Document Title: Impact Release 3 - Counter Design for Declaration, Correction and Revaluation

Document Type: *High Level Design*

Release: *S80*

Abstract: This document describes the high-level design for the Horizon Counter Application Declaration and Transaction Correction for IMPACT Release 3.

Document Status: APPROVED

Originator & Dept: *Roger Donato SIDU*

Contributors: *Martin Nixon SIDU, Gareth Jenkins, Pete Jobson*

Internal Distribution:

External Distribution:

Approval Authorities:

Name	Position	Signature	Date
Andy Kennedy	SIDU Development Manager		
Gareth Jenkins	ASS Design Authority		

0.0 Document Control

0.1 Document History

Version No.	Date	Reason for Issue	Associated CP/PinICL
0.1	26/07/2004	First Draft	CP3716
0.2	17/09/2004	Second Draft, incorporating feedback from review of 1st draft. Adjustment of mandatory reviewers/approvers as per PA/PRO/010.	
0.3	21/10/2004	Incorporation of feedback from review of 2nd draft, plus rework to include support for a new definition of the variances report	
0.4	28/10/2004	Incorporation of feedback from informal review of 3rd draft	
0.5	09/11/2004	Minor corrections from informal review of 4th draft	
1.0	12/11/2004	Approved version	
1.1	30/11/2004	Updated to include development feedback	
1.2	06/05/2005	Suppression of Cash Variance Report as a result of CP 3980. Corrections arising from implementation and test	CP3980, Various PEAKs (see section 0.5)
2.0	08/09/2005	Originator changed to Roger Donato (*Martin Nixon the document author has moved to a different project*) No technical changes since version 1.2	

0.2 Review Details

Review Comments by :	*N/A*
Review Comments to :	*Originator*

Mandatory Review Authority	Name
RASD Design Authority	Gareth Jenkins
SIDU Design Authority	Pete Jobson
SIDU Development Manager	Andy Kennedy
CS Introduction Manager	Reg Barton
CS Operations Manager	Mik Peach
CS Security Manager	Bill Mitchell
ITU Test Design	Janusz Hollender
ITU Test Team Leader	Neil Gormley
Optional Review / Issued for Information	
SIDU Design	Phil Hemingway, Roger Donato, Duncan MacDonald, Rex Dixon, Chris Bailey, Peter Ashdown
SIDU Development	Mark Scardifield, Walter Wright, Martin McConnell, Phil Orton, Mike Coon, Roger York, Gerald Barnes, Rob Dinnadge, Jon Hulme
SIDU Test	Harjinder Hothi, Miriam Bell

ITU	Peter Robinson, Linda Miller
Programmes	Bill Reynolds

0.3 Associated Documents

Reference	Version	Date	Title	Source
PA/TEM/001			Fujitsu Services Document Template	PVCS
PA/PRO/010			Post Office Account Document Management & Control Process	PVCS
DE/PRO/003			ICL Pathway Development Directorate Processes	PVCS
NB/STD/001			Counter VB Coding Standards	PVCS
EA/CDE/002	1.0		Branch Trading Reporting, Management and Control and Transaction Management Conceptual Design	POL via PVCS
EA/DPR/004	1.1		IMPACT Release 3 Design Proposal	PVCS
CP 3842			IMPACT Release 3: Counter Changes	PVCS
PC0081427			PEAK Incident: FAD014614 - ONCH prompt for SU BM did not come up	PEAK
PC0091605			PEAK Incident: Cash Declaration event messages – no drawer id	PEAK
EA/HLD/005			Impact Release 3 - Counter Design for Balancing, Rollover and Stock Processing	PVCS
AD/DES/041			TPS Agents High Level Design	PVCS
EA/HLD/007			TPS POL FS Summarisation HLD	PVCS
EA/HLD/008			IMPACT Release 3 Migration High Level Design	PVCS
EA/HLD/009			TPS HR SAP Summarisation & Transaction Corrections HLD	PVCS
EA/HLD/010			IMPACT Release 3: Agents High Level Design	PVCS
EA/IFS/002			Transaction Corrections AIS	PVCS
EA/IFS/011			IMPACT Release 3 - Report Production User Interface	PVCS
EA/IFS/012			IMPACT Release 3 - Declaration, Correction and Revaluation User Interface	PVCS
EA/IFS/013			IMPACT Release 3 - Balancing and Trading Statement User Interface	PVCS
EA/LLD/004			Counter Balancing Functional Low Level Design	PVCS

Reference	Version	Date	Title	Source
EA/LLD/005			In Day Transactions Low Level Design	PVCS
EA/LLD/006			Impact Release 3 – Trading Period Rollovers Low Level Design	PVCS
EA/LLD/007			Maintain Office Variances Low Level Design	PVCS
EA/LLD/008			Impact Release 3 Transaction Correction Application	PVCS
EA/LLD/015			Impact Release 3 Receipts & Reports	PVCS
EA/LLD/019			Data Protection Low Level Design	PVCS
EP/DES/002			EPOSS Attribute Grammar Catalogue	PVCS
EP/DES/025			EPOSS End of Day Service HLD	PVCS
EP/LLD/012			EPOSSCore Low Level Design	PVCS
EP/LLD/013			EPOSSDeclare Low Level Design	PVCS
EP/LLD/016			EPOSS Watchdog Low Level Design	PVCS
RD/DES/056			Reference Data End to End High Level Design for S80 (Impact, Track & Trace, +1 Sales)	PVCS
RD/DES/059			S80 Impact – Reference Data Requirement to Support Functional Changes and to control Migration	PVCS
SD/DES/005			Horizon OPS Reports and Receipts – Pathway Horizon Office Platform Service	PVCS
SD/SPE/016			Horizon OPS Menu Hierarchy	PVCS
SD/SPE/022			Horizon OPS Menu Hierarchy: Changes Supplement	PVCS
SD/STD/001			Pathway Horizon Office Platform Service Style Guide	PVCS

Unless a specific version is referred to above, reference should be made to the current approved versions of the documents.

0.4 Terms and Abbreviations

Term/Abbreviation	Definition
AG	Attribute Grammar.
Add/Remove Cash	This means declaring the addition of cash to (or removal of excess cash from) a Stock Unit in order to rectify a locally detected 'variance'. It should be distinguished from a 'Make Good' transaction that adjusts the value of a Stock Unit in order to rectify a previous transaction error that satisfied local balancing checks, but was subsequently detected centrally by POL FS Note that the user interface (see EA/IFS/012) currently describes both kinds of action as 'Make Good', but to avoid ambiguity throughout this document, the term 'Add/Remove Cash' is used instead where appropriate.
API	Application Programming Interface.
BP	Balance Period
BSTR	COM data type representing strings as 16 bit characters
CAP	Cash Account Period
Discrepancy	A mismatch between the declared value of actual cash, stamps or value-stock holdings for a stock unit, and the system derived figures based on opening figures for the balancing period and the record of transactions that have occurred since. On balancing a stock unit (or even producing a trial balance), any mismatch is committed by converting it into a loss or surplus 'discrepancy' product, which then appears on balance reports. Note that the 'Discrepancies' report on the Stock Balancing menu reports on uncommitted discrepancies, whereas the Stock Balance report describes committed ones.
DLL	Dynamic Link Library. A unit of executable code.
DP	Design Proposal
EOD	End Of Day.
EPOSS	Sub-system of Horizon desktop which is responsible for representing sales of goods
GUI	Graphical User Interface. In the context of this document, interaction with the user via the Counter Desktop.
HLD	High-level design specification. See DE/PRO/003.
ISDN	Integrated Services Data Network.
ISO	International Standards Organisation.
LFS	Logistics Feeder Service.
LLD	Low-level design specification. See DE/PRO/003

Term/Abbreviation	Definition
Make Good	A 'Make Good' transaction adjusts the value of a Stock Unit in order to rectify a previous transaction error that satisfied local balancing checks, but was subsequently detected centrally by POL FS It should be distinguished from 'Add/Remove Cash', which is the act of declaring the addition of cash to (or removal of excess cash from) a Stock Unit in order to rectify a locally detected 'variance'. Note that the user interface (see EA/IFS/012, SD/SPE/016) might describe both kinds of action as 'Make Good', but to avoid ambiguity throughout this document, the term 'Add/Remove Cash' is used instead where appropriate.
NT	Microsoft operating system
OBC	Operational Business Change. A mechanism for POL to request day to day changes to data within the Horizon system.
ONCH	Overnight Cash Holding. A Cash Declaration originally intended to support LFS and SAPADS in provisioning the branch with denominated cash
OPS	Outlet Processing System.
POL	Post Office Ltd.
PPD	Procedures and Processes Document. See DE/PRO/003.
PVCS	Configuration Management tool used by Fujitsu Services (Post Office Account)
RDMC	Reference Data Management Centre
RAD	Rapid Application Development
SAPADS	SAP Advanced Distribution System. The cash management system (in terms of stocks of cash as opposed to accounting) run by Post Office Ltd.
SCO	Single Counter Outlet, i.e. an outlet that has only one counter PC.
SLA	Service Level Agreement. An agreement, usually encapsulated in a contract, specifying the ways in which the delivery of a service will be measured and the level of such measures that must be achieved.
SU	Stock Unit
Till	A notional subcontainer within a shared Stock Unit that corresponds to a particular cash, stamps or stock declaration id. Since transactions are only recorded against a Stock Unit, any discrepancy calculation between declared holdings and system derived figures must be done against the sum of all the till declarations for the Stock Unit. (For an individual Stock Unit, there is only one 'till', so the distinction is academic)

Term/Abbreviation	Definition
TIS	Technical Interface Specification. See DE/PRO/003.
TP	Trading Period
TPS	Transaction Processing System; application which collects transaction information and returns it to TIP
UI	User Interface. The screen.
Variance	An uncommitted cash discrepancy – i.e. a mismatch between the declared value of cash holdings for a Stock Unit, and the system derived figures based on opening figures for the balancing period and the record of transactions that have occurred since. Cash holdings are declared denominationally to support SAPADS, but variances are recorded simply as a value.
VB	Visual Basic – development tool produced by Microsoft.
VB6	VB version 6
VSS	Visual Source Safe

0.5 Changes in this Version

Version	Changes
0.1	None, this is the first version.
0.2	There have been a significant number of minor changes since the previous issue, so it is not practical to simply show the differences by markup. The following list gives an indication of the types of change. • Incorporation of comments from Gareth Jenkins, Roger Donato, Neil Gormley, Linda Miller, Martin McConnell, Mike Coon • More detailed cross-reference to EA/DPR/004 requirements being addressed. • Added section about Revaluation • Removed section about Counter Weekly Redeemed Savings Stamps report (now in scope of EA/HLD/005) • Clarifying the sections on Cash Declaration regarding ONCH replacement • Clarifying the sections about Variance handling regarding maintenance of the persistent objects. In particular the need to prune them at a certain age, and the need for background EOD activities to update distinct objects to avoid race conditions on the message store between nodes. • Clarifying that Non-cash declarations will also be preserved across period boundaries and pruned at a certain age. • Clarifying the transactions created in response to the various kinds of Transaction Correction request, and the ModeParameters definitions that support them. • Clarifying migration phasing – particularly with regard to reference data for EOD scheduling • Rationalising overlap with EA/HLD/005 - particularly regarding end of day protection against loss of data. Changes to EA/HLD/005 will be required to complete the rationalisation.
0.3	There have been a significant number of minor changes since the previous issue, so it is not practical to simply show the differences by markup. The following list gives an indication of the types of change. • Incorporation of comments from Gareth Jenkins, Roger Donato, Neil Gormley, Mik Peach, Pete Jobson, Martin McConnell, Mike Coon, Gerald Barnes • Added key section about validity of cash declarations at S80 • Clarify that pre-S80 cash declarations will be ignored, but that prior ONCH declaration will be honoured at first S80 logon after migration point 20 • Clarify that previous working day for cash declaration checking has the same meaning as for current ONCH – i.e. it is not simply yesterday • "Make Good" term deprecated in favour of "Add/Remove Cash" to avoid ambiguity with "Make Good" transaction corrections • Added section about Rollover Discrepancy Checking that clarifies the relationship between rollover and Check for Variances, and the calculation of discrepancies for volume stock • Clarify that although the user interface for revaluation is being removed, the underlying mechanism is retained for auto-revaluation of currency at rollover. • Clarify the in-day and end-of-day responsibilities regarding maintenance of Stock Unit Variance History and Till Declaration History objects, and added mark to Stock Unit Variance History object to support 'traded since' checking when assessing variance declaration validity

	• Clarify the catchup strategy for the end-of-day task that maintains variance objects. • Disentangle and clarify the bring forward algorithms for variance objects for the in-day and end-of-day cases. • Clarify the requirement that maintenance of these objects be disabled on disconnected counters. • Clarify that individual till declarations (or adjustments to them) for a shared stock unit do not affect the validity of an existing variance declaration. • Add support for new 'Committed Discrepancies' section on the variance report via new 'Stock Unit Committed Discrepancy History' persistent objects . • Add support for 'adjustments' summarisation in stock unit balance reports by writing an additional 'local transaction' as well as the event message. • Clarify the calculation of adjustments, discrepancy, suspense, and local suspense for the variance report. • Clarify the handling of variance report reprints • Clarify that 'remove cash' may not make a declaration negative. • Reworked the pruning strategy for cash and non-cash declaration persistent objects. • Clarified that removal of support for non-value stock declaration and checking will not occur until migration point 50 • Added missing role checks for transaction correction processing when invoked from logon • Clarified the locking strategy for transaction corrections • Changed the 'make good with cash' transaction processing logic to be more data driven in readiness for support of debit card at S90. • Clarified the use of secondary mappings on transaction correction transactions to support their counting in the branch trading statement • Clarified the use of sale/loss/zero product prices for transaction correction transactions, and that such transactions may not be reversed. • Clarified that the variances functionality is live from migration point 20, but will not report retrospectively on pre-S80 data. • New Type C reference data added (EPOSSStockUnit.TCParams, EPOSSStockUnit.OfficeVariances). • New EPOSSWatchdog configuration definitions added • New Event mode (EVNT – mode 20) defined • Event definitions enhanced to ensure valuable information such as declaration id included where appropriate • Details of non-counter changes removed (out of scope) • Implementation matrix removed (obsolete)
0.4	Changes since the previous issue are shown by markup, and may be summarised as follows • Incorporation of detailed comments from Gareth Jenkins, Neil Gormley, Mike Coon, Gerald Barnes, Jon Hulme • Preserve denominational cash declarations for inactive stock units (as currently done for ONCH) via new 'Cash Auto-Declaration on Inactive Rollover' logic • Add resolution for longstanding PEAK PC0081427 (spurious ONCH prompt) • Clarify the node isolation handling for the end-of-day Maintain Office Variances task • 'Calculate Stock Unit Committed Discrepancies' now caters for deleted stock units • 'Calculating the Suspense' now avoids message expiry for stock units that have rolled ahead of the office, and caters for deleted stock units. • Fill in various product numbers where now known.

	• Clarify the variance sign and the meaning of 'empty' and 'missing' for containers in Weekly Variance Persistent Objects • Identify the error conditions no longer reported by EPOSS Reconciliation • Revise the Type C reference data EPOSSStockUnit.OfficeVariances • Add cross-references to supporting LLDs, and update the risks section.
0.5	Changes since the previous issue are the result of detailed comments from Mark Scardifield, Jon Hulme, Walter Wright, Gerald Barnes, Mike Coon, Martin McConnell. They are shown by markup, and may be summarised as follows • Update cross-references to supporting LLDs (0.3) • Correct bring-forward pseudocode (4.1.8.2, 4.1.10.7.1) • Remove erroneous local suspense accounting nodes (4.1.8.4.2, 4.11.6.3.2) • Clarify use of 'x', 'n/a' and sign for the variance report (4.1.9, 4.10.3) • Extend section 4.2 to clarify loss price for stock adjustment. • Renumber events 55/56/57 as 65/66/67 to avoid duplicate use of 55 (4.11.3) • Correct Riposte query syntaxes (4.11.6.3.2)
1.0	• Incorporation of missing accounting node numbers (4.11.6.3), allowing simplification of Local Suspense query syntax (4.11.6.3.2) • Simplification of bring-forward logic (removal of status 'obsolete') to resolve status problem introduced at issue 0.2. See 4.10.3.2.1, 4.1.8.2, 4.1.9, 4.1.10.7.1.
1.1	• Protection Against Loss of Data (4.6.5, 4.7.1) now disables archiving more aggressively to support suspense report detail. • Calculate Office Variance (4.1.8.4) now scans for suspense and local suspense movements without fear of message expiry, and caters for clearance of local suspense 'ahead of the office'. • The Outstanding Transaction Corrections check at logon (4.7.4) disallows transaction corrections if the user has logged onto the default stock unit. • CAS Schedule definition for 'Maintain Office Variances' and 'Data Protection' tasks updated. See 4.11.2.2. • Addition of 'AllSuspense' attribute to Type C reference data (see 4.11.6.3.2) to support the Suspense report (see EA/HLD/005
1.2	• In principle, suppression of the Cash Variance report (CP3980) removes the need for the end-of-day Maintain Office Variances task altogether. However, the CP only requests suppression of the report buttons, so the underlying implementation still exists and continues to be described here (albeit with warning banners at the head of the relevant sections). • Committing the Transaction Correction (4.5.4) now clarifies that SW mode transaction corrections may use the correction Instruction product for settlement in obscure cases • Sections 4.5.5, 4.5.6 describe transaction correction handling for Linked and Bureau products respectively. • Protection Against Loss of Data (4.6.5) is now an End of Day check run with Administrator privileges from a Windows AT job rather than under the Counter Scheduler. • APS Transactions Report (4.9.2) and Desktop Buttons (4.11.5) now deal with softlaunch and MandatorySummaries issues

0.6 Changes Expected

Changes
None

0.7 Table of Contents

1.0 Introduction

This document forms part of the overall Counter High Level Design for the implementation of Impact Release 3. The high level designs provide the counter design definition in response to the IMPACT Release 3 Design Proposal provided in EA/DPR/004, which provides the design specification for the implementation of requirements stated in the Branch Trading Reporting, Management and Control and Transaction Management Conceptual Design [EA/CDE/002].

This document is one of seven design documents providing the counter design solution. It is complemented by:

EA/HLD/005	Balancing, Rollover and Stock Processing HLD
EA/HLD/008	Migration High Level Design for Impact Release 3
RD/DES/056	Reference Data End to End High Level Design for S80
EA/IFS/011	Report Production User Interface
EA/IFS/012	Declaration, Correction and Revaluation User Interface
EA/IFS/013	Balancing and Trading Statement User Interface

This document specifically provides a high level statement of the following subset of functionality that will change as a result of the requirements stated in EA/DPR/004:

- Cash Declarations and Variance Reporting (4.1)
- Stock Declaration (4.2)
- Non-Value Stock Declaration and Reporting (4.3)
- Rollover Discrepancy Checking (4.4)
- Transaction Correction Processing (4.5)
- Changes to End-of-Day (4.6)
- Changes to Log-On Checks (4.7)
- Revaluation (4.8)
- Remuneration Reporting (4.9.1)
- APS Transactions Report (4.9.2)
- Reports to be Removed (4.9.3)

See Appendix A – Design Proposal Cross Reference for more details.

1.1 Scope

This document forms part of the high level design for the IMPACT Release 3 Branch Trading project. The overall Post Office requirements for this project are stated in the Branch Trading Reporting, Management and Control and Transaction Management Conceptual Design [EA/CDE/002] and the Impact Release 3 Design Proposal [EA/DPR/004].

The scope of the changes required at the counter for Impact Release 3 have been analysed and it has been found that a natural split can be achieved in the changes required. A compliance Matrix is provided in section 10.0 that defines the specific responsibilities for satisfying the counter aspects of the DP between the two counter High Level Designs.

This document describes the changes required to the branch counters to handle Counter Declaration, Correction and Revaluation and is complemented by the high level design for Balancing, Rollover and Stock Processing [EA/HLD/005].

This document is only concerned with changes to the counter required to implement Impact release 3, changes to data centre systems being outside the scope of this document.

There is some overlap with the high level design for the Balancing, Rollover and Stock Processing Functions of Impact Release 3. This overlap will be stated where necessary.

The two counter HLDs both include elements of the Migration process to introduce Impact Release 3. This migration detail will supplement that described in the Migration HLD EA/HLD/008.

