

14.6.2 Rejected Replenishment Delivery Contents Header record

Field	Format	Description / Rules
Record_Id	Char(5)	= 'RDHR1'
Replenishment_Delivery_Id	Number(8)	Uniquely identifies a Replenishment Delivery from a Cash Centre. Not Validated
FAD_Code	Char(7)	Format 999999x where x is a numeric digit 0 to 9 or 'X'
Reject_Code	Number(1)	Set depending on value of Office details for rejected Replenishment Delivery Header rows Code why the Line was rejected 1. FAD Code does not exist. No Office Row selected. 2. Outlet Permanently Closed Outlet_Dets.Office_Status = 2 3. Outlet not Automated Outlet_Dets.Office_Status = 3 4. Outlet Temporarily Closed Outlet_Dets.Office_Status = 4

One record per selected Planned Order Header representing each rejected Planned Order.

14.6.3 Rejected Replenishment Delivery File Trailer record

Field	Format	Description / Rules
Record_Id	Char(5)	= 'RDFT1'
Record Count	Number (7)	Number of records in the file inclusive of the File header and trailer records. With leading zeros.

Last record in each file to verify the complete file has been processed.

FUJ00117539

14.7 DAILY CASH STATEMENT FILE

14.7.1 Daily Cash Statement File Header record

14.7.2 Daily Cash Statement Header record

Field	Format	Description / Rules
Record_Id	Char(5)	= 'DCHR1'
FAD_Code	Char(7)	FAD_Code from Outlet_Det for the Group_Id on TMS_RX_LFS_CASH_HDR.
Created_Tsmp	Date/Time	Created_Tsmp from TMS_RX_LFS_CASH_HDR Date and Time when the Daily Cash Statement was created
Last_ONCH_Tsmp	Date/Time	Last_ONCH_Tsmp from TMS_RX_LFS_CASH_HDR Date and Time when the Daily Cash statement was harvested Spaces if Null_statement_Ind is 'Y'
Null_statement_Ind	Char(1)	Incomplete_ONCH from TMS_RX_LFS_CASH_HDR 'Y' or 'N' 'Y' = Incomplete ONCH at EOD, No Cash Statement 'N' = Cash statement available for Outlet
Generated_Cash_Ind	***Char(1)***	***Indicating whether the calculated value of the branch cash balance is available ('Y' or 'N')*** ***'Y' = Generated Figure exists*** ***'N' = Generated Figure does not exist***
Generated_Cash_Value	***Number(11)***	***Numeric value of the Branch generated cash balance. This will be blank if Generated_Cash_Ind = 'N'***

FUJ00117539

14.8 WEEKLY STOCK STATEMENT FILE

14.9 POUCH DELIVERY FILE

14.10 POUCH COLLECTION FILE

14.11 MIS FILES

14.12 CONTROL FILE

14.13 HIGH WATER MARK FILE

15 APPENDIX 3 – LFS MESSAGES

15.1 PLANNED ORDERS

15.2 PLANNED ORDERS READ

15.3 ADVICE NOTICES

15.4 ADVICE NOTICES READ

15.5 POUCH DELIVERY

Pouch Deliveries will be created as messages when the delivery is confirmed. Pouch Delivery messages will be identified by the attributes *Application:LFS* and *Data.TransType:PouchDelivery*.

Messages that are more than the Riposte limit in length will be separated into a message and an attachment. The *Data.Items* container will be separated from the message and it will be stored as an attachment. The remaining message will always fit within the Riposte message length limits.

```
<Application:LFS>
<WAIndex:
  <LFSFlag:PD>
>
<Data:
```

FUJ00117539


```
<TranType:PouchDelivery>
<DeliveryId:aabbbbbb>
<TransactionDate:DD-Mon-YYYY>
<TransactionTime:hh:mm:ss>
<Items:
   <Item:
      <PouchId:Barcode>
      <Manual:ManualIndicator>
   >
   <Item:
      ..........
   >
>
>
```

Where:

aabbbbbb ..... aa = Counter (Node)
bbbbbb = Session (Message Number)
Both Counter and session must be padded with zeros to produce a 8 digit number. If the session is more than 6 digits then it should be trimmed from the left.
Examples:
Counter 1, Session 123 _____ 01000123.
Counter 12, Session 1234567 __ 12234567

TransactionDate and TransactionTime
These attributes are only created if date and time are entered for a recovery transaction.