The objective of this document has a number of functions:

- To define a formal high level design in accordance with the Pathway Engineering process.
- To cross-reference to the Design Proposal in such a way as to ensure that the full scope of the document has been achieved.
- To identify where change will be incurred to any existing counter modules
- To identify any new modules that need to be delivered

1.2 Readership

This document is intended for application developers concerned with development of Impact Release 3. It is also intended to provide a detailed understanding of the software impacts incurred by Impact Release 3 as an aid to devising testing strategies and scripts.

It is suggested that readers have already read and understood EA/DPR/004 before continuing.

2.0 Design Principles

The following design principles will be adopted:

- ❑ 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.
- ❑ Functionality should be flexibly employed using soft-driven reference data rather than by hard-coding wherever practical.
- ❑ All user interactions will conform to SD/STD/001 wherever possible, to provide the clerk with an image consistent with the rest of the system.
- ❑ All application reference data used by the system will be temporal and must therefore be accessed through appropriate interfaces that provide support for this.

3.0 Requirements

Requirements for Impact Release 3 are documented in EA/CDE/002.

4.0 System Components

This section contains a full description of the new, changed and removed system components that are affected as a result of implementing the changes referred to in sections 1.1 and Appendix A – Design Proposal Cross Reference

4.1 Cash Declarations and Variance Reporting

One of the main changes to the counter process at S80 is to move from a weekly balancing mechanism (by Cash Account Period) to a monthly balancing mechanism (by Trading Period). However, there is a requirement to enable reconciliation of cash on a more regular basis.

This section discusses the changes that will be made to the process of cash declarations, the reporting of variances between declared and actual cash and the process of rectifying any differences between declared and actual. For a description of the dialogues and report formats, please refer to EA/IFS/012.

Note: As a result of CP 3980, the Cash Variance Report has been suppressed by removing the buttons that produce or re-print it. However, the underlying logic to support it remains in place and is described in this document for reference.

4.1.1 Validity of Cash Declarations

Prior to S80, only cash declarations made in the current BP are presented in declaration lists, and taken account of during discrepancy reporting and rollover checks. In other words, shared stock unit till declarations have a lifetime of the current BP. Once a Stock Unit has been rolled over into a new BP, all Till Ids are "forgotten", and you have to re-declare them in the new BP even if the Till is idle.

This presents a problem in reporting on Till level declarations on the Cash Variance Report which is aligned to a calendar week (Thursday to Wednesday) rather than to a BP (which is arbitrary).

To address this, the following changes will be made

- Each non-zero Cash Till Declaration made after S80 will have an indefinite lifetime.
- If an S80 Till Declaration id is no longer needed, it should be declared with zero cash, and the id will be logically deleted at the next Desktop load (usually 3am the next morning)
- All S80 Cash Till Ids (i.e. those that have not yet been logically deleted) will be presented in the list of Tills when doing a Check for Variances and equivalent functions
- Existing (pre-S80) Cash Declarations will be ignored for both declaring, reporting and balancing purposes. This avoids problems caused by obsolete till declarations suddenly being deemed valid and interfering with balancing discrepancy checks. (This approach should not cause a problem because normal practice is to make ONCH declarations rather than Cash Declarations except just before balancing).
- S80 Cash Till Ids will remain on the Cash Variance Report until the end of the current week, however if they have been zeroed and logically deleted, their value will be reported as zero
- These changes do not apply for Stock, Currency and Stamps Till Ids because they are not required for the Cash Variance Report - i.e. they will be obsoleted at each rollover and

pruned after two CAPs, just as prior to S80. They also remain valid until rollover even if zero

- Pre-S80 Cash declarations will also be pruned after two CAPs (as prior to S80) – purely as a housekeeping exercise.
- A consequence of keeping S80 Cash declarations indefinitely is that if a stock unit has been inactive for longer than the Riposte message expiry period, then the original denominational breakdown for the declaration will not be available. This would be a regression, because inactive stock unit rollover currently supports LFS by refreshing ONCH declaration messages and bringing them forward to the current week. Inactive stock unit rollover will therefore now regenerate S80 Cash declarations as described in 4.1.11.

Note that continuity of declaration causes a subtle change to rollover behaviour for inactive stock units. Explicit rollover currently requires a cash declaration to have been made in the current BP, but this will no longer be true. There is of course no difference in behaviour if the stock unit has been active, because cash movements invalidate existing declarations anyway.

Note: Explicit rollover of an inactive stock unit is atypical, because it is normally rolled over using the separate 'inactive stock unit' process which takes no account of declarations, relying instead on declaration trailers brought forward from the previous BP.

To support the above changes, non-zero S80 cash declaration persistent objects will now be retained indefinitely rather than being pruned after two CAP/TPs, but will be logically deleted after being explicitly zeroed. See 4.1.12 for details.

This change will be implemented from migration point 20 onwards, so will be equally applicable to stock units operating in CAP or TP mode.

4.1.2 Removal of ONCH Declarations

The pre-S80 system provides two mechanisms by which cash can be declared:

- Cash Declaration (usually prior to stock unit rollover)
- ONCH Declaration

The ONCH declaration mechanism is visibly very similar to the Cash Declaration process but was introduced to encourage daily declarations of cash for the purpose of relating this information to the SAP ADS system via Cash Statement records.

Since the ONCH and the Cash Declaration functions are very similar, the ONCH Declaration function will be retired at S80 and it is intended that the Cash Declaration function will be used to declare cash on a daily basis.

Therefore the existing ONCH declaration button (F9 on the Counter Daily menu) will be updated at migration point 20 via SoftLaunch such that it invokes the same process as the Declare Cash button on the Stock Balancing menu (thereby replacing ONCH with Cash Declaration).

In addition, the process that checks that ONCH declarations have been made, and forces an ONCH declaration during the logon process will be modified such that it checks Cash declarations have been made, and calls the Cash Declaration process instead of the ONCH Declaration process. See section 4.1.7 for more details.

The ONCH declaration function is responsible for the creation of persistent objects that are used during the production of the Cash Statement messages at the end of each day. Since the ONCH declaration function is being retired, then in theory the LFS End of Day function should be updated. However, the LFS EOD function is quite happy using existing Cash Declaration messages in the absence of ONCH messages, so no change is required (see 4.1.3).

4.1.3 LFS Cash Statement Generation

The LFS Cash Statement messages are generated by the LFS EOD process. This process uses the ONCH declaration or the Cash Statement declaration made by each stock unit (whichever declaration is the latest for the current day).

Once the ONCH declaration mechanism is removed, then the LFS EOD process will always use the latest Cash Declaration performed on the current Trading Day as the basis for the generation of the Cash Statement.

Therefore no change is required to the LFS EOD process except that reference to ONCH messages ought to be removed at some future release (if the counter is retained in its current form).

Note: Although 'Add/Remove Cash' actions (see 4.1.10) affect the original declaration in terms of discrepancy calculation, they are not picked up by the LFS EOD process, so the declared cash statement may be out of date.

The LFS EOD process is discussed in further detail in section 4.6.1.

4.1.4 Changes to Cash Discrepancy Processing

4.1.4.1 Current Mechanism

Cash discrepancies are identified within a number of different functions within the system:

- During Cash Declaration for an individual stock unit
- During the Discrepancies Check (menu SBAL L2/S4 Stock Balancing)
- During BP and Trading Period Rollover

For an individual stock unit, there is just one cash declaration. For shared stock units, there can be a number of declarations, each one identified by a declaration id (sometimes referred to as a 'till').

The discrepancies are evaluated by using the DrawerItemDeclarationId attribute within the Declaration_sss[1] collection to identify the most recent denominational declaration messages made by the stock unit. These are summed and compared with the derived cash value within the stock unit to determine any variance/discrepancy.

During the BP and CAP Rollover process for a Stock Unit, the deduced discrepancy amount is recorded as an EPOSSTransaction pair in order to Increase/Reduce the cash value of the Stock Unit and post the same amount to the appropriate Gain or Loss Discrepancy product (as defined via EPOSSStockUnit.Parameters).

The most recent Cash Declaration denominational information is brought forward as part of the Rollover process but this is not used.

(During Office rollover, the holdings of the various Stock Units (including discrepancy products) are also accumulated against notional containers '##' and '#1' to support production of the Cash Account and Office Snapshot reports respectively, but it should be emphasized that these containers are not true Stock Units. See EA/HLD/005 for more details).

If the user wishes to rectify a cash discrepancy using the pre-S80 mechanisms, they can add or remove the value of discrepancy to/from the till and then re-declare the cash (using the Cash Declaration function). This then updates the declared value to equate to the derived value of cash for the stock unit and the most recent cash declaration will then balance.

4.1.4.2 Changes to Current Mechanism

A variance/discrepancy between the declared cash balance and the derived cash balance can be rectified at any time prior to Stock Unit rollover using the new 'Add/Remove Cash' functions that are described in section 4.1.10.

If no such action is taken, then rollover will automatically transfer the discrepancy to the Local Suspense account for the branch, and this will need to be resolved prior to Office rollover (see EA/HLD/005 and EA/IFS/013).

For shared stock units, the variance is based on the sum of all its cash declarations, but Add/Remove Cash applies at the declaration level. To determine the effect on variance of such an action, the user can manually invoke the Check for Variances function.

[1] Where *sss* is the Stock Unit identifier

Note also that 'Add/Remove Cash' functions cannot update the denominational declaration messages that were made in the most recent Cash Declaration since they do not request the user to enter the changes to the Stock unit cash value at a denominational level.

In addition, the adjustment amount cannot simply be written as a new message with the same DrawerItemDeclarationId attribute as the most recent cash declaration since, again, it would contain no denominational breakdown and would impact the LFS Cash Statement generation as well as the rollover process.

Instead, the Adjustment value entered during the 'Add/Remove Cash' functions will be stored as an additional attribute '<EPOSSTransaction.Adjustment>' in the Cash Declaration Persistent Object (see 4.10.1). This additional attribute value then needs to be taken into account during the variance/discrepancy calculation by adding it to the sum of the denominational declaration messages before then comparing the result with the system derived figure.

These changes should be made within the EPOSSDeclare function so as to have minimal impact on the StockUnit functions and other areas of the system.

It should be noted that any re-declaration of cash must set the value of '<EPOSSTransaction.Adjustment>' to zero since a re-declaration is a completely new statement of what cash is present against a stock unit or till. In other words, any previous assertion that physical cash has been added or removed is irrelevant, since it is the current holding that is being declared.

Nevertheless, when updating an existing declaration, the existing declared value and adjustment need to be presented separately on the screen in order to clarify the current position. See section 4.1.10.9 (Re-Declaration Following Adjustment) and EA/IFS/012.

4.1.5 Cash Declaration

The existing Cash Declaration process remains largely unchanged. Any ONCH processing embedded within the existing cash declaration code can be removed since the changes to the cash declaration functions will be implemented immediately at migration Point 20.

The Cash Declarations process needs to be able to take a parameter to indicate whether the declaration being made is for today or the previous working day. Normal invocation of the function will cause the declaration to be recorded against today. Invocation that is forced during the logon process will cause the declaration to be recorded retrospectively against the previous working day.

Note that 'previous working day' is subtly different from 'yesterday', in that it ignores days on which the branch would normally be closed (regardless of whether it was actually open on those days this week). This approach is the same as the existing ONCH logic, and is based on the existing Reference Data object OpeningPeriods.OfficeHours.

The Cash Declaration report dialogue needs to be furnished with an additional button that will allow the 'Check for Variances' function to be invoked immediately following a cash declaration. This additional button is only applicable to Shared Stock units since individual stock units do not require the 'Check for Variances' functionality. A check on the Stock Unit type will therefore determine whether the button is visible.

4.1.5.1 Individual Stock Units

For individual stock units, the pre-S80 system automatically checks to see whether there is a discrepancy between the declared value of cash and the derived value of cash.

The only change to existing functionality is that the value of the '<EPOSSTransaction.Adjustment>' attribute in the Cash Declaration Persistent Object (see 4.10.1) should be set to zero as part of the declaration since re-declaration should include any amount made good (Refer to section 4.1.10.4).

However, the functionality will be extended to create or update a Stock Unit Variance History persistent object as described in section 4.10.3.2. The relevant Decl_ddd container will be written depending on the day of the week for which the declaration is being made.

When updating a Stock Unit Variance History in this manner, opportunity must be taken to prune old markers, as described in 4.10.3.2.2, in order to contain the total size of the message to the Riposte limit of 2048 characters.

4.1.5.2 Shared Stock Units

The only change to existing functionality is that the value of the '<EPOSSTransaction.Adjustment>' attribute in the Cash Declaration Persistent Object (see 4.10.1) should be set to zero as part of the declaration since re-declaration should include any amount made good (see section 4.1.10.4).

However, the functionality will be extended to create or update a Till Declaration History persistent object as described in section 4.10.3.1. The relevant Decl_ddd container will be written depending on the day of the week for which the declaration is being made.

4.1.6 Check for Variances

A new function is required that will perform a cash discrepancy (also known as cash variance) check for shared stock units. This new function may either be invoked from a new desktop button (F13 on the Stock Balancing menu) or from the report dialogue that appears following a cash declaration (refer to section 4.1.5).

The 'Check for Variances' function will display an error message if is has been invoked by a user who is attached to an Individual stock unit since the function is only relevant to Shared stock units.

4.1.6.1 Display Declarations

Initially, a list of S80 cash declarations is presented so that the user is able to determine when the most recent declarations have been made for each till (declaration id). The user is then able to continue with the variance check or may abandon the operation using the 'prev' or 'desktop' button.

The list of S80 cash declarations will be prepared from the Cash Declaration Persistent Objects for the stock unit (see 4.10.1), and the declaration id, date, time, user and node information will be displayed for review.

Note: Any pre-S80 cash declarations will not be included.

If no such cash declarations exist for the current stock unit (probably due to this being a new stock unit), then the user will be informed that no declarations have yet been made.

Currently, the cash declaration list is restricted to those declarations made in the current BP, but this constraint will be removed. See 4.1.1.

4.1.6.2 Variance Check

Selecting 'Ok' from the list of declarations that is presented, will cause the system to compare the system derived stock unit cash total with the sum of the declared cash values in the same way as for the overall 'Discrepancies' function (stock balancing menu – F8).

Both functions use the collection 'Declaration_sss*'* persistent objects during the discrepancy/variance checking process.

- If the sum of the declared values matches the derived value for the Stock Unit, then a message is presented that informs the user that there are no variances with the system derived figures.
- If the sum of the declared values does not match the derived value for the Stock Unit, then a dialogue informs the user that there is a difference. This difference is the variance, and is presented to the user as either a gain or a loss (as in the existing Discrepancies function).

Note:This function (and any other system discrepancy check) must take account of any adjustments made during 'Add/Remove Cash' as described in section 4.1.4.2.

4.1.6.3 Variance History Update

Regardless of the result of the check, the Variance Check function will finally update or create the Stock Unit Variance History persistent object for the current day with values as described in section 4.10.3.2.

When updating a Stock Unit Variance History in this manner, opportunity must be taken to prune old markers, as described in 4.10.3.2.2, in order to contain the total size of the message to the Riposte limit of 2048 characters.

4.1.6.4 Check Variance Event Recording

The 'Check for Variances' function will be audited by means of writing a new event each time the function is executed. Different events will be written depending on the outcome of the function:

Event 63	Check for variance function completed.
Event 64	Check for variance function completed with variance/discrepancy.

These events are fully defined in section 4.11.3.

4.1.7 Retrospective Cash Declaration at Logon

Pre-S80, there is a process that checks, during logon, whether an ONCH declaration was made for the current stock unit on the previous working day, and if not, forces the user to make a retrospective ONCH declaration.

This will be modified such that a failure to make a Cash Declaration on the previous day will force the User to make a retrospective Cash Declaration instead, as described in section 4.1.5.

It is also the opportunity to resolve a longstanding PEAK (PC0081427) regarding needless ONCH declaration. Pre-S80, when a user attached to a stock unit that had not been declared on the previous working day, they were not required to make a retrospective ONCH declaration. However, if they then made a voluntary ONCH declaration (i.e. for today), they had to make another retrospective declaration if they logged out and back in again.

Since we must not require the logon declaration more often than pre-S80, we will relax the logon check to regard a declaration for today as an acceptable substitute for one made on the previous working day.

As an aside, if a Cash Declaration was performed on the previous day, then *regardless of whether any trading has happened since then for the current stock unit*, the check will not require a new declaration to be made at logon. This is not an ideal specification, but reflects the current pre-S80 ONCH implementation, and there is no mandate to change it. In particular we must not require the logon declaration more often than pre-S80.

Note also that forcing a retrospective Cash Declaration function has little effect on the system since the declared figures cannot be included in the LFS Cash Statement for the previous night. However, there will be a minor benefit to the completeness of the Cash Variances report and this benefit depends on the type of stock unit.

With the removal of ONCH and its associated persistent objects, the actual implementation of the check is based on new Stock Unit Variance History persistent objects (for individual stock units) and Till Declaration History persistent objects. (for shared stock units) that will be maintained by the cash declaration logic. See sections 4.10.3.2, 4.10.3.1 respectively.

To avoid forcing a logon-time cash declaration on first start-up after S80 code is activated, any residual ONCH objects must also be checked. However, since these will not even exist once S80 has been up and running for some while, the check for their existence should be done **after** the checks on Stock Unit Variance History/Till Declaration History described above.

If a logon-time cash declaration is made, it must be recorded as being on behalf of the previous working day in the appropriate Stock Unit Variance History or Till Declaration History object as described in 4.1.5.

In addition, for shared stock units, the option will be given to perform the ‘Check for Variances’ function (see section 4.1.6). This function must also record its results against the previous working day in the appropriate Stock Unit Variance History object, as described in 4.1.6.3.

In both cases, this means that the current date will be captured into the declaration itself, but the declaration will be recorded in the Decl_ddd container for the previous working day, in the persistent object for the week containing that day.

4.1.7.1 Individual Stock Unit

An individual stock unit that has not explicitly declared cash on the previous day will be forced to make a Cash Declaration for the previous working day as part of the logon process.

The results of this declaration can immediately be assessed and a variance calculated and recorded in a Stock Unit Variance History persistent object as described in section 4.1.5.1. This variance will then be reported on the Cash Variances report correctly.

Note that Stock Unit Variance History objects brought forward from earlier days do not count as explicit declarations, so do not affect the logon check

4.1.7.2 Shared Stock Unit

The verbally stated requirement is that there is ‘no change to the current ONCH process’.

However, the underlying messages in the message store are changing and it is assumed that the requirement means that there should be no change to the user interface. Unfortunately, the dialogue during a shared stock unit Cash Declaration has already changed since the Cash Declaration allows the user to update a previous declaration and finally allows the user to invoke the ‘Check for Variances’ function. The following is therefore an interpretation of the requirement.

Where no user within a shared stock unit has explicitly declared cash on the previous working day, then the first user to logon to that shared stock unit during the following day will be forced to make a Cash Declaration for that previous working day.

Note that Till Declaration History objects brought forward from earlier days do not count as explicit declarations, so do not affect the logon check

This user then has the ability to follow-up the retrospective cash declaration with a retrospective ‘Check for Variances’ function that will write the results of the variance check against the previous working day.

The retrospective declaration made will be shown on the Variances report under the column for the previous working day, but any brought forward declarations for other Declaration Ids of the same Stock Unit will be unaffected. In other words they will remain the ones brought forward from the day before that, since it is assumed that they performed no trading.

If the retrospective ‘Check for Variances’ function is declined as part of the logon process then the Cash Variances, Derived Figures and Declared Figures of the Cash Variances report will be based on brought forward values (if any). These may be deemed valid or not, depending on whether trading has occurred since the values were last calculated, but that decision is not affected by the new declaration.

If the retrospective ‘Check for Variances’ function is accepted as part of the logon process then the Cash Variances, Derived Figures and Declared Figures of the Cash Variances report will have newly calculated values. However, it should be understood that a variance check made by the first user logging-on will always result in a variance (discrepancy) figure unless that user was the only person trading in that stock unit on the previous working day (or is declaring the totality of the Stock Unit – which is sometimes the case – i.e. a single physical till with cash / stock into which multiple users dip).

4.1.8 Maintain Office Variances

This is a new End of Day process that calculates cash variances across the branch so that production of the Cash Variance Report, described in section 4.1.9, simply needs to read a set of persistent objects (Till Declaration History, Stock Unit Variance History, Stock Unit Discrepancy History and Office Variance History) rather than having to perform complex processing across the period of one week.

For an overview of those objects and their lifecycle, see 4.10.3

A consequence of this approach is that if EOD doesn't run (eg due to a counter being switched off or having failed), then the variance report will miss out the relevant days until the EOD process successfully executes and performs a catch-up activity.

Note: As a result of CP 3980, the Cash Variance Report has been suppressed by removing the buttons that produce or re-print it. However, the underlying logic to support it remains in place and is described in this document for reference.

That report contains information at four levels for each day

- a breakdown of cash declaration by till for each shared stock unit
- a summary of the derived cash, declared cash and cash variance for each stock unit
- a summary of the committed discrepancy for each stock unit
- a summary of cash-related movements for the whole office

The End of Day process therefore carries out the following actions

- As described in 4.1.8.1, it brings forward the Till Declaration History values for each shared stock unit from the previous physical day and week, updating the status to 'bfwd' rather than 'ok'. (This distinction is important to the logon check logic described in 4.1.7).
- As described in 4.1.8.2, it brings forward the Stock Unit Variance History values for each stock unit (individual or shared) from the previous physical day and week, updating the status to reflect any change of validity.
- As described in 4.1.8.3, it calculates the committed discrepancy for each stock unit by analysing transactions since the start of the balancing period, and records them in a Stock Unit Discrepancy History persistent object.
- As described in 4.1.8.4, it calculates various summary figures for the office by analysing transactions for the day, and records them in an Office Variance History persistent object.

When calculating these office summary figures for a particular day, it does so by processing transactions for the current Trading Day, i.e. those between yesterday's EOD marker and today's EOD marker. However, there are some cases where the search start-point is different. See the individual descriptions for details.

The existing End-of-Day Reconciliation module (see EP/DES/025) will be cloned and modified to meet the requirement since it contains existing infrastructure that can scan all messages for each outstanding Trading Date. This helps caters for scenarios where catch-up is required because previous end-of-day processes have failed to run.

Catch-up should go back as far as the most recent successful Maintain Office Variances execution, which can be determined by checking for existence of the Office Variance History persistent object corresponding to each EOD marker.

Note that existence of Till Declaration History or Stock Unit Variance History objects cannot be used to determine how far to go back because they are also (indeed mostly) maintained by in-day logic.

Note also that the existing Daily Reconciliation logic only attempts catch-up for a limited period in arrears, but the catch-up here should if necessary go back as far as the point where transactions are archived.

If no Office Variance History object is found even then, it should be assumed that Maintain Office Variances is being run for the first time following counter upgrade to S80, and so only today need be handled.

Having determined the extent of catch-up required, each day can be processed in turn going forward, so the processing for a given day can assume yesterday's figures are either available or not applicable.

Because of the need to support 'catchup', all references to 'today' and 'yesterday' in the sections below should be taken to mean days relative to the day being processed, rather than the physical system date.

The end-of-day logic runs only in the gateway node, and must not run if that node is disconnected. In principle, this decision could be according to EPOSSWatchdog topology rules[2], since the in-day functions also obey those rules (see 4.11.6.1). However, it is simpler to adopt the more conservative stance that the gateway node must be able to see all other nodes.

Note: It is not sufficient to check for the existence of an EOD mark for today, because that could have been written by the EODMarkers task at an earlier time. The check should therefore based simply on using Riposte 'NextNode' and 'GetConnections' APIs to enumerate and verify connectivity to the configured local neighbour nodes.

Any in-day logic that affects the variance history objects (i.e. Cash Declaration, Check for Variances, Add/Remove Cash) must also be disabled on disconnected nodes. See 4.11.6.1 for details of configuring EPOSSWatchdog to achieve this.

The fact that both in-day and end-of-day logic may need to create or update the same Till Declaration History or Stock Unit Variance History object can cause problems if the updates occur at the same time. To get round this problem, we introduce the idea of 'background' and 'foreground' objects. For more information, see section 4.10.3

The following sections describe the activities in more detail

4.1.8.1 Bring-Forward Shared SU Till Declarations

Some tills within a shared stock unit may not have declared cash today. This may be due to the fact that the person responsible for the till did not work today, or it may be that the person simply did

[2] Apart from degenerate cases, the EPOSSWatchdog deems nodes other than the gateway to be connected if they can see the gateway, or can see all other nodes except the gateway. It deems the gateway node to be connected unless it can see none of the other nodes. For more details, see EP/LLD/016

not bother to perform a declaration. Either way, the system cannot tell the difference and it will assume that the last cash declaration performed for each of the tills is valid and that no further transactions have been performed following the most recent declaration.

Therefore, if one or more till cash declarations have not been made for the tills in a shared stock unit for today, they can be brought forward from yesterday if available.

If the EOD process is operating in catch-up mode, this logic may be applied repeatedly and can result in a cash declaration being propagated forward a number of days. However, the decision as to far how far to go back is based on the Office Variance History rather than the existence of Till Declaration History objects because those are also maintained by in-day logic.

The result of the 'bring forward' logic is to create a 'Background' default for today, which will only be used if there is no corresponding 'Foreground' object. For more information about 'foreground' and 'background' objects, see 4.10.3.

The EOD process will therefore process each Declaration Id for each Stock Unit for yesterday, trying to read either the 'Foreground' till declaration (as created by an in-day action), or failing that the 'Background' till declaration object (as created by an earlier invocation of the EOD process).

For each one found, it tries to read the 'Foreground' till declaration for the same Declaration Id and SU for today.

If a 'Foreground' till declaration is *not* found for today, then a corresponding 'Background' till declaration object is created, based on yesterday's till declaration object (Foreground or Background) with the following differences:

```
Data.Decl_ddd.Date = Current Date/Time
Data.Decl_ddd.User = 'eod'
Data.Decl_ddd.S = 'bfwd'
```

If today is Thursday, and the value to be brought forward is non-zero, then it is also stored in the 'BFwd' container of the 'Background' till declaration object, regardless of whether there is a 'Foreground' till declaration already made for Thursday.

This is because in-day logic never fills in the 'BFwd' container, so the EOD activity on Thursday night (or subsequent catch-up) must always do so.

However, if today is Thursday, and the value to be brought forward is zero, and there is no 'Foreground' till declaration already made for Thursday, then neither the 'BFwd' container or the 'Decl_Thu' containers should be created. In this way, Declaration Ids that are no longer used will be retired from variance reporting after at most one week of being dormant.

For more details, see Till Declaration History (4.10.3.1)

Note: If no cash declaration is found for yesterday, the effect is simply that nothing is brought forward to today. However See 4.1.9 for how this affects the Cash Variance Report *– in particular during the first week after Point 20 migration or creation of a stock unit.*

4.1.8.2 Bring Forward Stock Unit Variances

Where individual Stock Units have not declared cash today or where the 'Check for Variances' function has not been performed for a Shared stock unit today, then yesterdays overall stock unit cash declarations and variances, recorded in Stock Unit Variance History objects, can be brought forward if available.

However, they should not be brought forward if trading has occurred since then, or if the associated Stock Unit has already been logically deleted from the system.

When updating a Stock Unit Variance History in this manner, opportunity must be taken to prune old markers, as described in 4.10.3.2.2, in order to contain the total size of the message to the Riposte limit of 2048 characters.

The bring forward action described below produces a ‘Background’ value. (Similar logic can also occur on demand during the day as described in 4.1.10.7.1, producing a ‘Foreground’ value).

For more information about ‘foreground’ and ‘background’ objects, see 4.10.3.

The EOD bring forward logic can be expressed in Pseudo-Code as shown below:

```
If SU has been logically deleted or yesterday’s declaration (Data.Decl_ddd) is missing
Then
    Do nothing

Else
    If SU has declared (Individual)
        or checked for variance (shared) today { i.e.today’s  Data.Decl_ddd.S = ‘ok’ }
    Then
        { declaration is ok for today => no need to check for subsequent trading }

    Else If declaration brought forward and still valid { i.e.today’s  Data.Decl_ddd.S = ‘bfwd’ }
    Then
        If the SU has traded since the declaration { i.e. between today’s Data.Decl_ddd.Mark and today’s EOD mark }
        Then
            Set today’s declaration status (Data.Decl_ddd.S) to ‘bad’
        Endif

    Else If declaration brought forward but invalid { i.e.today’s  Data.Decl_ddd.S = ‘bad’ }
    Then
        { declaration status already ‘bad’ => no need to check for subsequent trading }

    Else { declaration needs to be brought forward }
        If yesterday’s declaration status (Data.Decl_ddd.S) is ‘bad’
        or SU has traded since yesterday’s declaration { i.e.between yesterday’s Data.Decl_ddd.Mark and today’s EOD mark }
        Then
            Set today’s declaration status (Data.Decl_ddd.S) to ‘bad’
        Else
            Set today’s declaration status (Data.Decl_ddd.S) to ‘bfwd’
        Endif

        Set today’s Data.Decl_ddd .Date To current date/time
        Set today’s Data.Decl_ddd .Value to yesterday’s value
        Set today’s Data.Decl_ddd .Mark to yesterday’s mark
        Set today’s Data.Decl_ddd .Var to yesterday’s Var
        Set today’s Data.Decl_ddd .User to ‘eod’
    Endif

    If today is Thursday
    Then
        Set BFwd.Decl_ddd .Value to yesterday’s value
        Set BFwd.Decl_ddd .Var to yesterday’s Var
        Set BFwd.Decl_ddd .S to yesterday’s status
    Endif

Endif
```

By ‘traded’, we mean there have been transactions that affect the cash position for the stock unit. This is determined using search criteria derived from the ‘Queries.Traded’ attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2), filling in the stock unit id as necessary.

If today is Thursday, then the brought forward figures are also stored in the ‘BFwd’ container of the ‘Background’ till declaration object, regardless of whether there is a ‘Foreground’ declaration already made for Thursday. (This is because in-day logic never fills in the 'BFwd' container, so the EOD activity on Thursday night (or subsequent catch-up) must always do so.)

For more details, see Stock Unit Variance History (4.10.3.2).

Note that if no declarations are found for yesterday, the effect is simply that nothing is brought forward to today. See 4.1.9 for how this affects the Cash Variance Report – in particular during the first week after Point 20 migration or creation of a stock unit.

4.1.8.3 Calculate Stock Unit Committed Discrepancies

This activity calculates the currently committed discrepancy value for each stock unit by analysing transactions since the start of the balancing period, and recording the results in a Stock Unit Discrepancy History persistent object.

The committed discrepancy (not just cash) for each stock unit is intended to provide continuity of variance information on the Cash Variance Report when a stock unit is balanced, since the effect of doing a trial balance is to zeroise cash (and other) variances in favour of a committed discrepancy.

Without this information on the report, it would not be obvious why the variance had suddenly disappeared. It includes non-cash discrepancy as well because that will also need to be resolved at rollover into the next TP.

Although it would be plausible to record this information during rollover itself when calculating a new value of the committed discrepancy, it is better done as an end-of-day snapshot, by the following reasoning :.

- The rollover process converts variance into committed discrepancy as part of the initial trial balancing phase.
- If the rollover is abandoned at that stage, or the rollover is into a new BP within the existing CAP/TP, then any committed discrepancy remains outstanding, ready to be adjusted by subsequent rollovers.
- However, if the rollover is into a new TP, the discrepancy is converted into an adjustment to the office-wide local suspense.
- The local suspense position for the office is calculated as an end-of-day snapshot
- If the stock unit committed discrepancy were recorded during rollover, it is not clear whether it should be the value before trial, after trial, or after TP-rollover. Whichever one was chosen, the value would not tally with the cash variance beforehand (because it also includes non-cash discrepancies), and only the 'after' value would tally with the local suspense.
- Assuming the 'after' value is the appropriate one, it might as well be recorded as part of the end-of-day snapshot, so that it is synchronized with the local suspense value, and so that the various alternative rollover outcomes (trial-only, BP-rollover, TP-rollover) are handled consistently.

The mechanism to calculate the committed discrepancy for each stock unit is a Riposte message store scan from the start of the applicable balance period up to the day's EOD mark, summing the <EPOSSTransaction.SaleValue:> attribute for all transactions that create discrepancy products.

The message store scan and selection of the appropriate balance period must take account of the day for which the analysis is being done. This means walking the chain of rollover trailers back from the relevant stock unit persistent object until one is found that occurred on or before the day in question.

Note: If in catch-up mode, we may be analysing the position for some days ago, in which case we need to consider deleted stockunits as well as current ones. This means iterating the DelStockUnits collection as well as the StockUnits collection.

The scan must include the brought forward discrepancies (opening figures) for the BP, and any subsequent movements.

The discrepancy for each stock unit (apart from the default stock unit of course) is obtained as follows:

- Read the StockUnits.*sss* or DelStockUnits.*sss* persistent object, and use the 'RolloverTrailer' attribute to read the latest RolloverTrailer message. If no such message is found, the stock unit is ignored. (This can only occur if the stock unit has not rolled for a long time and message expiry has occurred. In that obscure situation, it is better to output 'n/a' on the variance report than wrong figures)
- Compare the message id of that trailer with the EOD mark for the day to see if the rollover was completed on or before that day. If not, use the 'Previous' attribute to follow the chain of RolloverTrailer messages until an appropriate one is found. If no appropriate one is found, the stock unit is ignored (for the same reason as above).
- Use the 'OpeningFigures' attribute from the RolloverTrailer message to read the OpeningFiguresTrailer message.
- Get the 'OpeningFiguresId' from the OpeningFiguresTrailer message.
- Use the 'Rollover' attribute from the RolloverTrailer message to read the Rollover message
- Get the 'Mark' attribute from the Rollover message, and use that as the start-point for the scan.
- Search for messages between this start mark and the EOD mark for the day, using search criteria derived from the 'Queries.Discrepancies' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2), filling in the stock unit id and opening figures id as necessary.
- For each message found, sum the 'EPOSSTransaction.SaleValue' attribute to produce the discrepancy. The total sum is a figure that is -ve for a nett shortage (i.e. Discrepancy Short), and +ve for a nett surplus (Discrepancy Gain)

Having determined the discrepancy value, the relevant Decl_ddd container for the Stock Unit Discrepancy History object will be written depending on the day of the week for which the discrepancy calculation is being made.

In addition, if today is Wednesday, the value will be also stored in the 'BFwd' container for the following week.

Note: We 'push' the value forward on Wednesday into the next week rather than 'pull' it forward on Thursday simply because it is more convenient to do so.

It is OK to do this because the object is only updated by the End-of-day activity. Other objects such as Stock Unit Variance History that are maintained both by in-day and end-of-day must use a 'pull' approach to avoid problems with carrying forward figures and status that may be superseded later.

4.1.8.4 Calculate Office Variance

This activity calculates various summary figures for the office by analysing transactions and recording the results in the relevant Decl_ddd container for the Office Variance History persistent object as described in the following sections.

In addition, if today is Wednesday, the values will be also stored in the 'BFwd' container for the following week.

Note: We 'push' the value forward on Wednesday into the next week rather than 'pull' it forward on Thursday simply because it is more convenient to do so.

It is OK to do this because the object is only updated by the End-of-day activity. Other objects such as Stock Unit Variance History that are maintained both by in-day and end-of-day must use a 'pull' approach to avoid problems with carrying forward figures and status that may be superseded later.

4.1.8.4.1 Calculating the Suspense

This calculates the current Suspense position – i.e. the nett loss or gain for all suspense products.

Unlike the Suspense report, it takes account of all suspense movements, not just those within a particular office CAP/TP. In other words, movements to suspense from a stock unit that is ahead of the office CAP/TP will be included as well.

In principle, the mechanism to achieve this is a Riposte message scan up to the day's EOD mark, summing the <EPOSSTransaction.SaleValue:> attribute for all transactions that contribute to the Suspense position – i.e. those that have a Level 2 Primary Mapping of 740 (Unclaimed Payments) or 490 (Uncharged Receipts).

We need to include the brought forward suspense (opening figures) for the office CAP/TP, plus any relevant movements for each stock unit. Since stock units roll into the office CAP/TP ahead of the office, these movements may be prior to the office rollover itself.

Therefore, the overall solution approach is

- Obtain the brought forward suspense (opening figures) for office CAP/TP n. This summarises all movements from stock units in TP n-1 and before.
- For each of the stock units, add in the latest suspense movements since the stock unit rolled into TP n.

The message store scans and selection of the appropriate office and stock unit rollover trailers must of course take account of the day for which the analysis is being done. This means walking the chain of trailers back from the relevant persistent object until one is found that occurred on or before the day in question.

The logic is therefore as follows :

<u>Determine the brought forward office suspense (opening figures)</u>

- Read the EPOSSCAP.Office persistent object, and use the 'CAPTrailer' attribute to read the CAPTrailer message. If no such message is found, then the brought forward suspense will be assumed to be zero.

- Compare the message id of that trailer with the EOD mark for the day to see if the office rollover was completed on or before that day. If not, use the 'Previous' attribute to follow the chain of CAPTrailer messages until an appropriate one is found. If no appropriate one is found, then the brought forward suspense will be assumed to be zero.
- Use the 'RolloverTrailer' attribute from the CAPTrailer message to read the appropriate RolloverTrailer message.
- Use the 'OpeningFigures' attribute from the RolloverTrailer message to read the OpeningFiguresTrailer message.
- Get the 'OpeningFiguresId' from the OpeningFiguresTrailer message.
- Use the 'CAPRollover' attribute from the RolloverTrailer message to read the CAPRollover message
- Get the 'Mark' attribute from the CAPRollover message, and use that as the start-point for the scan.
- Since we are only interested in messages written before (and on the same node as) the RolloverTrailer, use the message id for that to build an end mark from the start mark
- Search for messages between the start and end marks, using search criteria derived from the 'Queries.OfficeSuspense' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2), filling in the opening figures id as necessary.

Determine any subsequent suspense movement for stock units

Note: If in catch-up mode, we may be analysing the position for some days ago, in which case we need to consider deleted stock units as well as current ones. This means iterating the DelStockUnits collection as well as the StockUnits collection.

The suspense movement for each stock unit is obtained as follows:

- Read the StockUnits.*sss* or DelStockUnits.*sss* persistent object, and use the 'CAPRolloverTrailer' attribute to read the RolloverTrailer message that identifies the start of the current CAP/TP (not BP) for the stock unit. If no such message is found, then the stock unit is ignored.
- Compare the message id of that trailer with the EOD mark for the day to see if the rollover was completed on or before that day. Also compare the Riposte <Date:> and <Time:> attributes of the trailer with those of the office RolloverTrailer (see above) to see if the stock unit rollover occurred before the office.
- If either condition is false, use the 'PreviousCAP' attribute to follow the chain of RolloverTrailer messages until an appropriate one is found. If no appropriate one is found, the stock unit is ignored.
- Use the 'Rollover' attribute from the RolloverTrailer message to read the Rollover message
- Get the 'Mark' attribute from the Rollover message for use as the start-point for the scan
- Search for messages between the rollover start mark and the EOD mark for the day, using search criteria derived from the 'Queries.SUSuspense' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2), filling in the stock unit id as necessary.

For each of these scans, we sum the value of 'EPOSSTransaction.SaleValue' attribute. The total sum of the office and stock unit scans is the overall suspense position, and is -ve for a nett Suspense loss, or +ve for a nett Suspense gain.

4.1.8.4.2 Calculating the Local Suspense

This calculates the current Local Suspense position, and is similar to Calculating the Suspense (4.1.8.4.1), except that there is no brought forward local suspense for the office.

Note: Under normal circumstances, local suspense movements for a given TP will occur as part of stock unit rollover, i.e. after the office rollover into that TP. However, it is technically possible for a user to clear local suspense from the Housekeeping menu when ahead of the office, so we must scan for movements from the point at which each stock unit rolled into the office TP, just as for normal suspense.

Therefore, the overall solution approach is

- Determine when the office rolled into the current CAP/TP (relative to the day being analysed).
- For each of the stock units, add in the local suspense movements since the stock unit rolled into that TP (up to the end of the day being analysed).

The message store scans and selection of the appropriate office and stock unit rollover trailers must of course take account of the day for which the analysis is being done. This means walking the chain of trailers back from the relevant persistent object until one is found that occurred on or before the day in question.

The logic is therefore as follows :

<u>Determine the start of the office CAP/TP</u>

- Read the EPOSSCAP.Office persistent object, and use the 'CAPTrailer' attribute to read the CAPTrailer message. If no such message is found, then the brought forward suspense will be assumed to be zero.
- Compare the message id of that trailer with the EOD mark for the day to see if the office rollover was completed on or before that day. If not, use the 'Previous' attribute to follow the chain of CAPTrailer messages until an appropriate one is found. If no appropriate one is found, then the entire local suspense movement will be assumed to be zero.
- Use the 'RolloverTrailer' attribute from the CAPTrailer message to read the appropriate RolloverTrailer message.

<u>Determine any subsequent local suspense movement for stock units</u>

Note: If in catch-up mode, we may be analysing the position for some days ago, in which case we need to consider deleted stock units as well as current ones. This means iterating the DelStockUnits collection as well as the StockUnits collection.