Barcode..... Barcode read by wand or supplied manually.

ManualIndicator..... ***This is either Y or N and indicates whether the value of a Cash Pouch was entered manually or derived from a Replenishment Delivery Notice. This attribute is only present for Cash Pouches.***

15.6POUCH COLLECTION

A Pouch Collection can result in three message types being created:

- Collection Message.
  One of these messages will be created for each Pouch Collection. Items for the Collection Message will identify each of the Pouch Messages associated with the Pouch Collection.
- Pouch Message.
  One of these will be created for each pouch that is collected within the Pouch Collection. Each Pouch Message will have items for each Remittance Transaction that was associated with the pouch. Where there is no Remittance Session for the pouch then a dummy item will be created with Product Number, Quantity and Value set to zero ***(Cash Pouches will always have associated Remittances).***
- Empty Pouch Message ***(Stock Pouches Only).***
  These messages will be created when there is no Remittance session for the pouch

FUJ00117539

when it is collected. It records the collection of an empty pouch and it provides a unique key for the dummy item created in the pouch message.

15.6.1 Collection Message

Collection messages will be created when a collection is confirmed. Pouch Collection messages will be identified by the attributes *Application:LFS* and *Data.TranType:PouchCollection*. The Collection message will always be created after all Pouch messages for the collection have been created. This will ensure that the Pouch messages are replicated to the correspondence server before the Collection message.

Messages that are more than the Riposte limit in length will be separated into a message and an attachment. The *Data.Items* container will be separated from the message and it will be stored as an attachment. The remaining message will always fit within the Riposte message length limits.

```
<Application:LFS>
<WAIndex:
   <LFSFlag:PC>
>
<Data:
   <TranType:PouchCollection>
   <CollectionId:aabbbbbb>
   <TransactionDate:DD-Mon-YYYY>
   <TransactionTime:hh:mm:ss>
   <PouchItemCnt:PouchCnt>
   <ACC:ACCCard>
   <Items:
      <Item:
         <PouchId:Barcode>
         <PouchMessNum:PouchTxnId>
      >
      <Item:
         ..........
      >
   >
>
```

Where:

aabbbbbb ..... aa = Counter (Node)
bbbbbb = Session (Message Number)
Both Counter and session must be padded with zeros to produce a 8 digit number. If the session is more than 6 digits then it should be trimmed from the left.
Examples:
Counter 1, Session 123 _____ 01000123.
Counter 12, Session 1234567 __ 12234567

DD-Mon-YYYY and *hh:mm:ss*
These attributes are only created if Transaction Date and Transaction Time are entered for a recovery transaction.

FUJ00117539

PouchCnt..... Count of the number of Pouches in the session.
I.e. Number of <Item:> containers within the <Items:> container.
The count must be greater than zero.

Items.Item..... This container will be repeated for each Pouch in the collection session.

ACCCard*..... *This is the barcode number of the ACC Card that was scanned during the collection. If the Branch does not require ACC Card scanning, then this value will be hard-coded to value 'N/A'

Barcode..... Barcode read by wand or supplied manually.

PouchTxnId..... Value of <Num:> container for the associated Pouch Message.

15.6.2 Pouch Message

15.6.3 Empty Pouch Message

No Changes required to this message.

Note: Empty Pouch Messages can no longer be produced for Cash Pouches.

15.7 REMITTANCE OUT (FOR POUCH CONTENTS)

15.8 CASH MARKER

15.9 CASH STATEMENT

Note: Additional attributes are required to this section to record the Generated Cash Value.

This message is generated by the LFS End of Day process every Day after 19:00. It represents the total holding of Cash denomination products at the Outlets. It is generated summarising the Cash Marker messages.

Cash Statement messages are harvested by the LFS Interactive Agent and sent to SAPADS.

```
<Application:LFS>
<WAIndex:
   <LFSFlag:CS>
>
<Data:
   <TranType:CashStatement>
   <IncompleteONCH:ONCH_Ind>
   <GeneratedCash:GeneratedValue>
   <LastONCHDate:DD-Mon-YYYY>
   <LastONCHTime:hh:mm:ss>
```

FUJ00117539


```
<Items:
   <Item:
      <ProductNo:CashDenomination>
      <Qty:nnnnnnn>
      <Value:mmmmmmmmmmm>
   >
   <Item:
      ..........
   >
>
>
```

If IncompleteONCH is set to Y then only TranType and IncompleteONCH attributes are created within the <Data> container.