The suspense movement for each stock unit is obtained as follows:

- Read the StockUnits.*sss* or DelStockUnits.*sss* persistent object, and use the 'CAPRolloverTrailer' attribute to read the RolloverTrailer message that identifies the start of the current CAP/TP (not BP) for the stock unit. If no such message is found, then the stock unit is ignored.
- Compare the message id of that trailer with the EOD mark for the day to see if the rollover was completed on or before that day. Also compare the Riposte <Date:> and <Time:> attributes of the trailer with those of the office RolloverTrailer (see above) to see if the stock unit rollover occurred before the office.

- If either condition is false, use the 'PreviousCAP' attribute to follow the chain of RolloverTrailer messages until an appropriate one is found. If no appropriate one is found, the stock unit is ignored.
- Use the 'Rollover' attribute from the RolloverTrailer message to read the Rollover message
- Get the 'Mark' attribute from the Rollover message for use as the start-point for the scan
- Search for messages between the rollover start mark and the EOD mark for the day, using search criteria derived from the 'Queries.SULocalSuspense' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2), filling in the stock unit id as necessary.

For each of these scans, we sum the value of 'EPOSSTransaction.SaleValue' attribute. The total sum of the office and stock unit scans is the overall local suspense position, and is -ve for a nett Local Suspense shortage, or +ve for a nett Local Suspense surplus.

4.1.8.4.3 Summarising the Adjustments

This calculates the nett value adjustment made to cash declarations by means of the 'Add/Remove Cash' functions during the day (see section 4.1.10)

The mechanism to do this is a Riposte message store scan from the previous day's EOD mark up to the current day's EOD mark, summing the <EPOSSTransaction.SaleValue:> attribute for all local adjustment event transactions, as described in 4.1.10.5.

The actual search criteria will not be hard-coded, but obtained from the 'Queries.Adjustment' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2).

The result of this is a figure that is -ve for a nett addition of cash (i.e. cash declaration –ve value increased), and +ve for a nett removal of cash (i.e. cash declaration –ve value reduced).

4.1.8.4.4 Counting the Transaction Corrections Processed

The count of Transaction Corrections that have been processed on the current Trading Day will be determined by counting the number of such messages (see 4.10.4.2).

The actual search criteria will not be hard-coded, but obtained from the 'Queries.TCProc' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2.)

4.1.8.4.5 Counting the Transaction Corrections Outstanding

The mechanism for selecting Outstanding Transaction Corrections differs from the simple search between EOD markers that is described in the sub-sections above.

It must search for Transaction Correction messages (see 4.10.4.1) from the beginning of the message store to the End of Day marker for the current Trading Day.

The actual search criteria will not be hard-coded, but obtained from the 'Queries.TCReq' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2.). As noted there, restricting the search to messages originating from a correspondence server will optimise the scan.

It must also search for Processed Transaction Correction messages (see 4.10.4.2) from the beginning of the message store to the End of Day marker for the current Trading Day.

The actual search criteria will not be hard-coded, but obtained from the 'Queries.TCProc' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2.). As noted there, restricting the search to messages originated locally will optimise the scan.

For each of the Transaction Correction messages found, the values of <Data.Ref:> and <Data.Iter:> will be noted in memory (e.g. stored in a temporary array).

For each of the Processed Transaction Correction messages found, the values of <Data.Ref:> and <Data.Iter:> will also be noted in memory (e.g. stored in a second temporary array).

At the end of the message store scan(s), the values in the two arrays can be compared to find the outstanding corrections, and to count them if necessary.

See also section 4.5.9.1

4.1.9 Cash Variance Report

Note: As a result of CP 3980, the Cash Variance Report has been suppressed by removing the buttons that produce or re-print it. However, the underlying logic to support it remains in place and is described in this document for reference.

The report is implemented as part of BESReports and operates on data that is pertinent to a specific week number. This week number will be the current week in the year for the latest report figures or may be a prior week in the year (or a previous year) for report re-prints.

Therefore, when printing a current report, the current week number will be evaluated from the system date. For retrospective prints, the set of available Office Variance History objects will be used to generate a pick-list of available dates sorted from oldest to newest, from which the week number (or date) will be passed to the report as part of the report criteria.

Note that retrospective print should not be offered for the current week (even if a Office Variance History exists) because the data will not be frozen until the end of the week.

If reporting the current week (i.e. this is not a retrospective print), then data will not yet be available for today and subsequent days. The columns for these days will therefore be reported as blank (rather than “n/a” or “x”). However, if the current day is a Thursday, then data will not yet be available for any part of the week. Therefore, when run on a Thursday, the previous week will be reported instead (i.e. the reporting week is deemed to be the current week – 1).

All of the Weekly Variance Persistent Objects for the reporting week will be retrieved. These are identified by the Collection name = ‘Variance_*ww*’ as described in section 4.10.3

The layout of the report is described in EA/IFS/012 and the report sections will be built-up in the following manner:

Heading	The week always begins on a Thursday and ends on a Wednesday. The dates shown in the heading are calculated from the Reporting Week number. Since the week number itself is not shown, it is convenient for programming that this is a VB calendar week, and is not related to the POL accounting calendar.

Cash Variances	The cash variance will be shown for each Stock Unit, and Stock Units will be sorted in alphabetical order. The data shown on the report will be taken from the 'Data.BFwd.Var' or 'Data.Decl_ddd.Var' attribute of the relevant Stock Unit Variance History persistent object. The value is first looked up from the 'foreground' object. If that object does not exist, or the container is empty, the 'background' object is looked up instead. 'n/a' will be shown if the data container does not exist or is empty. 'x' will be shown if the data container status is bad (i.e. Data.BFwd.S or Data.Decl_ddd.S is 'bad'). See 4.1.9.1 below. Otherwise, the value will be shown with its stored sign (i.e. –ve means shortfall) The 'Total' line is the sum of the variances in the column above or is shown as 'x' if any of the figures in the above column is 'x'. If any figures are 'n/a', those values are ignored in calculating the total.
Derived Figures	The derived figures will be shown for each Stock Unit., and Stock Units will be sorted in alphabetical order. The data shown on the report will be the respective ('Data.BFwd.Var' + 'Data.BFwd.Value') or ('Data.Decl_ddd.Var' + 'Data.Decl_ddd.Value') values for the relevant Stock Unit Variance History persistent object. The values are first looked up from the 'foreground' object. If that object does not exist, or the container is empty, the 'background' object is looked up instead. 'n/a' will be shown if the data container does not exist or is empty. 'x' will be shown if the data container status is bad (i.e. Data.BFwd.S or Data.Decl_ddd.S is 'bad'). See 4.1.9.1 below. Otherwise, the value will be shown with the reverse of its stored sign (since that is always -ve) The 'Total' line is the sum of the derived figures in the column above, or is shown as 'x' if any of the figures in the above column is 'x' If any figures are 'n/a', those values are ignored in calculating the total..

Declared Figures	The declared figures will be shown for each Stock Unit, and Stock Units will be sorted in alphabetical order. The data shown on the report will be taken from the Data.BFwd.Value or Data.Decl_ddd.Value attribute of the relevant Stock Unit Variance History persistent object. The value is first looked up from the 'foreground' object. If that object does not exist, or the container is empty, the 'background' object is looked up instead. 'n/a' will be shown if the data container does not exist or is empty. 'x' will be shown if the data container status is bad (i.e. Data.BFwd.S or Data.Decl_ddd.S is 'bad'). See 4.1.9.1 below. Otherwise, the value will be shown with the reverse of its stored sign (since that is always -ve) The 'Total' line is the sum of the declared figures in the column above, or is shown as 'x' if any of the figures in the above column is 'x'. If any figures are 'n/a', those values are ignored in calculating the total.
SU Breakdowns	These figures will be listed in Declaration ID numeric order within Stock Unit alphabetic order for shared stock units. The data will be extracted from the 'Data.BFwd.Value' or 'Data.Decl_ddd.Value' attribute of the relevant Till Declaration History persistent object. The value is first looked up from the 'foreground' Till Declaration History object. If that object does not exist, or the container is empty, the corresponding 'background' object is looked up instead. 'n/a' will be shown if the data container does not exist or is empty. 'x' will be shown in this section of the report if the declaration status is 'bfwd' and the variance status in the corresponding Stock Unit Variance History object is 'bad'. See 4.1.9.1 below. Otherwise, the value will be shown with the reverse of its stored sign (since that is always -ve)
Known Discrepancies	The commited discrepancies figures are taken from the Stock Unit Discrepancy History. The suspense and local suspense figures are taken from the Office Variance History. The values will be shown with their stored sign (i.e. –ve means shortfall)
Adjustments	These figures are taken from the Office Variance History. The values will be shown with the reverse of their stored sign (since –ve stored sign means cash added)
Txn Corrections	These figures are taken from the Office Variance History.

Note: Some values need to have their sign reversed before being output on the report, because of the convention within Horizon that stock values are held internally with –ve value. See 4.10.3 for details.

When a stock unit is added to the Branch mid-week, or a till declaration is added to a stock unit, the variance report will show 'n/a' for the portion of the week when the stock unit or till declaration did not exist. This is detected by the fact that neither the foreground nor background Stock Unit Variance History or Till Declaration History objects contain values for the day.

There are obscure situations where the variance and declaration history objects in a particular week might be completely absent for the office as a whole, or a particular stock unit.

For example, following point 20 migration, or if the EOD process has not run the previous night and it was the first of the week (ie Thursday night).

If the Office Variance History object is missing, the report will be abandoned with a bland ‘report is not available’ error message.

If the variance or declaration history objects for a particular stock unit are missing, it will simply not appear on the variance report until such time as a declaration has been made (which will normally be forced when a user logs on to the stock unit for the first time).

When a stock unit is deleted partway through a week, the variance report will continue to show the value brought forward from the final declaration until the end of the week. (This value will always be zero because a Stock Unit must be emptied, declared and rolled over into a new CAP/TP before it can be deleted.)

4.1.9.1 Determination of Stock Unit Variance History status

Care must be taken when reading the Stock Unit Variance History status to decide whether to output an ‘x’ or not.

The normal strategy (as described in 4.10.3) is to read attribute values from the ‘foreground’ data container in preference to the ‘background’ data container.

However, this does not work for the case where figures have been brought forward in-day by Add/Remove Cash (and thereby written to the foreground container), but subsequently downgraded to status ‘bad’ at end-of-day by Maintain Office Variances (and thereby written to the background container).

If the Cash Variance Report logic were to simply read the foreground data container, it could wrongly regard the figures as valid (by virtue of an out of date ‘bfwd’ status). The strategy must therefore be slightly modified :-

- If foreground data container exists, use its values in preference to the background container.
- However, if both containers exist, and the foreground container status is ‘bfwd’, use the background status in preference (since it may imply a downgrade to ‘bad’).

Other than the status, values should still be taken as normal from the foreground container, in order to reflect the effect of all Add/Remove Cash adjustments.

See also Stock Unit Variance History (4.10.3.2.1).

4.1.10 Add/Remove Cash

If a cash variance has been identified via the Cash Variances report (or in any other way), then the user will be able to rectify this variance by either Adding Cash (adding money to a Till) or by Removing Excess Cash (removing excess money from a Till). These events will be recorded on the system by the use of two new functions that will be invoked via additional menu buttons on the 'SBAL L2/S4 Stock Balancing' menu (see the Menu Hierarchy document SD/SPE/016).

The dialogue following either of these buttons is identical.

4.1.10.1 Shared SU Till Identification

Where the variance correction is being applied to a Shared Stock unit, then the user will be required to identify to which Declaration Id (Till) the correction is being applied. This will be achieved by presenting the user with a list of Declaration Ids for the current stock unit in the same manner as described in section 4.1.6.1

4.1.10.2 Duplicate Amendment Check

Following selection of a particular declaration id, the declaration should arguably be locked so as to prevent more than one person reviewing and updating the declaration at the same time.

However, it is rather unlikely that two people would choose to declare or adjust the same physical cash asset at the same time, and the worst that can happen is that some adjustment or declaration is effectively ignored. Since the implementation effort appears to outweigh the actual benefit, locking will not be done.

However, a check will be made that 'Add/Remove Cash' has not already been recorded since the last declaration event. This is done by checking for a non-empty and non-zero value in the 'Adjustment' attribute of the Cash Declaration Persistent Object for the till (see 4.10.1).

If no such object even logically exists, then no declaration has been made, and it is not meaningful to attempt to 'Add/Remove Cash'. The request will therefore be rejected as described in EA/IFS/012.

4.1.10.3 Amount Entry

A standard dialogue will be presented to the user to enter the Amount of cash to be made good or removed from the Till.

The figure entered must be greater than zero and must be a monetary value expressed as pounds and pence. In practice the calculator entry control will limit it to £99,999,999.99.

If removing excess cash, the adjustment will be rejected if the declaration value would become negative.

4.1.10.4 Writing the Adjustment Event

An 'Excess Cash Removed' or 'Cash Shortage Made Good' event (see 4.11.3) will be written as a result of removing Add/Remove Cash.

In both cases, an attribute <P1:> will be written with the event message containing the amount to be made good or removed (this value will always be positive, representing pounds and pence in the format N.NN).

For shared stock units, an additional attribute <P2:> will record the Declaration Id against which the event was written.

4.1.10.5 Writing the Adjustment as a Local Transaction

As described in EA/HLD/005 and EA/IFS/011, the stock unit balance report needs to include the nett cash adjustment since the start of the balancing period.

In principle, this can be determined by summing the <P1:> attributes for all the adjustment event messages. However, to avoid the balancing process having to manually re-scan the message store, it is advantageous to write a message that can be summarised by DataServer along with the other normal transactions as part of normal balancing. The nett adjustment can then simply be read out of an appropriate node of the tree for population onto the report.

Unfortunately the format of event messages is incompatible with this (because events already have primary mappings that are ignored by DataServer during balancing).

Event messages (see 4.11.3), are normally harvested as 'TranType=E' messages by TPS and transformed to feed POL MIS. That transformation (see AD/DES/041) converts each event id into the corresponding POL event product, and synthesises a POL 'event' transaction mode (mode 20).

Therefore, an additional ‘local transaction’ message will be written, based on the Horizon product corresponding to the POL event product, and the primary mappings for these products will be made to map under the ‘3017’ tree that DataServer captures during balancing, but under a distinct subtree so that it does not affect the balancing itself. For more details, see EA/HLD/005.

This event product is obtainable from the additional <PN:> attribute of the relevant EPOSS Event Definitions (see 4.11.3).

(This transaction is also used for Summarising the Adjustments (4.1.8.4.3) performed as part of the Calculate Office Variance end-of-day task).

The value of the transaction represents the adjustment, and the sign is determined by the accounting sense (session effect) of the Horizon product. For ‘Make Good’, the transacted value will be –ve (increasing the –ve declaration value), and for ‘Remove Excess Cash’ it will be +ve (decreasing the the –ve declaration value).

The transaction is executed in a new Horizon mnemonic mode ‘EVNT’, that corresponds to the existing POL ‘event’ mode (20). This is because the POL products are already defined to be transactable in that mode, so the Type A Horizon product will be transactable in that mode (and only that mode).

No settlement transaction will be required because the ‘local transaction’ has no effect on the branch balance status, and is intended to be ignored outside the counter.

The transaction will have ‘TranType=L’ so that it is ignored by TPS, and the EPOSSTxnSelection.Criteria01 selection criteria controlling End-of-day EPOSS reconciliation will be amended to ignore it as well (See 4.11.6.2.1).

It should also not appear on the Transaction Log report, so the desktop buttons that produce that report will be amended to ignore it as well. (See 4.11.5.1)

4.1.10.6 Updating the Declaration

In addition to this, the adjustment event data will be used to update the most recent cash declaration performed for the stock unit (and Declaration Id for Shared Stock Units). This is done by updating the Cash Declaration Persistent Object (see 4.10.1) associated with Declaration Id and Stock Unit id.

A new attribute '<EPOSSTransaction.Adjustment>' will be updated to record the additional Amount Made Good (-ve amount) or Excess Cash Removed (+ve amount).

In addition to updating the Cash Declaration Persistent Object, the stock unit must be marked as 'dirty' so that inactive stock unit rollover is prevented, just as for a normal Cash declaration. This is achieved by setting the 'BalanceStatus' attribute to 'Dirty' in the StockUnit persistent object (see EP/DES/002)

4.1.10.7 Updating the Variance History

Following this, the system must update the variances persistent objects described in section 4.10.3.

4.1.10.7.1 Individual Stock Units

For Individual stock units, the Stock Unit Variance History persistent object described in section 4.10.3.2 will be updated.

Where an individual Stock Unit has not declared today, then yesterdays stock unit cash declaration and variance, recorded as a Stock Unit Variance History object, should be brought forward if available, to act as a basis for the adjustment. However, the declaration and variance should not be brought forward if trading has occurred since then.

(It should also not be brought forward if the associated Stock Unit has already been logically deleted from the system. However, that cannot happen in this case since the user is still attached to it!)

When updating a Stock Unit Variance History in this manner, opportunity must be taken to prune old markers, as described in 4.10.3.2.2, in order to contain the total size of the message to the Riposte limit of 2048 characters.

Note that if no declarations are found for yesterday, the effect is simply that nothing is brought forward to today. See 4.1.9 for how this affects the Cash Variance Report – in particular during the first week after Point 20 migration or creation of a stock unit.

The bring forward action described below produces a 'Foreground' value. (Similar logic can also occur at end of day as described in 4.1.8.2, producing a 'Background' value). For more information about 'foreground' and 'background' objects, see 4.10.3.

The in-day bring forward logic can be expressed in Pseudo-Code as shown below:

```
If yesterday's declaration (Data.Decl_ddd) is missing
Then
    Do nothing

Else If SU has declared today { i.e.today's  Data.Decl_ddd.S = 'ok' }
Then
    { declaration is ok for today => no need to check for subsequent trading }

Else If declaration brought forward and still valid { i.e.today's  Data.Decl_ddd.S = 'bfwd' }
Then
    If the SU has traded since the declaration { i.e. between today's Data.Decl_ddd.Mark and now }
    Then
        Set today's declaration status (Data.Decl_ddd.S) to 'bad'
    Endif
```

```
Else If declaration brought forward but invalid { i.e.today's  Data.Decl_ddd.S = 'bad' }
Then
    { declaration status already 'bad' => no need to check for subsequent trading }

Else {declaration needs to be brought forward }

    If yesterday's declaration status (Data.Decl_ddd.S) is 'bad'
    or the SU has traded since yesterday's declaration { i.e.between yesterday's Data.Decl_ddd.Mark and now }
    Then
        Set today's declaration status (Data.Decl_ddd.S) to 'bad'
    Else
        Set today's declaration status (Data.Decl_ddd.S) to 'bfwd'
    Endif

    Set today's Data.Decl_ddd .Date To current date/time
    Set today's Data.Decl_ddd .Value to yesterday's value
    Set today's Data.Decl_ddd .Mark to yesterday's mark
    Set today's Data.Decl_ddd .Var to yesterday's Var
    Set today's Data.Decl_ddd .User to current user

Endif
```

By 'traded', we mean there have been transactions that affect the cash position for the stock unit. This is determined using search criteria derived from the 'Queries.Traded' attribute of the Type C reference data object EPOSSStockUnit.OfficeVariances (see 4.11.6.3.2), filling in the stock unit id as necessary.

If today's Stock Unit Variance History now exists, its <Data.Decl_ddd.Value:> and <Data.Decl_ddd.Var> attributes must be adjusted by the amount made good or removed.

Since a -ve adjustment represents an amount made good, this means adding the adjustment to the declared value, but subtracting it from the variance (since –ve variance means a shortfall).

Note: If a cash declaration has not yet been made today and bringing forward the figures as described above is not successful, then the recording of the adjustment event against the stock unit will have no effect on the variance report.

See also Stock Unit Variance History (4.10.3.2).

4.1.10.7.2 Shared Stock Units

For Shared stock units, the Till Declaration History persistent object described in section 4.10.3.1 will be updated.

Note that the corresponding Stock Unit Variance History is NOT updated, because (1) the updated till declaration reflects current status, but the existing variance describes a different point in time. (2) there is no guarantee that the set of applicable declarations is the same as when the original variance was calculated.

First of all, if there is no Till Declaration History for today for the declaration id, it must be brought forward from yesterday if possible.

The logic will therefore read the Declaration Id for the Stock Unit for yesterday, trying to read either the 'Foreground' till declaration (as created by in-day logic), or failing that the 'Background' till declaration object (as brought forward by EOD).

A corresponding 'Foreground' till declaration object is created for today, based on yesterday's till declaration object (Foreground or Background) with the following differences:

Data.Decl_ddd.Date = Current Date/Time
Data.Decl_ddd.User = current user
Data.Decl_ddd.S = 'bfwd'

If today's Till Declaration History now exists, its <Data.Decl_ddd.Value> attribute must then be increased by the amount made good, or decreased by the Excess removed.

For more details, see Till Declaration History (4.10.3.1)

Note: A till declaration for a zero amount of cash should not be carried forward into a new week. In this way, Declaration Ids that are no longer used will be retired from variance reporting after at most one week of being dormant

Note: If no declaration is found for yesterday, the effect is simply that nothing is brought forward to today. See 4.1.9 for how this affects the Cash Variance Report *– in particular during the first week after Point 20 migration or creation of a stock unit.*

.

4.1.10.8 Reporting Adjustments on the Balance Report

The sum of adjustments made during a Trading Period will be reflected on the Balance Report at the end of the period. The mechanism and format of this will be described in EA/HLD/005.

4.1.10.9 Re-Declaration Following Adjustment

The Cash Declaration process allows a previous declaration to be updated rather than starting a new declaration from scratch. If this option is chosen following a 'Made Good' or 'Excess Cash Removed' event, then the amount adjusted will not be reflected in any previous declaration details.

However, on selecting an existing declaration, any current amount 'made good' will be presented as part of the dialogue title as a reminder. See EA/IFS/012 for details.

4.1.11 Cash Auto-Declaration on Inactive Rollover

Currently, inactive stock unit rollover takes care to produce new ONCH figures for the current week based on the most recent ONCH declaration. This ensures that a denominational breakdown of cash holdings remains available even when a stock unit remains inactive for a long time.

This logic will be replaced by a similar propagation of S80 Cash Declaration(s), and associated update of the Stock Unit Variance History.

4.1.11.1 Propagating the Cash Declaration

During transition to S80 at migration point 20, there is no guarantee that the original pre-S80 Cash Declaration details are still present in message store, and even if they are, that they are relevant to the currently declared position for the stock unit. However, the most recent ONCH declarations can provide the necessary detail (courtesy of the pre-S80 propagation logic).

The propagation therefore consists of

- selecting the appropriate Cash Declaration Persistent Objects or ONCH Declaration Persistent Objects for the stock unit as described below
- finding the associated DrawerItemDeclaration and DrawerItemDeclarationDetail messages
- rewriting those messages (with new 'DrawerItemDeclarationId')
- creating/updating the corresponding Cash Declaration Persistent Object.

To select the appropriate Cash Declaration Persistent Objects, iterate all "*tt*_4_1" objects in the "Declaration_*sss*" collection, where "*tt*" is a till (declaration id) and "*sss*" is the stock unit id (see 4.10.1), ignoring any pre-S80 ones (i.e. ones with no "Adjustment" attribute). The declaration trailer and associated messages are then found via the "DrawerItemDeclarationMsgId" attribute.

If no S80 declarations are found, select the appropriate ONCH declarations, by iterating all "*tt*_4" objects in the "ONCH_*ww*_*sss*" collection, where "*tt*" is a till (declaration id), "*ww*" is the current VB week number, and "*sss*" is the stock unit id. Within each object found, select the latest drawer item declaration message id from the 'CarryFwd' attribute, or if that attribute is missing/empty, from the 'BrFwd' attribute. If both attributes are empty/missing, the declaration must be ignored. See 4.10.2 for details.

Finding the DrawerItemDeclaration and DrawerItemDeclarationDetail messages means using the msg id from the Cash Declaration Persistent Object or ONCH Declaration Persistent Object to read the DrawerItemDeclaration message, then using the 'DrawerItemDeclarationId' attribute to construct a start/end marks and selection criteria for a scan to read the DrawerItemDeclarationDetail messages.

Creating/updating the Cash Declaration Persistent Object means updating the CAP, BP, DrawerItemDeclarationId and DrawerItemDeclarationMsgId fields. If creating a declaration from ONCH, the 'adjustment' should be set to zero, otherwise it should be left as is. The Mark field should be set to the Rollover mark.

Note that propagating a declaration does not mark the stock unit BalanceStatus as 'dirty', otherwise it would prevent the stock unit from being rolled forward as an inactive stock unit.

4.1.11.2 Updating the Stock Unit Variance History

As described for Cash Declaration (individual stock unit) and Check for Variances (shared stock unit), the Stock Unit Variance History must be written to record the declared, derived and variance values.

The original declaration (with or without subsequent adjustment) must have been made before the last rollover (otherwise the stock unit would have been marked as dirty and ineligible for inactive rollover). Therefore, the current variance will be zero, and the derived value can be assumed to be the same as the declared value. The declared value is the sum of the copied declarations (including adjustments).

The Status should be set to ‘ok’ (just as if a real declaration had been made), and the Mark should be set to the Rollover mark.

(When updating a Stock Unit Variance History in this manner, opportunity must be taken to prune old markers, as described in 4.10.3.2.2, in order to contain the total size of the message to the Riposte limit of 2048 characters.)

4.1.12 Pruning of Cash Declaration Objects

As described in 4.1.1, non-zero S80 cash declaration persistent objects will now be retained indefinitely rather than being pruned after two CAP/TPs.

Cash Declarations with zero value and zero adjustment will however be logically deleted by EPOSSDeclare at desktop start-up on the day after they were zeroed, with the date comparison based on the standard Riposte 'Date' attribute.

Note: The declarations are not logically deleted immediately on zeroing because of the need to provide at least a dummy declaration for rollover checking purposes.

Note: Declarations are not logically deleted on the same day as they were zeroed to avoid problems if a desktop is restarted during the day..

As with existing CAP-based declaration pruning, the removal is logical (ie: the addition of an attribute 'LogDel' to indicate it is logically deleted) rather than physical. This approach avoids a known Riposte problem (PC97077) if the same object name is reused subsequently.

Non-cash declarations and pre-S80 cash declarations (distinguishable by lack of '<EPOSSTransaction.Adjustment:>' attribute – see 4.10.1) are unaffected by these changes – they will continue to be logically deleted after two CAP/TPs as prior to S80.

4.1.13 Pruning of Variance History Objects

Office Variance History, Stock Unit Variance History, Stock Unit Discrepancy History and Till Declaration History persistent objects that are sufficiently old will be physically deleted by EPOSSDeclare at desktop start-up.

The decision on deleting a particular object is based on its Riposte <Date:> attribute (i.e. the date when it was last written).

The age at which these objects become eligible for physical deletion is defined by the 'VarianceHistoryExpiry' attribute of the Type C reference data collection 'CounterConfigParams' (see EA/HLD/005), and defaults to 42 days (i.e. greater than a Trading Period).

(Unlike Declaration objects, there is no need to use logical deletion because the object names (actually the collection names) include a week number. This means that the known Riposte problem (PC97077) on reusing the same object name around the time of DeletedObjectVersionExpiry will not occur because the reuse is not until the following year.)

4.2 Stock Declaration and Adjustment

Stock declarations can be performed on a regular basis using the existing functionality that allows declarations to be made and any discrepancies to be checked. This functionality is restricted to Shared stock units only.

When making a declaration, the user is invited to make a new declaration (by entering a new declaration id) or to update an existing declaration (by selecting an existing declaration id).

Following the choice of declaration, a pick list of stock items is presented and the quantity of stock items for each volume-based product may be entered and updated, just as prior to S80.

Adjacent to each product is listed the price of the product. At S80, this will change such that the Loss Price of the product is shown instead. However, since not all products have a separate loss price, then the loss price will be displayed when known, otherwise the sale price will be displayed. When updating an existing declaration, the 'value' column shows the effect of multiplying the 'loss price' by the volume.

Stock level adjustments are handled similarly. The Loss Price of the product is shown instead of the retail price, and the 'value' column shows the effect of multiplying the 'loss price' by the volume.

A new optional attribute '<AP:>' (Adjustment Price) will be added to the EPOSSProducts collection. If a numeric value is specified for this attribute, then this value will be displayed and used in preference to the '<RP:>' (Retail Price) attribute during the stock declaration and adjustment processes.

Any difference between the declared quantity of stock and the system-derived quantity of stock must be corrected or a discrepancy transaction will be generated as part of the stock unit rollover process. The discrepancy transaction takes the form of an increase/reduction in the quantity of product. If the number of product declared is greater than the derived quantity, then the value of the difference will be posted to the 'gain' discrepancy product. If the number of product declared is less than the derived quantity, then the value of the difference will be posted to the 'loss' discrepancy product. These products are determined from the EPOSSStockUnit.Parameters persistent object.

The value that will be posted will be a multiple of the Adjustment Price attribute <AP:> if an adjustment price is specified otherwise it will be a multiple of the Retail Price <RP:>. See also 4.4

Note: As described in EA/HLD/005, the transaction that changes the stock volume uses its tertiary mapping to account for the change on the sale/receipt side of the balancing tree just as for a normal sale. The effect of that could be to produce a line in the balance report that represents a mixture of transactions at retail and adjustment prices. The volume vs value information on that line may therefore be rather meaningless.

4.3 Non-Value Stock Declaration and Reporting

Since LFS Weekly Stock reporting is no longer needed, the whole concept of Non-Value stock will be removed from the system.

This involves

- Suppressing the LFS Weekly Stock Report (see section 4.6.1.1)
- Removing the 'Declare Non-Value Stock' menu option from the stock unit balancing menu.
- Removing the 'Confirm Non-Value Stock' menu option from the office balancing menu
- Removing the option to remit out non-value stock from the 'Remit Out ADC' menu
- Removing the checks for completed non-value stock declarations at stock unit rollover
- Removing the checks for completed non-value stock confirmation at office rollover

On the assumption that the LFS report has been suppressed beforehand (see section 4.6.1.1) the menu options could be suppressed from migration point 20 onwards.

However, to avoid confusion, the menu changes and removal of checks will all be made at the point that the Stock Unit and Office cut over to TP mode at point 50.

The menu changes will be achieved by rollover-triggered softlaunch, as described in EA/HLD/008.

The non-value stock confirmation check at office rollover is controlled by an <LFSNVStockCheck:True> parameter to the 'Cash Account' menu item, so this can also be suppressed by softlaunch as part of the transition to TP mode.

The non-value stock declaration check at stock unit rollover is controlled by the presence of Type C reference data collection member 'DrawerItems.5'. Unfortunately, this collection cannot be modified dynamically at point 50 when rolling over a stock unit without affecting other stock units. Instead, the EPOSSDeclare code that controls access to it will skip the presence of that element if the stock unit is already in TP mode.

4.4 Rollover Discrepancy Checking

Discrepancies are currently checked as part of stock unit rollover, and also via the Discrepancies function on the Stock Balancing menu.

All discrepancies are checked and reported on (not just cash).

Additional logic is required as follows

- Cash discrepancy checking must now take account of any 'Add/Remove Cash' actions, as described in section 4.1.4.2.
- Cash discrepancy checking must now take account of all S80 cash declarations, regardless of the BP in which they were made, and must ignore any pre-S80 cash declarations. See 4.1.1
- For fixed price products, stock discrepancy checking should be volume rather than value based once stock-by-volume trading is in force – i.e. after migration point 50.
- When a stock discrepancy is found as part of rollover, the total mismatch value is converted into a committed cash discrepancy. The value calculation should use the adjustment price rather than the retail price for fixed price products. See also 4.2.
- Regardless of the result of the check, cash discrepancy checking **during rollover** for a shared stock unit should update the Stock Unit Variance History persistent object for the current day (see 4.10.3.2) before committing any discrepancy. This is effectively equivalent to performing the Check for Variances immediately beforehand.

Note: The Stock Unit Variance History should not be updated if the check is merely being done as part of the Discrepancies function, because nothing is being committed, and the declarations have not been confirmed as appropriate for variance calculation.

Note: When updating a Stock Unit Variance History in this manner, opportunity must be taken to prune old markers, as described in 4.10.3.2.2, in order to contain the total size of the message to the Riposte limit of 2048 characters.

4.5 Transaction Correction Processing

Transaction Corrections are messages that are distributed to Branches if the branch has accumulated values in the Suspense account that now requires intervention by the central accounting body of POL to clear and bring the values to account.[3]

The transaction correction notices are distributed from the POL FS SAP system via the TPS Host system from where the Agent Loader distributes to individual Branches via the Correspondence Servers.

The format and content of the Transaction Correction messages is shown in section 4.10.4.1.

Note that the term 'Make Good' used in this context is not to be confused with the act of making good or removing excess cash as described in 4.1.10.

4.5.1 Identify and List Outstanding Transaction Corrections

Transaction correction processing is invoked by a new menu button on menu 'HKPG L3/S5 Housekeeping' or via an Outstanding Transaction Corrections check that is performed at Logon (see 4.7.4).

These functions are only available to those Roles that can perform an Office Balance. When accessed via the Housekeeping menu, this is controlled by the menu security settings. When accessed from logon checks, this is controlled by explicitly comparing the role of the current user against those defined in the 'Role' attributes of EPOSSStockUnit.TCParams.

If the counter node is isolated, then transaction correction processing must be abandoned with suitable error message as described in EA/IFS/012. This applies on initial entry to the function, or whenever the list of outstanding corrections is recalculated following completion or abandonment of a particular item.

To ensure a consistent approach to the various network topologies, node isolation is determined by asking EPOSSWatchDog for its current LAN status rather than by inspecting the status of individual connections. (This requires a minor interface enhancement to EPOSSWatchdog, since the state information is already available internally)

On entry to the function, a check is made to determine whether there are any outstanding Transaction Corrections, as described in section 4.1.8.4.5. If there are none, then a message informs the user and the process exits. Otherwise, the outstanding Transaction Corrections are listed as described in EA/IFS/012.

4.5.2 Transaction Correction Display

Following selection of an outstanding Transaction Correction, the full details of the Correction will be displayed and a number of processing options will be presented depending on the permitted Modes that are present on the Transaction Corrections message.

At this point, the Transaction Correction should be locked so as to prevent more than one person reviewing and updating a Transaction Correction at the same time.

[3] Other reasons include having errors identified by Clients that need to be corrected

Locking of a particular correction will be implemented using the generic EPOSSCommon NotifyAdd mechanism, which can broadcast ownership of a named resource lock to the other connected nodes, and manage recovery in the case that a node is restarted.

It is also possible that the item cannot be processed because someone has already done so on another counter since the list of outstanding items was displayed.

On completing the processing of an item (or failing to do so because it is locked or has already been processed), control is returned to the list of outstanding items, which must be regenerated to reflect any recent changes caused by this or other counters.

On displaying a particular item, there are a number of optional buttons that are dependent on the permitted Modes that are recorded in the Transaction Correction. The buttons that will be presented on the main Transaction Correction display screen will be as follows:

Mode Mnemonic	Mode	Button Text
MG	Make Good	Accept Now
HD	Plead Hardship	Accept Now
EV	Request Evidence	Seek Evidence
WO	Write Off	Send to P&L
AN	Assign Nominee	Assign To HO
SW	"Stock" Write Off	Stock WO

Where both MG and HD modes are present on a Transaction Correction, then only a single 'Accept Now' button will be used to represent both mode options. This 'Accept Now' button will then navigate to an additional option screen that is described in section 4.5.3. See also EA/IFS/012 for details.

4.5.3 Transaction Correction Additional Options

If the branch is given the option "Make Good" and/or "Plead Hardship", an 'Accept Now' button is presented on the Transaction Correction Display screen described in section 4.5.2.

The 'Accept Now' button will navigate to a screen that allows the Transaction Correction value to be settled to a number of alternative products.

As described in EA/IFS/012, the alternatives are presented as a picklist, even if there is only one option available.

If 'Make Good' is available

The normal option is to 'Make Good' with the product defined in <Data.Instruction> for the Transaction Correction, and the Long Name of that product will be displayed after the text 'Make Good:'

However, if the accounting sense of the correction is TCINV and the product indicated by the Instruction on the Transaction Correction is Cash (product #1), then there may be additional options given.

Although the additional options will be initially be limited to payment by cheque, the actual picklist entries will not be hard coded, but instead derived from items in the 'MakeGoodOptions' attribute of the Type C reference data object EPOSSStockUnit.TCParams. (see 4.11.6.3.1)

If 'Plead Hardship' is available — The option will be given to 'Settle Centrally'

4.5.4 Committing the Transaction Correction

The resulting correction transactions will be constructed as follows :-

Mode	Action
MG (Make Good)	The product defined in the Transaction Correction <Data.Article> will be transacted in sale or receipt mode depending on the value of <Data.AccountingSense>. The transaction will be settled to the product selected from the option list described in 4.5.3.
EV (Request Evidence)	The 'request evidence' option should create no net movement of product yet still generate transaction(s) so that POL FS understands that the Postmaster requires further proof/evidence. The fixed 'Evidence Required' product (as defined by the SettlementProduct for the mode in the ModeParameters collection) will be transacted in sale or receipt mode (depending on the value of <Data.AccountingSense>), and the transaction will be automatically settled to the same product, resulting in no actual net movement of value or volume.
HD (Plead Hardship) WO (Write Off) AN (Assign Nominee)	The product defined in the Transaction Correction <Data.Article> will be transacted in sale or receipt mode depending on the value of <Data.AccountingSense>. By virtue of the mode, the transaction will be automatically settled to the fixed 'Hardship', 'Write Off' or 'Assign Nominee' SettlementProduct for the mode in the ModeParameters collection (or optionally in the ProductModes collection <RData.Data.Modes.Mode.S:>) as per other automatically settled transactions.

SW ("Stock" Write-Off)	Despite the name, this option was originally intended to support correction of Non-Accounting Data (as originally transacted via '10g' transactions). These products have volume but no value, so we will generate a transaction where the value is zero. The mode can also be used to adjust levels for Stock-by-Volume products with non-zero saleprice. In this case, the SW mode will cause EPOSS to use a 'zero value' saleprice (by virtue of the VolSValue for the mode in the ModeParameters collection) when calculating the salevalue. In both cases, the zero value means that no settlement will be required (just as with NAD mode transactions). In the unlikely case that the mode is used for other kinds of products, the transaction correction may or may not succeed. For a variable-price value product, the zero value may provoke a range error. For a fixed-price value product, the value will be determined from the quantity, and a settlement may be required. In that case, the settlement product will be taken from the Transaction Correction <Data.Article>.

In addition, there are common considerations that are common to all modes.

* To support production of the Branch Trading Statement (see also EA/HLD/005), Transaction Correction transactions must contain a secondary mapping that allows them to be counted under a separate part of the balancing tree as part of stock unit rollover. This avoids rollover having to do a separate scan for corrections.

Normally this would be achieved by specifying the mapping on the mode parameters. However, that mechanism applies the mapping if the product can be used as a settlement product (i.e. has product attribute <AS:True>) rather than if it is being used as a settlement in the particular transaction. Since EV mode transacts the same 'Evidence Required' product as transacted and settlement product, the effect would be to count these twice (or not at all).

Therefore, the secondary mappings will be manually applied to the correction transaction, based on the settings given in EPOSSStockUnit.TCParams, and omitted from any associated settlement transaction.

* Transaction Correction transactions will be written with additional attributes. The attributes and values are as follows:

<EPOSSTransaction.BlackBoxData.Ref:>

Contains the Transaction Correction reference number that is contained within the Transaction Correction message as <Data.Ref>

<EPOSSTransaction.BlackBoxData.AddRef:>

Contains the Client reference number that is contained within the Transaction Correction message as <Data.ClientRef>

* If the transaction correction attribute <Data.Qty> is non-zero, it is used as the transaction quantity and <Data.Value> will be ignored (assumed to be zero). For fixed price products, EPOSS will then derive the sale value from the quantity and the product sale price, loss price or zero, according to the mode.

Otherwise, the value of attribute <Data.Value> is used as the transaction value and the transaction quantity is set to '1'. For fixed price products, EPOSS will verify whether the value is appropriate. In particular, it will compare it against sale price or loss price according to the mode. For modes that specify zero, the check will not apply.

See 4.11.1 for details of which price is used for which mode.

* The session effect of the <Data.Article> product does not affect the transaction – the session effect is purely determined by <Data.AccountingSense>. TCINV will cause session effect 'In', and TCCRM will cause session effect 'Out', even if sale of the product would normally have the opposite effect.

Thus (for example), a TCINV/MakeGood for a Cash Deposit would result in a +ve-valued transaction for the 'Cash Deposit' product, paired with a –ve-valued settlement transaction for Cash (or Cheque). The effect would be as if another deposit had been made for the adjustment amount (ie more money taken in).