Where:

ONCH_Ind..... Y for one or more ONCH declaration outstanding
N for All ONCH declarations available.

GeneratedValue ..... The value of cash held at the Branch at the end of the trading day. This consists of the Brought-Fwd value from yesterday for all Cash Products plus all the transactions within the Trading day for the same Cash Products. Value in pence.

LastONCHDate and LastONCHTime
Date and Time attributes of the latest ONCH declaration message used.

CashDenomination..... Product Number for the cash denomination.
EPOSSTransaction.INVI.ProductNo from ONCH and Cash Declarations.

nnnnnnn ..... Sum of all Quantities for the denomination.
EPOSSTransaction.INVI.Qty from ONCH and Cash Declarations.

mmmmmmmmmmm ..... Sum of all values in pence for the denomination.
EPOSSTransaction.SaleValue from ONCH and Cash Declarations.
Note:
The SaleValue on Cash declarations is a negative value in Pounds.pence. It must be converted to a positive number in pence. I.e. SaleValue * -100.

FUJ00117539

15.10 STOCK MARKER

15.11 STOCK STATEMENT

15.12 ONCH AND CASH DECLARATION

15.13 STAMP DECLARATION

15.14 ONCH DECLARATION TRAILER

15.15 CASH AND STAMP DECLARATION TRAILER

15.16 STOCK UNITS

15.17 EPOSS PRODUCTS OBJECT

15.18 VALUE STOCK DETAILS

15.19 NON-VALUE STOCK DECLARATION

15.20 NON-VALUE STOCK CONFIRMATION

15.21 GROUP LISTS

15.22 BARCODE EVENT

The following Collection and IO Events are used to drive the Riposte Peripheral broker when reading LFS Barcodes.

The current definitions do not support the processing of the Barcode Check digit directly from the peripheral broker. The check digit is checked by the LFS Application via a separate check digit module. ***In addition, the following validation will be carried-out by the LFS Application on the Pouch Barcodes:***

Carrier: ***Must always be value '3'***

FUJ00117539

ItemType: *The two digits of the ItemType will be validated in conjunction with the SerialNoType as follows:*

Pouch Type	*ItemType plus SerialNoType Value*
Note Pouches	*012 – 019* *021* *031*
Stock Pouches	*252 – 259* *271* *281*
Coins	*992* *571*

SerialNoType: *Validated as follows:*

'1' *Pouch being sent from Branch to Cash Centre*

2-9 *Pouch being sent from Cash Centre to Branch*

CheckDigit: *Fully described in [7].*

The EPOSS Remittance-out function (or menu hierarchy) will ensure that Stock, Notes and coins cannot be mixed within an EPOSS Session. The LFS Remittance-out function will ensure that the transaction items within the EPOSS Stack match the type of Pouch that the Items are being packed into.

It should noted that, Branches that do not use an ACC Card-carrying courier (as defined by the Presence of the ACC Non-core product) will pack their Notes into Coin Pouches. The presence of the ACC Product therefore indicates that the value of 0 (zero) as the first character of the ItemType is prohibited.

The ACC Barcode will also be validated by the LFS Application. This will ensure that he Check-Digit is valid and that the month/year of expiry is greater than or equal to the current date.

All validation of the barcodes will be performed within the LFS Code.

15.22.1 Pouch Delivery / Collection Barcodes

The following IOEvent and Collection are used to capture Barcodes for Pouch Delivery and Pouch Collection functions.