A TCINV/MakeGood for a Green Giro (which normally has a session effect of 'out') would also result in a +ve-valued transaction for the 'Green Giro' product, paired with a –ve-valued settlement transaction for Cash (or Cheque). The effect would be as if the original giro cheque had been reduced by the adjustment amount (ie less money paid out = more money taken in).

Note: Transactions for non-accounting data products are zero-valued, and use +ve quantity to reflect a 'sale'. If such a product has been 'over-sold' (e.g. data entry error causes misclaim of 1000 'Airsure's), then one might naively expect a TCINV to be issued by POL in order to correct the problem. However, as described above, a TCINV will result in a +ve valued transaction, increasing the count of such items 'sold'. To achieve reduction of the count, a TCCRM must be used instead.

For each Transaction Correction that is processed, a 'Processed Transaction Correction' message will also be written, so that the processed corrections can be tallied against the outstanding ones. The format of this message is described in section 4.10.4.2.

The expiry period of the 'Processed Transaction Correction' should be set such that it expires one day after the original Transaction Correction message is due to expire.

(This then means that the Expiry period of the Transaction Correction may be increased or decreased with impunity.)

4.5.5 Transaction Corrections for Linked Products

If a transaction correction is being applied to a volume stock product, and that product is linked to a product that is also volume stock, then the correction must be applied to that linked product too.

The linkage is represented by the <MP:> attribute of the first product, and its value is the product number of the second product. Note that the linkage is not bidirectional – if a transaction correction is applied to the second product directly, there is no way of knowing (and no requirement to apply a transaction correction to) the first product.

The only known instances of such products are Postal Orders and their linked Fee products.

4.5.6 Transaction Corrections for Bureau de Change Products

Transaction Corrections cannot sensibly be applied to Bureau products such as currency and travellers cheques, because of the need to produce a consistent set of product/margin/commission messages with BDC additional data (such as PQty). Without PQty, the counter does not know the volume of currency transacted (because the transacted product is variable priced).

Restricting Transaction Corrections in this way must be controlled by Type A reference data (i.e. Bureau products must be set up such that they cannot be transacted in the new Transaction Correction modes).

Nevertheless, the counter must be defensive to the possibility that such corrections are applied. The only known problem is that stock unit rollover must still be allowed in the situation that travellers cheques have been bought/sold via Transaction Corrections (and no other means).

Normally, travellers cheque transactions must be reported on prior to rollover, and this is policed via the MandatorySummaries mechanism. However, travellers cheque transactions applied as Transaction Corrections do not (and should not) appear on such reports. The problem is that the MandatorySummaries mechanism detects that travellers cheque transactions have occurred, but the associated reports produce no output, so cannot be cut off in readiness for rollover.

The solution is therefore to restrict the MandatorySummaries search travellers cheque transactions where **EPOSSTransaction.BlackBoxData.S=1** (i.e. sold normally rather than via Transaction Correction). This means adding a SpecificCriteria to the relevant MandatorySummaries definitions. See 4.11.6.4 for details.

4.5.7 Transaction Correction Reversal Control

Transactions created as a result of a Transaction Correction may not be reversed. This will be controlled by a new <DR:> (disable reversal) attribute on the ModeParameters objects for the new transaction correction modes (see 4.11.1).

The checking for this will be implemented within EPOSSCore when the counter attempts an existing reversal. In other words, it will check the mode in which the original transaction occurred, and reject the reversal if the ModeParameters object for that mode has <DR:True> set.

There are already hard-coded checks within EPOSSCore that prevent reversal of certain transactions originally carried out in certain modes. In principle some of these could be replaced by data-driven checks based on <DR:>. However, these checks are also conditional on the product being transacted, or the non-EPOSS application that created the original transaction, so will not be changed at this time.

Therefore we will allow two states

<DR:True>	Reversal not allowed
Other	Reversal **may** be allowed, depending on other conditions

4.5.8 Transaction Correction Validation

As described above, the Transaction Correction transaction will be partially fabricated by the Transaction Correction function and then committed to the stack and then settled to the message store by EPOSS Core via the ‘SettleProduct’ interface without GUI involvement.

EPOSS Core will be changed such that it provides an interface for the validation of transaction data. The EPOSS validation (value range checks etc) will be performed without any user GUI interaction and any errors found as a result of the validation will be returned as status value(s) to the calling function.

The Transaction Correction function will therefore call the new EPOSS validation function to ensure that the transaction correction is valid. Following successful validation, the Transaction Correction function will then call EPOSS Core to save and commit the transaction. In this way, the user will not be harassed by spurious EPOSS messages that might otherwise appear.

If the EPOSS validation function returns with an error status, then the Transaction Correction cannot be saved in the message store. In this instance, an error message will be given to the user (as described in EA/IFS/012). Failing validation, the Transaction Correction transaction will not be committed to the message store however the Processed Transaction Correction message will be saved (section 4.10.4.2). In this instance, the values of the following attributes will vary from that of a successful Transaction Correction:

```
<Settlement:> This will be NULL
<Outcome:>    Value ‘ERROR’
```

4.5.9 Transaction Correction Reporting

Two Transaction Corrections reports are required; a list of Outstanding Transaction Corrections and a list of Processed Transaction Corrections.

4.5.9.1 Outstanding Transaction Corrections

As defined in EA/IFS/012, all Outstanding Transaction Correction messages will be retrieved and printed. The report will be implemented within BESReports.

Outstanding Transaction Correction messages are identified in the manner described in section 4.1.8.4.5.

4.5.9.2 Processed Transaction Corrections

As defined in EA/IFS/012, Processed Transaction Correction messages will be selected between a specified start and end date range, then retrieved and printed. The report will be implemented within BESReports.

Processed Transaction Correction messages are identified in the manner described in section 4.1.8.4.4.

Note that the Branch Trading Statement is also required to report the number of transaction corrections processed, but in that case the period of analysis is a particular trading period rather than a particular day. Although the approach described above would work, it is more convenient to use the existing DataServer-based approach to tally them as part of rollover. See 4.5.4 and EA/HLD/005 for details

4.6 Changes to End-of-Day

4.6.1 LFS End of Day

The LFS End of Day function performs both the Weekly Stock summaries and the Daily Cash Statement summaries. These summaries are collected by the LFS Harvester for onward transmission to SAPADS.

4.6.1.1 Weekly Stock Statements

At migration point 10, the LFS Harvester at the Host will be updated such that the Weekly Stock Statements are no longer harvested. Therefore, the production of the Weekly Stock Statement messages can be stopped at any point after that.

Note that the suppression does not require code change, but merely a change to the reference data controlling the Counter Application Scheduler

The LFS EOD process will only generate weekly stock statements if it is invoked with the parameter 'WEEKLY'. This is driven by the counter scheduler and therefore a simple change to the scheduler reference data is required to stop the Stock Statement being generated. Two persistent objects within the counter scheduler collection therefore need to be removed:

Collection: SchedulerTimeEvents

Object: LFSStockStatement1

Object: LFSStockStatement2

These changes will be made along with the other Type C reference data changes prior to migration point 20.

Note that the LFS EOD process does not *need* to be changed to remove support for weekly declaration, and will not be for this release.

4.6.1.2 Daily Cash Statements

The generation of the daily Cash Statement remains unchanged. Although it was specifically designed to operate using the daily ONCH declarations, the process will also accept Cash Declarations as an alternative to ONCH declarations. Therefore, in the absence of ONCH declarations, the LFS daily Cash Statement generation will continue to operate correctly.

Note that the LFS EOD process does not *need* to be changed to remove support for ONCH, and will not be for this release.

4.6.2 POL FS Summarisation

At migration point 25, the summarisation of data for delivery to POL FS will be performed within the TPS Host system. This means that the summarisation that was introduced at S60 can be removed at any time thereafter.

However, the POL FS Summarisation process (CABSProcess) is responsible for the calculation of the derived branch cash balance that is used during the generation of the Cash Statement by the LFS EOD Process. Therefore it would be ideal if this process was updated such that the POL FS summarisation was removed and the CABSProcess to simply retain the functionality to generate the LFS Branch Cash balance.

Unfortunately, since the removal of the POL FS Summarisation is at some point after the initial code-release, then any change would have to be implemented by a soft-launch mechanism and this would mean increasing the code complexity rather than reducing it.

The POL FS Summarisation process will therefore be retained without change. Simplification of the code will be performed at a future release.

4.6.3 Maintain Office Variances

The maintain Office Variances function is described in section 4.1.8.

This end of day process will be available for implementation from the point of counter software rollout (ie: point 20). However, new counter schedule reference data will be required to invoke it, and this will need to be delivered at the same time

4.6.4 EPOSS Reconciliation

4.6.4.1 Daily Reconciliation

The EPOSS Daily reconciliation process performs a Mini Cash Account reconciliation summary. This should continue until migration point 30 but must cease before migration of the counters at Point 50 since the results would be unpredictable after this time. The daily reconciliation process will therefore be re-written to exclude the Mini Cash Account functionality and will be delivered as a new module that simply includes the Daily Transaction Reconciliation (with a different name to the original).

The simplest migration mechanism is to redeliver the Counter Scheduler reference data to call the new function instead of the old function. This reference data should be made available in RDDS at the data centre at Migration Point 40.

It should be noted that this new Counter Scheduler reference data MUST be made available in RDDS before any point 50 soft-launch data is distributed to the counters. This will ensure that the new Daily Reconciliation process is executing before Point 50 migration.

It should be noted that the auto-settlement products (in the range 112*nn*) will be re-issued as POL controlled TypeA reference data (in the 1-10,000 range) so that they can be harvested and summarised to POL FS. However, these products are not summarised as part of reconciliation totals since they do not belong under the 3017 accounting hierarchy. This should have no affect on the TPS reconciliation since the same products will be prevented from being sent to TIP/MIS and it is this set of transactions (TIP/MIS) that is used to reconcile with the Branch totals.

As a result of removing the Mini Cash Account functionality, the following error conditions are no longer detected or reported

008	Transaction Product has no Cash Account Mapping Object
009	No Mapping for Mode in Product Cash Account Mapping
010	Cash Account Node does not exist for Cash Account Mapping node stipulated
014	Error Reversal Transaction does not have the Omode attribute set

No additional reconciliation will be provided for this release.

4.6.4.2 Weekly Reconciliation

In addition to the Daily Reconciliation function, there is also a Weekly Cash Account reconciliation function. This function also needs to be ceased at the same time as the Daily reconciliation function. The same mechanism will be used to cease this function by removing the Scheduler reference data that invokes the function. This reference data will be issued at the Host at migration point 40.

4.6.5 Protection Against Loss of Data

If a Branch or Stock Unit is not rolled-over into a new Trading Period within the period of expiry of messages within the message store then messages that were written within the current Trading Period will be expired/deleted by the Riposte archiver.

Even if individual Stock Units and the Branch are all rolled regularly, there can still be a problem if a Stock Unit rolled into a new Trading Period a long time ahead of the Branch. This is because many reports (particularly the Suspense Report) need to read messages from the point at which the stock unit originally rolled into the office trading period. This could be almost double the normal rollover period, and hence cause message expiry.

To avoid this situation arising, the Riposte archiver must be turned-off before then to prevent the messages in the message store being lost.

A new end of day process will be implemented on each node (not just the gateway) to disable Riposte archiving if the Branch or Stock Units have not have rolled-over within a pre-defined period, and enable it otherwise.

This end of day process must be available for implementation from the point of counter software rollout (i.e. migration point 20). (See also EA/HLD/005 for how disabling archiving at system startup also relies upon this to re-enable archiving when safe to do so).

The algorithm to check for the rollover condition is described in 4.6.5.1, and the age at which archival must be disabled is defined by the 'RollExpiry' attribute of the Type C reference data collection 'CounterConfigParams' (see EA/HLD/005), and defaults to38 days (i.e. greater than a normal Trading Period, but less than the default message expiry setting)..

There is a special case for single counter outlets. For those sites, a second Riposte service (RiposteMirror) mirrors the message store onto a removable hard disk (for movement to replacement hardware when required). Disabling and re-enabling archival must be done for both services, otherwise the mirror backup will only be partial.

Disabling and re-enabling Riposte archiving implies an update to the registry settings for the service concerned, so requires administrator privileges. The end of day process to do this is therefore **not** run under the Counter Scheduler as usual, but is instead run directly from a Windows AT job and thereby executed with administrator privileges.

To avoid unintended abuse of these privileges, the task runs as a separate DataProtection executable, whose functionality is limited to simple Riposte access.

The AT command is held within a batch file that is executed just once at first system boot

```
C:\AdminCfg\OnceOnly\DataProtection.bat
```

The recurring AT job ensures that the logic is run every day thereafter:-

```
AT 03:15 /every:m,t,w,th,f,s,su cmd /c "<cmd>"
```

where

```
<cmd> = C:\counters\bin\DataProtection.exe Riposte RiposteMirror
```

The launch time is set to 0315 so that it occurs before the nightly ClearDesk (which runs at randomized time between 0330 and 0400), and during a time when Riposte service(s) should normally be available. (Software distribution windows occur between 2030-0130 and could result in Riposte being taken down and up again).

The command-line parameters specify the name of the Riposte services for which archive disabling is to be checked:-

- The first parameter indicates the primary service, and is the one that will be interrogated to see if archiving should be disabled. If this service is unavailable, an NT error event will be generated.
- The second parameter indicates the secondary (mirror) service. Any enable/disable action determined for the primary service will also be applied to this service (if present). Since this service is only present on single counter outlets, an error event will not be generated if it is unavailable.

4.6.5.1 Old Rollover Check

Because of the interaction between stock unit and branch rollovers, there is no need to specifically check for timely branch rollover. Instead, the check is simply whether any of the stock units rolled into the current office trading period longer than a specified time ago.

The mechanism to be used is therefore

(1) Use the 'RolloverTrailer' attribute within the 'Office' object of the collection 'EPOSSCAP' to read the rollover trailer message for the last successful trading period (Cash Account @ S75 and Trading Period @ S80).

The date and time on this rollover trailer message indicate when the office successfully rolled-over into the current trading period.

(2) For each of the Stock Unit objects in the StockUnits collection (except the default Stock Unit), use the 'CAPRolloverTrailer' attribute to read the rollover trailer message for the last successful trading period (Cash Account @ S75 and Trading Period @ S80).

The date and time on this rollover trailer message indicate when the stock unit successfully rolled-over into the current trading period.

(3) If the stock unit rollover trailer message is newer than that of the office, the stock unit has rolled ahead of the office, so use the 'PreviousCAP' attribute of the message to read the rollover trailer message for the previous period (i.e. the one corresponding to the office)

(4) If the difference for any stock unit between the rollover trailer message date and the current system date is more than the number of days specified by 'RollExpiry', then the Riposte Archiver must be disabled.

4.6.5.2 Riposte Archiver Enable or Disable

If the Stock Unit Check or Branch Check reveals that the Riposte Archiver should be disabled, then the Riposte configuration item 'DisableArchiving' must be evaluated.

If it is currently set to zero, then it must be set to 1 and an error event raised via the NT event log (which will then cause Tivoli to alert SMC). This will be achieved via the LogErrorEvent() API provided in EGEventLog.dll. (This is already used in extremis by EPOSSCore, for example).

If it is already 1 (and should remain so), the error event should also be raised so that the event is raised every day until the problem is resolved.

If the result of the above checks is that the Riposte Archiver does not need to be disabled, then the Riposte configuration item DisableArchiving should be reset to zero (if necessary).

This will be achieved via the Riposte API functions 'RiposteSetConfigurationItem' and 'RiposteGetConfigurationItem', but this change to the Riposte configuration will have no effect until Riposte is restarted, which will be at the following day's Clear Desk (c. 03:30).

4.7 Changes to Log-On Checks

4.7.1 Rollover Warning

If a Branch or Stock Unit is not rolled-over into a new Trading Period within the period of expiry of messages within the message store then messages that were written within the current Trading Period will be expired/deleted by the counter archiver.

Even if individual Stock Units and the Branch are all rolled regularly, there can still be a problem if a Stock Unit rolled into a new Trading Period a long time ahead of the Branch. This is because many reports (particularly the Suspense Report) need to read messages from the point at which the stock unit originally rolled into the office trading period. This could be almost double the normal rollover period, and hence cause message expiry.

To avoid this situation arising, the counter archiver will be turned-off before then to prevent the messages in the message store being lost (see 4.6.5), and users will be warned at each logon that stock unit and office rollover must be carried out as soon as possible.

These rollover checks at logon use the same logic as described in 4.6.5.1, but the age at which warnings start is defined by the 'WarnLogonRolloverDays' attribute of the Type C reference data collection 'CounterConfigParams' (see EA/HLD/005). This defaults to38 days (i.e. the same as 'DisableArchiving' - greater than a normal Trading Period, but less than the default message expiry setting).

4.7.2 Cash Declaration Check

The existing check that an ONCH declaration has been made by the current stock unit on the previous working day will be modified to check that a Cash Declaration has been made on the previous working day. This is further described in section 4.1.7.

The change from ONCH to Cash Declaration checking is performed as soon as the new code is available at the counter at migration point 20.

4.7.3 Trading Period Check

At S75, there is a check during the logon process that the stock unit is in the correct cash account period. A warning is given if this is not the case.

At S80, the logon check needs to be changed such that the user is warned if the stock unit is not in the correct Trading Period. The full details of how this check is performed and how the counter migrates are defined in EA/HLD/005.

4.7.4 Outstanding Transaction Corrections

A new check will be implemented during the logon process that checks for any outstanding Transaction Corrections.

This check will only be performed where the user that is logging on has an appropriate role, determined by explicitly comparing the role of the current user against those defined in the 'Role' attributes of EPOSSStockUnit.TCParams.

The mechanism by which Outstanding Transaction Corrections are identified is described in section 4.1.8.4.5.

If any Outstanding Transaction Corrections are identified then a message will be presented to the user as a prompt. At this point, the user may decide to touch the continue button that will lead to the function described in section 4.5.1 or may choose to touch the Cancel button that will cause the system to continue with the logon process.

N.B. If the counter is disconnected from the LAN (see 4.5.1), or the user is logging on to the default stock unit, then the option to continue will not be offered. See EA/IFS/012 for details.

If the user chooses to continue with the Transaction Corrections processing then, on completion, the system must return to the logon process to complete any outstanding logon activities.

Whenever the Manager or Supervisor is prompted to process Outstanding Transactions at logon, an Event will be written to the message store. (This occurs whether or not they choose (and are able) to process them). This new Event (see 4.11.3.6) will contain a parameter (<P1:> that indicates the number of Outstanding Transaction Corrections at the time of the prompt.

The test for outstanding transaction corrections will be performed only once the stock unit has migrated at point 50.

4.8 Revaluation

A consequence of holding stock by volume rather than by value is that manual revaluation is no longer required as a branch accounting function, and so the user functionality to support revaluation of Stock Levels will no longer be required once Stock-by-Volume is introduced.

However, the underlying revaluation mechanisms are still required to support auto-revaluation of currency during stock unit rollover.

The (Transactions/Reval-Up, Transactions/Reval-Down) menu items will therefore be removed by SoftLaunch upon transition from CAP to TP trading mode at Point 50 in the migration.

The Counter Daily Revaluation Slip will also be no longer required from that point onward (see section 4.9.3).

However, the current mechanism of reporting imminent product revaluations is still required to ensure that stock levels are adjusted correctly prior to the revaluation and that staff are fully aware of the new prices.

Users are warned of revaluations at logon, and the message that appears will be modified. See EA/IFS/012 for the new wording. However, since the old message must be displayed until Stock-by-Volume is in force, the original message definition (MSG232) must be retained, and a new one introduced. The appropriate one will be selected according to the trading mode for the Stock Unit, as defined by the <TP:> attribute of the StockUnit persistent object. For more details, see EA/HLD/005

4.9 Other Reporting

Most of the changes to reporting are to do with the new requirement to handle stock by volume rather than by value, to provide new weekly cut-off reports and to modify the reporting during the Trading Period rollover. The changes for those reports are described in EA/HLD/005.

Transaction Correction reporting and Cash Variance reporting are covered elsewhere within this document (see sections 4.5.9, 4.1.9 respectively).

The other changes to reporting are covered in the following sections.

4.9.1 Remuneration Reporting

The Sales report that exists at S75 will become a Remuneration Report that helps the postmaster to determine the amount of remuneration owing to him however, it will still be known as the Sales Report. A number of changes are required to the report.

The Sales report accepts a single criterion such that it reports all those transactions in the current CAP/TP (RepCriteria = 9). This criterion will remain and will provide a default.

However, a new button will be presented on the Report Dialogue in the same place as a normal Cut-off/Action button. This new button will allow the user to enter a date range across which the transactions will be retrieved and will *replace* the criteria for the current Trading Period (ie: the report may run across a Branch rollover). The date range will be validated such that

- the end-date is greater than or equal to the start date
- the end date is no later than yesterday and EOD has been run for that day.
- the start date is not older than the default message expiry value (as defined in the Riposte Configuration Parameters) less one day. Initially, this will mean that the report may be run up-to 42 days in the past. (See EA/HLD/005 for more information about message retention).
- the EOD marker for the start date – 1 must be present

When a date range has been entered the report is required to select transactions between two Trading dates rather than selecting transactions between physical dates. This means that the report must first determine where the End Of Day markers are for the Start-Date –1 and for the End-Date. Once these are found, then the report can select all transactions between these points.

The report dialogues and format are specified in EA/IFS/011 and the changes will be implemented at migration point 20.

4.9.2 APS Transactions Report

At S75 the APS Report lists all those transactions that have a mapping either directly or indirectly to the accounting node 3026. Since other types of existing product are now being transacted as APS transactions, these products do not necessarily have a primary mapping to the same accounting node.

Therefore, APS transactions should be recognised by the attribute value <Application:APS> rather than by the primary mappings of the transaction.

Since all reports are driven by the primary mappings, then the primary mapping of the APS report needs to be changed to report on the Global Root Node via a change to the GlobalObjects.dat reference data. This will cause the report to pick-up *all* transactions.

The attribute RN requires update to a value of 3017:

```
<Collection:EPOSSReports>
<ObjectName:34>
<Data:
   …
   <RN:3017>
   …
>
```

In order to then limit the selected transactions, a new specific criteria needs to be added to the desktop button that invokes the report:

```
<Collection:DesktopButtonsIItem02657>
<ObjectName:IItem03669>
<RData:<Data:<InterfaceName:<EPOSSReport:
   …
   <SpecificCriteria:
            <Op1:Application>
            <Comp:EQ>
            <Op2:APS>
   >
   …
>>>>
```

The APS Transactions report also supports cut-off, specified by a cut-off id in the desktop button that invokes the report. When the cutoff action button is selected, a ‘tidemark’ (effectively the timestamp of the last transaction within the scope of the report) is written to a Riposte persistent object with that id in the CutOffs collection. Subsequent attempts to cut off the report only succeed if transactions have occurred within the scope of the report since that tidemark (or since the start of the BP if the tidemark is before that).

Since the scope of the APS Report is being changed to root node 3017, a new cut-off id must be introduced (since the previous one corresponds to transactions under 3026), and so the desktop button definition must be modified accordingly.

However, these desktop button changes must not become visible until S80 counter code is activated at migration point 20, so must be controlled by softlaunch. Therefore, instead of modifying the existing button definition in situ, we provide a new variant of the button, and make it hidden by default. The softlaunch definition then hides the old and reveals the new one as required.

4.9.2.1 The MandatorySummaries Problem

Some reports (including APS Transactions) are checked during stock unit rollover to see whether they need to be (perhaps optionally) produced. This decision is based on cutoffs as described above, and is controlled by the Type C reference data collection MandatorySummaries (see section 4.11.6.4).

Each entry in that collection describes a particular report, its cut-off id and root node, and the desktop report button used to produce it. The tidemark for the cut-off is compared with the one for that node in the rollover DataServer tree. If they do not match, the option to produce the report is offered. If the offer is accepted, report production is then launched as if from the specified button.

This MandatorySummaries usage conflicts with the use of softlaunch to switch button definition (and report scope) at migration point 20 as described above, because

(a) The cut-off id and root node in MandatorySummaries do not currently support selection criteria. This means that **any** transaction under 3017 would trigger a tidemark mismatch, resulting in the APS Transactions Report always being offered during rollover.

(b) The cutoff id and root node in MandatorySummaries would not change as a result of the softlaunch, so an obsolete cut-off id would be chosen, resulting in spurious tidemark mismatch.

(c) The button specification in MandatorySummaries would not change as a result of the softlaunch, so the obsolete cut-off id would be specified for the report, resulting in no content.

A variant of this last problem afflicts **any** report controlled by MandatorySummaries, even if the cutoff id and root node remain the same. CP3888 (see EA/HLD/005) introduces report variations at migration Point 30 which are controlled by alternative report parameters supplied by alternative buttons made available via softlaunch. If those reports are chosen during rollover via MandatorySummaries, the wrong report format could be produced.

The solution to these problems is as follows

(1) The softlaunch definitions are extended to define an override for a particular MandatorySummaries object as well hiding/revealing button alternatives. The way to achieve this is by setting a softlaunch SysAttr variable to the replacement grammar string if the relevant softlaunch condition is true. When stock unit rollover processes each object *XX* in MandatorySummaries, it looks for the override SysAttr variable "MandatorySummaries~*XX*". If non-empty, it uses that instead of the grammar defined in object *XX*. The special override value "<Deleted:1>" is used to ignore object *XX* altogether.

(2) MandatorySummaries object syntax (whether read from the object or a softlaunch override) is extended to support a SpecificCriteria attribute in addition to the root node. If this attribute is specified, the stockunit rollover logic ignores the current DataServer tree and instead makes a Riposte query using the specified criteria, starting from the cut-off tidemark (or start of BP). If any transactions are found, the report needs to be produced.

Taking the example of the APS Transactions report, the existing MandatorySummaries object is

```
<Collection:_MandatorySummaries>
<ObjectName:11_04>
<StartDate:01-JAN-1996 00:00:30>
<EndDate:>
<RData:
    <Data:
        <Description:APS Transactions>
        <CutOffID:34>
        <NodeID:3026>
        <Enforce:False>
        <ReportButton:
        <Collection:DesktopButtonsIItem02657>
        <ObjectName:IItem03669>
        >
    >
>
```

The softlaunch definition to hide the old button (3669), reveal the new one (20000208), and override the MandatorySummaries object at migration point 20 would be

```
<Collection:_AppConfig>
<ObjectName:EPOSSStockUnit-P20-01>
<StartDate:01-JAN-1996 00:00:00>
<EndDate:>
<RData:
    <Data:
        <S80_POINT_20:
            <1:
                <IItem01206~IItem02657~IItem03669:
                    <Invisible:1>
                >
                <IItem01206~IItem02657~IItem20000208:
                    <Invisible:0>
                >
                <SysAttr:
                    <MandatorySummaries~11:
                        <Description:APS Transactions>
                        <CutOffID:68>
                        <NodeID:3017>
                        <Enforce:False>
                        <SpecificCriteria:
                            <Op1:Application>
                            <Comp:EQ>
                            <Op2:APS>
                        >
                        <ReportButton:
                            <Collection:DesktopButtonsIItem02657>
                            <ObjectName:IItem20000208>
                        >
                    >
                >
            >
        >
    >
>
```

4.9.3 Reports to be Removed

The following reports are considered no longer necessary as a result of the changes introduced by Impact (or other changes) and so will need to be removed. However, not all are removed at the same time.

Report Name	Button	Report ID	Migration Point
Counter Weekly DVLA V10	Collection:DesktopButtonsIItem02928 ObjectName:Iitem03936	18	20
Counter Weekly DVLA V11	Collection:DesktopButtonsIItem02928 ObjectName:Iitem03937	18	20
Office Weekly Counters Revenue Schedule	Collection:DesktopButtonsIItem02755 ObjectName:Iitem03788	114	50 (see CP 3842)
Office Weekly Counters Revenue Schedule Reprint	Collection:DesktopButtonsIItem02919 ObjectName:Iitem03924	115	50 (see CP 3842)
Declaration and Confirmation – Non-Value Stock	Collection:DesktopButtonsIItem01469 ObjectName:Iitem02141960	LFS Report	50. (see 4.3)
Counter Daily Cash on Hand	Collection:DesktopButtonsIItem02657 ObjectName:Iitem03669	LFS Report	20 (see 4.1.1)
Office Weekly Cash Flow	Collection:DesktopButtonsIItem20000023 ObjectName:Iitem20000027	LFS Report	20 (see 4.1.9)
Counter Daily Revaluation Session Slip	Collection:DesktopButtonsIItem01469 ObjectName:Iitem02141960	LFS Report	See EA/HLD/005

4.10 Riposte Message Formats

This section defines the format of all new Riposte messages and Collections. In addition, changes to existing messages are also described.

4.10.1 Cash Declaration Persistent Object

This existing persistent object records a cash declaration made for a stock unit.

The object name of each cash declaration persistent object is '*tt*_4_1', where *tt* is the till identity (declaration id), and it is part of the collection 'Declaration_*sss*' where *sss* is the name of the stock unit. For individual stock units, there is no till id, so the object name is '_4_1'

The format of these objects is being extended to include an 'adjustment' attribute that records the cumulative effect of Add/Remove Cash actions since the last declaration was made.

```
<Collection:Declaration_sss>
<ObjectName:tt_4_1>
<TxnData:
    <Container:sss>
>
<EPOSSTransaction:
    <LogDel:True> { Only present if the object has been logically deleted – see 4.1.12 }
    <ProductNo:1>
    <MsgType:DrawerItemDeclaration>
    <TotalValue:DeclaredValue>
    <DrawerId:tt>
    <DrawerItem:Cash>
    <DrawerItemId:4>
    <DrawerItemDeclarationId:Id>
    <PG:1>
    <CAP:cc>
    <BP:bp>
    <DrawerItemDeclarationMsgId:MsgId>
    <Mark:Mark>
    <Adjustment:Adjustment>
>
```

Where:

sss	Stock unit identity
tt	Till identity (or Declaration Id). This is omitted for an individual stock unit
DeclaredValue	The total value declared (i.e. the sum of the denominational item declarations). As for other stock holding values in Horizon, this is a –ve value.
Id	A unique identifier recorded against each of the denominational DrawerItemDeclarationDetail messages associated with the declaration.
cc	The CAP/TP in which the declaration was made
bp	The BP in which the declaration was made

MsgId

The Riposte message id of the DrawerItemDeclaration trailer message (similar content to the EPOSSTransaction container in this persistent object, but without this MsgId, Mark or Adjustment).

Mark

The Riposte marker taken when the declaration was made

Adjustment

The adjustment to the declared value. This records the cumulative effect of Add/Remove Cash actions. It is made more –ve by Add Cash, and more +ve by Remove Cash.

All cash declarations at S80 will include this attribute. On initial declaration, the value will be zero.

Absence of the attribute signifies a pre-S80 declaration, which is handled differently. See 4.1.1

N.B. Cash declaration persistent objects and DrawerItemDeclaration messages have very similar syntax. The new **<Adjustment:>** attribute will also be present on DrawerItemDeclaration messages but the attribute value will be zero.

Care should therefore be taken to read the persistent declaration object rather than the original DrawerItemDeclaration message to get the effect of any cash adjustments.

4.10.2 ONCH Declaration Persistent Object

This existing persistent object records an overnight cash holding declaration made for a stock unit. It becomes obsolete as a result of the changes described in this document, but still needs to be interrogated to achieve a smooth transition at S80 migration point 20.

The object name of each object is '*tt_4*', where *tt* is the till identity (declaration id), and it is part of the collection 'ONCH_*ww_sss*' where where *ww* is the week number pertaining to the data being held, and *sss* is the name of the stock unit.

```
<Collection:ONCH_ww_sss>
<ObjectName:tt_4>
<TxnData:
   <Container:sss>
>
<EPOSSTransaction:
   <CAP:cc>
   <BP:bp>
   <BrFwd:BrFwd>
   <Decl_ddd:DeclarationPtr>
      repeated for each ddd value
   <Decl_ddd_Data:
      <Value:DeclarationValue>
      <Date:DeclarationDateTime>
      <User:DeclarationUserId>
   >
      repeated for each ddd value
>
<CarryFwd:CFwd>
```

Where:

ww Current calendar week number in the range 01-53 calculated from the date. N.B this is a VB calendar week (Jan-Dec) and is not related to the POL accounting calendar.

sss Stock unit identity

tt Till identity (or Declaration Id). This is omitted for an individual stock unit

cc The CAP in which the declaration was made

bp The BP in which the declaration was made

BrFwd A "pointer" to the DrawerItemDeclaration trailer message for the latest declaration brought forward from the previous week.

The pointer syntax is "*group_id_num*", corresponding to the Riposte message id

```
<Group:group><Node:id><Num:num>
```

Decl_ddd A "pointer" to the DrawerItemDeclaration trailer message for the declaration made for a particular day

FUJ00091090

of the week. The '*ddd*' value is one of 'Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Tue', or 'Wed'

The pointer syntax is "*group_id_num*", corresponding to the Riposte message id

<Group:*group*><Node:*id*><Num:*num*>

Decl_ddd_Data — Container describing the declaration made for a particular day of the week. The '*ddd*' value is one of 'Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Tue', or 'Wed'

DeclarationDateTime — The date and time that the last cash declaration was brought forward or made for the day indicated by the Decl_*ddd*_Data container.

DeclarationUserId — The ID of the user who performed that declaration.

DeclarationValue — The value of cash declared. As for other stock holding values in Horizon, this is a –ve value.

CarryFwd — A "pointer" to the DrawerItemDeclaration trailer message for the latest declaration. Unless subsequently updated, this will be carried forward to the following week.

The pointer syntax is "*group_id_num*", corresponding to the Riposte message id

<Group:*group*><Node:*id*><Num:*num*>

When creating/updating the ONCH object for a particular day of a particular week, the 'CarryFwd' attribute of that week is also written. In addition, a provisional object is created/updated for the following week, containing values in the 'BrFwd' and 'Decl_Thu_Data' attributes. Thus for a given week, the 'CarryFwd' attribute holds the latest information, or if that is absent, the 'BrFwd' attribute.

(Note that 'BrFwd' is not present if the stock unit has been created during this week, and 'CarryFwd' is not present if the week is still in the future).

4.10.3 Weekly Variance Persistent Objects

There are a number of persistent objects that are created/updated on a daily basis and describe the cash variance position for a given week. They support production of the Cash Variance Report (see 4.1.9), such that it simply needs to read a set of persistent objects rather than having to perform complex processing in real time across the period of one week.

Note: As a result of CP 3980, the Cash Variance Report has been suppressed by removing the buttons that produce or re-print it. However, the underlying logic to support it remains in place and is described in this document for reference.

Maintenance of these objects is split between the in-day logic that actually handles declarations and adjustments affecting variance for a particular stock unit, and end-of-day logic that summarises office-wide statistics.

End-of-day logic also helps to fill in gaps in the report by selectively bringing forward stock unit variance data for those days when no in-day declarations or adjustments were made.

The persistent objects contain a week-number in the name of the collection. The week number relates to a period between Thursday (start of the week) to the next Wednesday (end of the week) and is calculated from the first Thursday in the Calendar year (i.e. the physical Jan-Dec calendar, not the POL Apr-Mar accounting calendar).

Within each object, a set of Decl_*ddd* container attributes describe the position for each day, and a BFwd container holds figures brought forward from the 'Decl_Wed' container in the previous week.

Once the data for a particular week has been completed (by the Wednesday night end-of-day logic, or at the latest, the Thursday morning Retrospective Cash Declaration at Logon), the persistent objects for that week are not updated further, but remain available for reporting until they are pruned after about 6 weeks (see 4.1.13).

Stock unit variance data for a new week is created partly by new in-day actions, but also partly by bring-forward from the previous week.

To avoid obsolete information appearing on the report indefinitely, an entry in a Till Declaration History is qualified by its value. Zero valued cash declarations for shared stock units are deemed to be obsolete (see 4.1.1), and are not propagated forward beyond the current week.

An entry in a Stock Unit Variance History is qualified by its status. Once it is known that the stock unit has traded since the variance was calculated, the status is marked as 'bad', and the report will show 'x' values rather than misleading figures. However, the variance will continue to be propagated forward and reported until such time as the stock unit is deleted. At that time, because of the way that stock unit deletion requires declarations and variances to be resolved beforehand, the reported variance will become zero before it disappears from the report at the end of that week.

If both in-day and end-of-day logic need to create or update the same objects, difficulties can arise if the updates occur at the same time. Although Riposte has a strategy to resolve conflicting updates to a Persistent Object, it chooses the update from the node with the lowest node id. Since the end-of-day logic runs on the gateway node (id=1), end-of-day bring-forward could take precedence over in-day declaration – not what is needed.

To address this problem, we introduce the idea of **background** and **foreground** objects. Whenever the container for a particular day is to be read, the read should first be done on the 'foreground' object. If that object does not exist, or the required container attribute is missing or '*empty*' (i.e. is an empty string rather than having sub-attributes which are themselves empty strings), then it should be read from the 'background' object instead. Whenever a container is to be written, it is always created/updated in either the 'foreground' or 'background' object, according to whether the update is being done in-day or at end-of-day respectively.

Note that this technique is only necessary for those objects that are maintained both by in-day and end-of-day logic. More details are provided in the following sections where that is the case.

4.10.3.1 Till Declaration History

This set of new persistent objects records the history of cash declarations made for each till (i.e. declaration id) within a shared stock unit.

The primary responsibility for creating and updating these objects lies with the in-day logic that supports the Cash Declaration and Add/Remove Cash processes.

Nevertheless, end-of-day logic also updates them in order to bring forward old declarations for those days when no in-day Cash Declaration or Add/Remove Cash occurs.

Therefore two persistent objects ('foreground' and 'background') will be stored for each declaration id within each stock unit for each calendar week, using the foreground/background access strategy outlined in 4.10.3.

The 'foreground' objects are created/updated by the in-day Cash Declaration and Add/Remove Cash processes.

The 'background' objects are created/updated by the EOD 'Bring-Forward Shared SU Till Declarations' activity.

The object names of the persistent objects will be 'FG_*sss_tt*' and '*BG_sss_tt*' respectively, where *sss* is the name of the stock unit and *tt* is the till identity (declaration id), and they will be part of the collection 'Variance_*ww*' where *ww* is the week number pertaining to the data being held.

```
<Collection:Variance_ww>
<ObjectName:FG_sss_tt> {or BG_sss_tt}
<Data:
        <BFwd:              {only present in BG_sss_tt}
            <Value:DeclaredValue>
        >
        <Decl_ddd:
            <Date:DeclarationDateTime>
            <User:DeclarationUserId>
            <Value:DeclaredValue>
            <S:Status>
        >
        repeated for each ddd value
>
```

Where:

ww Current calendar week number in the range 01-53 calculated from the date. N.B this is a VB calendar week (Jan-Dec) and is not related to the POL accounting calendar.

sss Stock unit identity

tt Till identity (or Declaration Id)

BFwd This container holds values brought forward from the previous week. Since it is only ever written by end-of-day logic, it only occurs in the 'background' object.

Decl_*ddd* Container describing the declaration made for a particular day of the week. The 'ddd' value is one of 'Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Tue', or 'Wed'

DeclarationDateTime The date and time that the last cash declaration was brought forward or made for the day indicated by the Decl_ddd container.

It is not updated by Add/Remove Cash

Note that the actual date may be the day after that indicated by the container if the declaration was made as part of logon checks (See section 4.1.7)

DeclarationUserId The ID of the user who brought forward or performed that declaration. This is 'eod' if carried out by EOD code.

DeclaredValue The value of cash declared. As for other stock holding values in Horizon, this is a –ve value

If the user has 'made good' since that time, the value will include those adjustments as well (see 4.1.10.7.2)

Status Possible values are

ok: The declaration has just been made, so is valid.

bfwd: The declaration has been brought forward, but can still be assumed valid unless the corresponding Stock Unit Variance History is invalid.

Note that the 'BFwd' container does not need a status because it is implicitly 'bfwd'.

4.10.3.2 Stock Unit Variance History

This set of new persistent objects records the history of stock unit cash declarations and the difference ('variance') between the declared cash amount and the derived cash value.

The primary responsibility for creating and updating these objects lies with the in-day logic that calculates variance. (Note that Cash Declaration and Add/Remove Cash processes for shared stock units do not affect the calculated variance, because the applicable set of declarations has not been confirmed with the user at that point).

Nevertheless, end-of-day logic also updates them in order to bring forward old declarations for those days when no in-day variance calculation occurs.

Therefore two persistent objects ('foreground' and 'background') will be stored for each stock unit for each calendar week, using the foreground/background access strategy outlined in 4.10.3.

The ‘foreground’ persistent objects are created/updated during

- Cash Declaration and Add/Remove Cash (for individual stock units)
- Check for Variances and Rollover Discrepancy Checking (for shared stock units)
- Cash Auto-Declaration on Inactive Rollover (for either)

The ‘background’ persistent objects are created/updated during

- Bring Forward Stock Unit Variances’ EOD activity.