Riposte Collection:

```
<Collection:_LFSPouchInputEvents>
<ObjectName:LFSPouch>
<RData:
  <Data:
    <SupportedModes:
      <LFSConfirm:1>
```

FUJ00117539


```
        >
        <Validation:
          <LFSPouch:
            <Length:12>
            <Carrier:
              <Start:1>
              <Length:1>
              <Pic:N(1)>
            >
            <ItemType:
              <Start:2>
              <Length:2>
              <Pic:N(2)>
            >
            <SerialNoType:
              <Start:4>
              <Length:1>
              <Pic:N(1)>
            >
            <SerialNo:
              <Start:5>
              <Length:7>
              <Pic:N(7)>
            >
            <CheckDigit:
              <Start:12>
              <Length:1>
              <Pic:N(1)>
            >
          >
        >
        <LFSBarcode:
          <Include:
            <Carrier:>
            <ItemType:>
            <SerialNoType:>
            <SerialNo:>
            <CheckDigit:>
          >
        >
      >
    >
```

Riposte IOEvent:

```
    <IOEvent:
      <Name:LFSPouch>
      <DeviceType:BarcodeReader>
      <Date:
        <SupportedModes:
          <LFSConfirm:1>
        >
      >
    >
```

15.22.2 Remittance Out

The following IOEvent and Collection are used to capture Barcodes for Pouch at the end of a Remittance Out to Automated Distribution Centre.

Riposte Collection:

```
    <Collection:_LFSPouchInputEvents>
    <ObjectName:LFSPouchRead_01>
    <RData:
```

FUJ00117539


```
<Data:
   <SupportedModes:
      <ROAD:1>
   >
   <Validation:
      <LFSPouch:
         <Length:12>
         <Carrier:
            <Start:1>
            <Length:1>
            <Pic:N(1)>
         >
         <ItemType:
            <Start:2>
            <Length:2>
            <Pic:N(2)>
         >
         <SerialNoType:
            <Start:4>
            <Length:1>
            <Pic:N(1)>
         >
         <SerialNo:
            <Start:5>
            <Length:7>
            <Pic:N(7)>
         >
         <CheckDigit:
            <Start:12>
            <Length:1>
            <Pic:N(1)>
         >
      >
   >
   <LFSBarcode:
      <Include:
         <Carrier:>
         <ItemType:>
         <SerialNoType:>
         <SerialNo:>
         <CheckDigit:>
      >
   >
>
```

Riposte IOEvent:

```
<IOEvent:
   <Name:LFSPouchRead>
   <DeviceType:BarcodeReader>
   <Date:
      <SupportedModes:
         <ROAD:1>
      >
   >
>
```

15.22.3 Authorised Collectors Card

The following IOEvent and Collection are used to capture Barcodes for Pouch at the end of a Remittance Out to Automated Distribution Centre.

FUJ00117539

Riposte Collection:

```
<Collection:_LFSPouchInputEvents>
<ObjectName:LFSACCRead_01>
<RData:
   <Data:
      <SupportedModes:
         <HK:1>
      >
      <Validation:
         <LFSACC:
            <Length:16>
            <Carrier:
               <Start:1>
               <Length:2>
               <Pic:N(2)>
            >
            <Depot:
               <Start:3>
               <Length:4>
               <Pic:N(4)>
            >
            <SerialNo:
               <Start:7>
               <Length:5>
               <Pic:N(5)>
            >
            <ExpiryMonth:
               <Start:12>
               <Length:2>
               <Pic:N(2)>
            >
            <ExpiryYear:
               <Start:14>
               <Length:2>
               <Pic:N(2)>
            >
            <CheckDigit:
               <Start:16>
               <Length:1>
               <Pic:N(1)>
            >
         >
      >
      <LFSBarcode:
         <Include:
            <Carrier:>
            <Depot:>
            <SerialNo:>
            <ExpiryMonth:>
            <ExpiryYear:>
            <CheckDigit:>
         >
      >
   >
>
```

Riposte IOEvent:

```
<IOEvent:
   <Name:LFSACCRead>
```

FUJ00117539


```
<DeviceType:BarcodeReader>
<Date:
   <SupportedModes:
      <HK:1>
   >
>
```

15.23 REPLENISHMENT DELIVERY

Note:	***This is a new message definition. Messages will be generated by a new Agent loader process and delivered to individual branches for use during the Pouch Delivery function.***

A replenishment delivery notice is stored as a message identified by the attributes Application:LFS, WAIndex.LFSFlag:RD and DataTransType:LFSReplenishmentDelivery. The expiry date will be fixed to 10 days.

```
<Application:LFS>
<WAIndex:
   <LFSFlag:RD>
>
<Data:
   <TranType:LFSReplenishmentDelivery>
   <ReceivedDate:Received_Date>
   <PouchId:Pouch_Id>
   <CashCentreId:Cash_Centre_Id>
   <ReplenishmentId:Replenishment_Delivery_Id>
   <DeliveryDate:Delivery_Date>
   <Items:
      <Item:
            <ProductNo:Inventory_ProductId>
            <Value:Inventory_Value>
      >
      <Item:
            ……..
      >
   >
>
```

Where:

Received_Date..... Date element of the Date/time when the message was received from SAPADS.

Pouch_Id..... The Pouch Id received from SAPADS

Cash_Centre_Id..... The Cash Centre Id received from SAPADS.

Replenishment_Delivery_id

..... The Replenishment Delivery Id supplied by SAPADS

Delivery Date..... The date of expected delivery of the Pouch as supplied by SAPADS

FUJ00117539

Inventory_Product_Id .. The SAPADS Inventory Id for the cash product as supplied by SAPADS

Inventory Value ..... The total value of cash product as determined by Inventory Product Id – in pence

15.24 POUCH REVERSAL

Note: This is a new message definition. This message acts as an audit to indicate that the remittance-out of a pouch has been reversed.

A Reverse Pouch message is identified by the attributes Application:LFS, WAIndex.LFSFlag:PR. The expiry date will be fixed to 35 days.

```
<TxnData:
   <SessionId:Reverse_Session>
   <TxnId:Transaction_Id>
   <Container:Stock_Unit>
   <Mode:Transaction_Mode>
      >
<Application:LFS>
<WAIndex:
   <LFSFlag:PR>
>
<Data:
   <TranType:LFSPouchReverse>
   <PouchId:Pouch_Id>
>
```

Reverse_Session..... Session Identifier that ties all messages for the Reverse Pouch Session committed to the message store. It is allocated by Riposte. When the contents of the Remittance stack are committed, Riposte ensures that all messages are written to the message store or none of the messages are written.
Although the format of the session identifier reflects the message number of the first message committed LFS cannot rely on this because it may be changed without notice. LFS can only rely on the Session Identifier remaining constant for all messages in the session.

Transaction_Id ..... Identity of the transaction allocated by Riposte.

Stock_Unit..... Identity of the SU.

Transaction_Mode..... This is the same transaction mode as the original Remittance-out transaction that is being reversed.

Pouch_Id..... Barcode read by wand or supplied manually.

FUJ00117539

15.25 COLLECTION SACK

This LFS Object is written/updated by the Confirm Collection Group function and is updated by the Reverse Remittance Out function (sections 5.3.1.3.2.3 and 5.3.1.3.2.5).

The message is identified by the attributes Application:LFS, WAIndex.LFSFlag:SA. The expiry date will be fixed to 35 days.

```
<Application:LFS>
<WAIndex:
   <LFSFlag:SA>
>
<Data:
   <VersionNo:Version>
   <Id:Pouch_Id>
>
```

Version ..... This is initially set to '1'. The contents of the Collection Group (Sack) may be modified by the Prepare Collection and Reverse Remittance-Out function. Each time the contents are updated, the version is incremented.

Pouch_id ..... A list of Pouch Ids that are included in the Sack.

16 APPENDIX 4 – ATTRIBUTE MAPPING

The following tables map the database tables on the Host database to the Riposte messages.

16.1 PLANNED ORDERS

16.2 ADVICE NOTICES

16.3 REPLENISHMENT DELIVERY MESSAGE

Replenishment Delivery messages are generated using TMS_TX_LFS_RDC_HDR and TMS_TX_LFS_RDC_DET views.

Columns	*Attributes*	*Comments*
	Application	*= 'LFS'* *Message Id = LFS Message*

FUJ00117539

Columns	*Attributes*	*Comments*
	WAIndex.LFSFlag	*= 'RD'* *Index attribute for access to all LFS messages.* *'RD' allows Replenishment Delivery messages to be selected using the Index.*
	Data.TranType	*= 'LFSReplenishmentDelivery'* *LFS Message Type = Replenishment Delivery*
TMS_TX_LFS_RDC_HDR. Received_Tsmp	*Data.ReceivedDate*	*Date element of Received date timestamp in DD-Mon-YYYY format.*
TMS_TX_LFS_RDC_HDR. Pouch_Id	*Data.PouchId*	
TMS_TX_LFS_RDC_HDR. Cash_Centre_Id	*Data. CashCentreId*	*Not Used*
TMS_TX_LFS_RDC_HDR. Replenishment_Id	*Data. Replenishment_Id*	*Not Used*
TMS_TX_LFS_RDC_HDR. Delivery_Date	*Data.DeliveryDate*	*Not Used*
TMS_TX_LFS_RDC_DET. Item_Id	*Data.Items.Item. Product_no*	*Items is a repeating group within Item due to the 1…many relationship between TMS_TX_LFS_RDC_HDR and TMS_TX_LFS_RDC_DET*
TMS_TX_LFS_RDC_DET. Item_Value	*Data.Items.Item. Value*	
	Riposte Attachment	*If the overall size of the message exceeds 2Kb then the contents of DATA.Body container will be converted to an attachment and attached to the remaining message.* *The format of the data will be the same as the Body container excluding the container name.* *I.E. :<1:TextText……><2:TextText……>……*

Message Sizing:

Header = 121 bytes

Pouch details = 48 bytes

Average Pouch Details per Pouch = 12

Therefore average message size = 121+(12*48) = 700 bytes

Expected max Messages = 1100, therefore overall capacity = 7.7 Mb

FUJ00117539

16.4POUCH DELIVERY

16.5POUCH COLLECTION

16.6CASH STATEMENT MESSAGE

Cash Statement messages are harvested into tables with views of TMS_RX_LFS_CASH_HDR and TMS_RX_LFS_CASH_DET for TMS_RX_CASH_HDR_YYYYMMDD and TMS_RX_CASH_DET_YYYYMMDD tables respectively. Each message will create one row in the header table with a view of TMS_RX_LFS_CASH_HDR. The detail table with a view of TMS_RX_LFS_CASH_DET has one row for each item in the message unless the IncompleteONCH attribute is set to 'Y' when no rows are created.

16.6.1 TMS_RX_LFS_CASH_HDR

Columns	Attributes	Comments
Group_Id	GroupId	GroupId representing the FAD Code for the Outlet available on all messages
Counter_Position	Id	Id representing the Node or Counter within the Outlet from the Collection messages.
Riposte_Num	Num	Num representing the message number within the Node from the Collection messages.
Created_Tsmp	Date Time	Standard Message details concatenated and converted to Oracle Date.
Last_ONCH_Tsmp	LastONCHDate LastONCHTime	Date and time to be converted to Oracle Date. Optional, Null if Incomplete_ONCH = 'Y' Date/Time when the last ONCH declaration was made at the outlet. Note: This is irrelevant because there is no point before the Outlet 'End of Day' when the declarations can be considered as complete. ONCH declarations can be overridden by a subsequent declaration at any time before Outlet 'End of Day'.
Harvested_Tsmp		Sysdate
Incomplete_ONCH	IncompleteONCH	*ONCH_Ind* from message Values 'Y' or 'N'

FUJ00117539

Columns	Attributes	Comments
Generated_Cash_Ind		*Y or N. Set to 'Y' if the attribute GeneratedCash is present on the Cash Statement message.*
Generated_Cash_Value	*GeneratedCash*	*This attribute may not be present on all messages. Set column to NULL if the attribute is missing.*
Status_Flag		Null

One row per Cash Statement Message.

16.6.2 TMS_RX_LFS_CASH_DET

16.7 STOCK STATEMENT MESSAGE

17 APPENDIX 5 – MAESTRO SCHEDULE

17.1 LFS START OF DAY

17.2 LFS DAEMON

17.3 PLANNED ORDERS

17.4 PLANNED ORDERS END OF DAY

17.5 ADVICE NOTICES

17.6 ADVICE NOTICES END OF DAY

17.7 REPLENISHMENT DELIVERY NOTICES

This process will be scheduled to start after the LFS Start of Day has finished and the LFS Daemon has been started.

The process will run the following jobs:

1. ***Start Replenishment Delivery Notices (LFSC400)***
2. ***Accept Replenishment Delivery Notices (LFSC401)***

FUJ00117539

3. *Load Replenishment Delivery Headers (LFSC402)*
4. *Load Replenishment Delivery Trailers (LFSC403)*
5. *Validate Replenishment Delivery Notices (LFSC404)*
6. *Send Rejected Replenishment Delivery Notices (LFSC405)*
7. *Agent Replenishment Delivery Loader (Maestro Schedule LFS_RDCAGT)*
8. *Check Replenishment Delivery Notices Load (LFSC406)*
9. *End Replenishment Delivery Notices (LFSC099 'RDC')*

Job 1 will check if there is a file waiting to be processed and skip to Job 9 if a file is not found by cancelling the intermediate jobs.

The LFS Daemon process will create a list of all files that have been delivered from SAP ADS (file types 'SAN', 'PLO' and 'RDC') that have not yet been processed. A new Maestro daemon will periodically poll this list to determine whether it contains any files with an extension of '.rdc'. If there are, then Maestro will invoke an instance of the Replenishment Delivery Notices schedule. Following completion of the Replenishment Delivery Notices schedule and whilst there remain files with an extension of '.rdc' within the pending list, Maestro will re-execute the Replenishment Delivery Notices schedule each 90 seconds.

The list of outstanding SAP ADS files is generated by the LFS Daemon each time the LFS Daemon checks the $LFS_FILES/FTMS/RECEIVING directory for newly arrived files (currently set to occur each 60 seconds). The list of outstanding files is deduced by selecting rows from the FILE_REGISTRY table where JOB_SEQ is NULL. The filename of each outstanding file will be written to a new file $LFS_FILES/SAPADS/LFSSchedule.txt. The format of this file is as follows:

saccyymmddsss.plo
saccyymmddsss.san
saccyymmddsss.rdc
T:lines:sum

Where the first 3 lines represent filenames outstanding for processing and the final line is a trailer record. When there are no files outstanding for processing, then the file will only contain a trailer record. Therefore the content of the file is a list of filenames and this is terminated by a single trailer record. The format of the filenames is as follows:

"saccyymmddsss.typ" where

sa =	*source of file, namely 'sa' for SAPADS*
ccyymmdd =	*date*
sss =	*file sequence number starting at 001 on each transaction day*
typ =	*'plo' = Planned Orders* *'san' = Advice Notices* *'rdc' = Replenishment delivery notices*

Filenames in the list will be ordered by date/time of receipt of the file.

FUJ00117539

The format of the trailer record is as follows:

"T:lines:sum:" where

T =	*Constant value representing the trailer record*
Lines =	*The number of lines in the file excluding the trailer record.*
Sum =	*The sum of the integer ASCII values of all characters in the file excluding the characters in the trailer record. This sum will include the linefeed character that immediately precedes the trailer record.*

The new scheduling mechanism will be used for PLO and SAN files as well as RDC files. This then provides a common and demand-driven dynamic schedulic mecanism across all inbound data streams.

17.8 REPLENISHMENT DELIVERY END OF DAY

This process will be scheduled to start after the last Replenishment Delivery processes have finished. The FTMS acknowledgement of Rejected file(s) should have been received within an hour.

The process will run the following jobs:

1. *Start Replenishment Delivery Notices End of Day (LFSC407)*
2. *Process Replenishment Delivery Notices Acknowledgements (LFSC408)*
3. *End Replenishment Delivery End of Day (LFSC099 'RDC_EOD')*

FUJ00117539

17.9 CASH STATEMENTS

17.10 CASH STATEMENTS END OF DAY

17.11 STOCK STATEMENTS

17.12 STOCK STATEMENTS END OF DAY

17.13 POUCH DELIVERIES

17.14 POUCH DELIVERIES END OF DAY

17.15 POUCH COLLECTIONS

17.16 POUCH COLLECTIONS END OF DAY

17.17 MIS

17.18 END OF DAY

17.19 TIDY APPLICATION TABLES

17.20 TIDY CONTROL TABLES

17.21 ARCHIVE TABLES

17.22 DATABASE BACKUP

18 APPENDIX 5 – DB INTEGRITY

FUJ00117539

19 APPENDIX 6 – OUTSTANDING ISSUES

A full list of outstanding issues is documented in the associated Design Proposal (Reference [15]).