The object names of the persistent objects will be ‘FG_*sss*’ and ‘BG_*sss*’ respectively, where *sss* is the name of the stock unit, and they will be part of the collection ‘Variance_*ww*’ where *ww* is the week number pertaining to the data being held.

```
<Collection:Variance_ww>
<ObjectName:FG_sss>     {for BG_sss}
<Data:
        <BFwd:              {only present in BG_sss}
           <Value:DeclaredValue>
           <Var:Variance>
           <S:Status>
        >
        <Decl_ddd:
           <Date:DeclarationDateTime>
           <User:DeclarationUserId>
           <Value:DeclaredValue>
           <Var:Variance>
           <Mark:Mark>
           <S:Status>
        >
        repeated for each ddd value
>
```

Where:

ww

Current calendar week number in the range 01-53 calculated from the date. N.B this is a VB calendar week (Jan-Dec) and is not related to the POL accounting calendar.

sss

Stock unit identity

BFwd

This container holds values brought forward from the previous week. Since it is only ever written by end-of-day logic, it only occurs in the ‘background’ object.

Decl_*ddd*

This container holds values for a particular day of the current week. The ‘*ddd*’ value is one of ‘Thu’, ‘Fri’, ‘Sat’, ‘Sun’, ‘Mon’, ‘Tue’, or ‘Wed’.

DeclarationDateTime

The date and time that the declaration was brought forward, or determined during Cash Declaration (for individual stock units), Check for Variances/Rollover Discrepancy Checking (for shared stock units), or Cash Auto-Declaration on Inactive Rollover (for either).

It is not updated by Add/Remove Cash except as a consequence of bring-forward logic. See 4.1.10.7.1.

Note that this may not be the same day as that of the container if the declaration was made retrospectively as part of logon checks (See section 4.1.7)

DeclarationUserId

The ID of the user that brought forward the declaration, or performed the Cash Declaration, Check for Variances/Rollover Discrepancy Checking, or Cash Auto-Declaration on Inactive Rollover functions.

This is ‘eod’ if the bring forward is carried out by EOD code.

It is not updated by Add/Remove Cash except as a consequence of bring-forward logic. See 4.1.10.7.1.

DeclaredValue

The value of declared cash brought forward, or determined during Cash Declaration (for individual stock units), Check for Variances/Rollover Discrepancy Checking (for shared stock units), or Cash Auto-Declaration on Inactive Rollover (for either).

As for other stock holding values in Horizon, this is a –ve value.

For shared stock units, the total declared cash value is the sum of the declared cash values for each declaration id

If Add/Remove Cash has been done since that time, the value will include those adjustments as well (see 4.1.10.7)

Variance

The uncommitted cash discrepancy brought forward or calculated.

If calculated, this is the difference between the total derived cash value and the declared cash value as determined during Cash Declaration (for individual stock units), or Check for Variances/Rollover Discrepancy Checking (for shared stock units).

In other words, a shortfall is represented as a –ve value.

For Cash Auto-Declaration on Inactive Rollover, the variance is 0.

If Add/Remove Cash has been done since that time, the variance will include those adjustments as well (see 4.1.10.7)

Mark

This is a Riposte marker taken when the variance was calculated – i.e. during Cash Declaration (for individual stock units), Check for Variances/Rollover Discrepancy Checking (for shared stock units), or Cash Auto-Declaration on Inactive Rollover (for either).

It is used as the start-point for a message store scan to determine if the stock unit has traded since then.

It is not updated by Add/Remove Cash except as a consequence of bring-forward logic. See 4.1.10.7.1.

Note: To avoid problems with Riposte message size limits, the mark is only recorded for the latest two entries in the week. See 4.10.3.2.2.

Status

Possible values are

ok: The variance has just been calculated via Cash Declaration (for individual stock units), Check for Variances/Rollover Discrepancy Checking (for shared stock units), or Cash Auto-Declaration on Inactive Rollover (for either), so the figures will be ok for today, and to carry forward into tomorrow if no further trading takes place

bfwd: The figures have been brought forward. No trading has happened since the last calculation, so this figure will be ok for today, and to carry forward into tomorrow if no further trading takes place

bad: The figures are out of date because of subsequent trading, and cannot be used today.

It is not updated by Add/Remove Cash except as a consequence of bring-forward logic. See 4.1.10.7.1

N.B. Care must be taken when reading this value from foreground/background container. See 4.10.3.2.1 below

4.10.3.2.1 Reading Foreground/Background Status

The normal strategy (as described in 4.10.3) is to read attribute values from the 'foreground' data container in preference to the 'background' data container. This works well except for the case where figures have been brought forward in-day by Add/Remove Cash (and thereby written to the foreground container), but subsequently downgraded to status 'bad' at end-of-day by Maintain Office Variances (and thereby written to the background container).

If Cash Variance Report were to simply read the foreground data container, it could wrongly regard the figures as valid. The strategy must therefore be slightly modified :-

- If foreground data container exists, use its values in preference to the background container.
- However, if both containers exist, and the foreground container status is 'bfwd', use the background status in preference (since it may imply a downgrade to 'bad').

Other than the status, values should still be taken as normal from the foreground container, in order to reflect the effect of any subsequent Add/Remove Cash adjustments.

Note that this subtlety need only affect Cash Variance Report. The bring forward logic in Maintain Office Variances or Add/Remove Cash does trading checks anyway if the status is 'bfwd'. The correct effect is therefore achieved regardless of whether status is read from foreground or background.

(Earlier versions of this design used a separate status value 'obsolete' to optimize out the trading check in some other circumstances, but that is incompatible with the modified strategy described above, and has been withdrawn).

4.10.3.2.2 Pruning Stock Unit Variance History Markers

Ideally, we would simply record the stock unit variance history marker for each day as required, and keep these marks for the entire week. However, for branches with a large number of counters, the size of a mark can be hundreds of characters, so there is an issue about containing the total size of the message to the Riposte limit of 2048 characters.

In fact, we only need to keep two marks (for 'today' and 'yesterday') to support the bring-forward logic described in 4.1.8.2 and 4.1.10.7.1.

Therefore, any logic that updates a Stock Unit Variance History should remove the <Mark:> attribute for the Decl_*ddd* containers for the day before yesterday (and earlier) before writing a mark into the Decl_*ddd* container for 'today'. (There is of course no need to prune markers from previous weeks when doing this).

4.10.3.3 Stock Unit Discrepancy History

This set of new persistent objects records the history of committed discrepancies (not just cash) for each stock unit, and one persistent object will be stored for each stock unit for each calendar week.

The responsibility for creating and updating these objects lies with the End Of Day 'Calculate Stock Unit Committed Discrepancies' activity (4.1.8.3), so there is no need for 'foreground' and 'background' objects.

The object names of the persistent objects will be 'Discrepancy_*sss*', where *sss* is the name of the stock unit, and they will be part of the collection 'Variance_*ww*' where *ww* is the week number pertaining to the data being held.

```
<Collection:Variance_ww>
<ObjectName: Discrepancy_sss>
<Data:
        <BFwd:
           <Dis:Disc>
        >
        <Decl_ddd:
           <Dis:Disc>
        >
        repeated for each ddd value
>
```

Where:

ww Current calendar week number in the range 01-53 calculated from the date. N.B this is a VB calendar week (Jan-Dec) and is not related to the POL accounting calendar.

sss Stock unit identity

BFwd This container holds values brought forward from the previous week.

Decl_*ddd* This container holds discrepancy values for a particular day of the current week. The '*ddd*' value is one of 'Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Tue', or 'Wed'.

Disc The committed discrepancy (not just cash). A shortfall is held as a –ve value

4.10.3.4 Office Variance History

This set of new persistent objects records the overall movement of suspense items, adjustments and Transaction Corrections.

One persistent object will be stored for each branch for each calendar week.

The responsibility for creating and updating these objects lies with the End Of Day 'Calculate Office Variance activity (see 4.1.8.4), so there is no need for 'foreground' and 'background' objects.

The object name of the persistent object will be 'Office', and it will be part of the collection called 'Variance_*ww*' where *ww* is the week number pertaining to the data being held.

```
<Collection:Variance_ww>
<ObjectName:Office>
<Data:
        <BFwd:
           <Susp:SuspenseAccValue>
           <Local:LocSuspenseValue>
           <Adj:AdjustmentValue>
           <TCP:TxnCorrProcessed>
           <TCO:TxnCorrOutstanding>
        >
        <Decl_ddd:
           <Susp:SuspenseAccValue>
           <Local:LocSuspenseValue>
           <Adj:AdjustmentValue>
           <TCP:TxnCorrProcessed>
           <TCO:TxnCorrOutstanding>
        >
        repeated for each ddd value
>
```

Where:

ww — Current calendar week number in the range 01-53 calculated from todays date. N.B this is a VB calendar week (Jan-Dec) and is not related to the POL accounting calendar.

Office — The text 'Office'

BFwd — This container holds values brought forward from the previous week.

Decl_*ddd* — Day of the week. This will be one of 'Thu', 'Fri', 'Sat', 'Sun', 'Mon', 'Tue', or 'Wed'. This container is created during the EOD Process.

SuspenseAccValue — The total value outstanding in all suspense accounts (other than local suspense) at the end of the current Trading Day.

It summarises the total suspense value at the start of the current CAP/TP (i.e. the one in force at the end of the current Trading Day), plus the effect of any movements since then.

A nett loss is held as a –ve value

LocSuspenseValue

The total value outstanding in the local suspense account at the end of the current Trading Day.

It summarises the effect of any movements since the start of the current CAP/TP (i.e. the one in force at the end of the current Trading Day).

A nett loss is held as a –ve value

AdjustmentValue

The total movement of value that has been transacted via ‘Add/Remove Cash’ from all Stock Units in the current Trading Day.

A nett addition of cash is represented by a –ve value

TxnCorrProcessed

The total number of Transaction Corrections processed in the current Trading Day.

TxnCorrOutstanding

The total number of Transaction Corrections that are Outstanding at the end of the current Trading Day.

4.10.4 Transaction Correction Messages

4.10.4.1 Transaction Correction Message

The transaction correction messages are generated by the TPS Transaction Corrections Loader (see EA/HLD/009, EA/HLD/010). These messages are Horizon interpretations of the data presented across the Horizon contractual boundary as described in the Transaction Corrections AIS (EA/IFS/002).

The Transaction Correction messages will use the <WAIndex.LFSFlag:> indexed attribute to identify the messages and to support efficient message identification and retrieval.

```
<Application:TC>
<WAIndex:
        <LFSFlag:TC>
>
<Data:
        <TranType:TransactionCorrection>
        <Ref:ReferenceId>
        <Iter:IterationFlag>
        <Article:Product>
        <Instruction:SettlementProduct>
        <AccountingSense:AccountingSense>
        <Value:ErrorValue>
        <Qty:ErrorQuantity>
        <AllowedModesId:AllowedModes>
        <Modes:
           <1:ModeOption1>
           <2:ModeOption2>                              {May be omitted}
           <3:ModeOption3>                              {May be omitted}
        >
        <Text:MessageText>
        <ClientRef:ClientReference>
>
```

Where:

ReferenceId The transaction correction reference number. This is unique when in combination with the *IterationFlag*.

IterationFlag Contains a value of 'N' (New) or 'E' (Evidence Provided).

Product This may contain the Horizon Product Id of the product to be transacted as a result of processing the Transaction Correction. However, the actual product transacted may be different because of the selected mode. See 4.5.4 for details

SettlementProduct Depending on the Modes available to the PostMaster, this may indicate the product to which the Transaction Correction can be settled. See 4.5.4 for details

AccountingSense This contains the value 'TCINV' (invoice) if the article product is sold or 'TCCRM' (credit memo) if it is returned.

The effect of a 'TCINV' is to decrease the branch holding of the affected product. Conversely, a 'TCCRM' always increases the holding. However, since holdings are represented in the counter as –ve quanities and values, a 'TCINV' transaction will result in a transaction for the product with +ve (or zero) quantity and value. See 4.5.4 for details

ErrorValue The Value to be transacted as part of the Correction. If Value is non-zero, then Quantity will be zero.

ErrorQuantity The Quantity to be transacted as part of the Correction. If Quantity is non-zero, then Value is irrelevant (not to be used).

AllowedModes The Mode Id that is passed from the POL FS System and from which the ModeOptions are derived within the TPS System. (This is not used in the counter, and is provided purely for diagnostic purposes)

ModeOption1/2/3 The possible modes in which the Correction may be transacted.

MessageText Up to 500 characters of message text. Carriage returns are depicted by the presence of the vertical bar character '|'.

ClientReference A client reference number.

The Transaction Correction can be uniquely identified by concatenating *ReferenceId* and *IterationFlag*.

4.10.4.2 Processed Transaction Correction Message

Once transaction corrections have been processed, a message is written to indicate that they are no longer outstanding. The 'processed' message will record the details about the outcome of the transaction for use within the Processed Transaction Corrections report. Many of the Transaction Correction attributes are copied onto this message to make reporting easier.

The Processed Transaction Correction messages will use the <WAIndex.LFSFlag:> indexed attribute to identify the messages and to support efficient message identification and retrieval.

```
<Application:TC>
<WAIndex:
        <LFSFlag:TP>
>
<Data:
        <TranType:CompletedTC>
        <Ref:ReferenceId>
        <Iter:IterationFlag>
        <RcvdDate:DateReceived>
        <Product:Product>
        <Settlement:SettlementProduct>
        <AccountingSense:AccountingSense>
        <Value:ErrorValue>
        <Qty:ErrorQuantity>
        <Outcome:TxnMode>
        <Text:MessageText>
        <ClientRef:ClientReference>
>
```

Where:

ReferenceId The transaction correction reference number. This is unique when in combination with the *IterationFlag*.

IterationFlag Contains a value of 'N' (New) or 'E' (Evidence Provided).

DateReceived The date when the Transaction Correction was loaded into the message store from TPS by the agent loader. This is the date on the associated Transaction Correction Message.

Product This is the Horizon Product Id of the product that was actually transacted as a result of processing the Transaction Correction. It may or may not match the product specified in the original Transaction Correction. See section 4.5.4

SettlementProduct This indicates the product to which the Transaction Correction was settled, and is governed by the Mode that is selected during Transaction Correction Processing. See section 4.5.4

AccountingSense Containing the value 'TCINV' or 'TCCRM' as in the original Transaction Correction.

ErrorValue The Value transacted as part of the Correction. This may or may not match the value specified in the original Transaction Correction. See section 4.5.4

ErrorQuantity The Quantity transacted as part of the Correction. This may or may not match the quantity specified in the original Transaction Correction. See section 4.5.4

TxnMode The Mode Id the Transaction Correction was transacted. This will be one of the alternatives specified in the original Transaction Correction

MessageText Up to 500 characters of message text, as in the original Transaction Correction.

ClientReference The client reference number, as in the original Transaction Correction

The Processed Transaction Correction can be uniquely identified by concatenating *ReferenceId* and *IterationFlag*.

4.11 Reference Data Requirements

This section defines the new items of reference data that need to be delivered in order to support the functionality in the rest of this document. In some cases, the actual reference data items have been specifically defined. In other instances, the reference data has been identified at a high-level and the detail of the reference data values will be driven-out at low-level design stage.

4.11.1 New Transaction Modes

Six new modes are required to support the Transaction Corrections functionality. The reference data Collection 'ModeParameters' defines how the modes are used within the context of the Counter desktop. A new instance of the collection is required for each new Transaction Mode.

An additional EVNT mode needs to be introduced that corresponds to the existing POL Mode 20 ('event mode'). This mode will be used for local transactions created during Add/Remove Cash to support local balancing. These local transactions are in addition to any event messages, and must be ignored outside the counter. See 4.1.10.5

The above changes will be made prior to migration point 20 since they are benign within the counter software in the absence of S80 executables.

4.11.1.1 Mode EVNT (Mode 20)

No settlement product is required because the mode is used for local 'event' transactions that do not need to balance. See 4.1.10.5.

```
<Collection:ModeParameters>
<ObjectName:EVNT>
<ModeInfo:
        <Cmd:ChangeMode>
        <DASS:True>
        <MaxStackTotal:9999999.99>
        <Mode:EVNT>
        <MC:True>
        <PostSettleTxn:True>
        <ShowNoRed:True>
        <ModeTitle:Cash Adjustment Event
        <ReverseSense:False>
        <PrimaryMappings:>
        <SecondaryMappings:>
        <VolsValue:Sale>
>
```

4.11.1.2 Mode MG (Mode 29)

The settlement product will be determined dynamically (see 4.5.4).

```
<Collection:ModeParameters>
<ObjectName:MG>
<ModeInfo:
        <Cmd:ChangeMode>
        <DASS:True>
        <MaxStackTotal:9999999.99>
        <Mode:MG>
        <MC:True>
        <PostSettleTxn:True>
        <ShowNoRed:True>
        <ModeTitle:Accept Now>
        <ReverseSense:False>
        <PrimaryMappings:>
        <SecondaryMappings:>
        <VolsValue:Loss>
        <DR:True> { see 4.5.7 }
    >
```

4.11.1.3 Mode HD (Mode 30)

Transactions will automatically settle to the 'Hardship' product that is defined in this reference data (Product 6291).

```
    <Collection:ModeParameters>
    <ObjectName:HD>
    <ModeInfo:
        <Iitem:>
        <Cmd:ChangeMode>
        <DASS:True>
        <MaxStackTotal:9999999.99>
        <Mode:HD>
        <MC:True>
        <SettlementProduct:6291>
        <ModeTitle:Settle Centrally>
        <ReverseSense:False>
        <PrimaryMappings:>
        <SecondaryMappings:>
        <VolsValue:Loss>
        <DR:True> { see 4.5.7 }
    >
```

4.11.1.4 Mode EV (Mode 33)

The 'request evidence' option should create no net movement of product yet still generate transaction(s) so that POL FS understands that the Postmaster requires further proof/evidence.

The fixed 'Evidence Required' product will be transacted, and the transaction will be automatically settled to the same product, resulting in no actual net movement of value or volume.

```
<Collection:ModeParameters>

    <ObjectName:EV>
    <ModeInfo:
        <Iitem:>
        <Cmd:ChangeMode>
        <DASS:True>
        <MaxStackTotal:9999999.99>
        <Mode:EV>
        <MC:True>
        <SettlementProduct:6294>
        <ModeTitle:Seek Evidence>
        <ReverseSense:False>
        <PrimaryMappings:>
        <SecondaryMappings:>
        <VolsValue:Zero>
        <DR:True> { see 4.5.7 }
    >
```

4.11.1.5 Mode WO (Mode 31)

Transactions will automatically settle to the 'Write Off' product that is defined in this reference data (Product 6292).

```
<Collection:ModeParameters>
<ObjectName:WO>
<ModeInfo:
        <Iitem:>
        <Cmd:ChangeMode>
        <DASS:True>
        <MaxStackTotal:9999999.99>
        <Mode:WO>
        <MC:True>
        <SettlementProduct:6292>
        <ModeTitle:Send to P&L>
        <ReverseSense:False>
        <PrimaryMappings:>
        <SecondaryMappings:>
        <VolsValue:Loss>
        <DR:True> { see 4.5.7 }
>
```

4.11.1.6 Mode AN (Mode 32)

Transactions will automatically settle to the 'Assign Nominee' product that is defined in this reference data (Product 6293).

```
<Collection:ModeParameters>
<ObjectName:AN>
<ModeInfo:
        <Iitem:>
        <Cmd:ChangeMode>
        <DASS:True>
        <MaxStackTotal:9999999.99>
        <Mode:AN>
        <MC:True>
        <SettlementProduct:6293>
        <ModeTitle:Assign to HO>
        <ReverseSense:False>
        <PrimaryMappings:>
        <SecondaryMappings:>
        <VolsValue:Loss>
        <DR:True> { see 4.5.7 }
>
```

4.11.1.7 Mode SW (Mode 34)

Despite the name 'Stock Write Off', this option is intended to support correction of Non-Accounting Data (as originally transacted via '10g' transactions), and these products have volume but no value. However, it could also be used to adjust volume stock levels, in which case the product value should be ignored. Either way, the transaction will have value zero and require no settlement transaction.

```
<Collection:ModeParameters>
<ObjectName:SW>
<ModeInfo:
        <Cmd:ChangeMode>
```

FUJ00091090

```
          <DASS:True>
          <MaxStackTotal:9999999.99>
          <Mode:SW>
          <MC:True>
          <PostSettleTxn:True>
          <ShowNoRed:True>
          <ModeTitle:Stock WO>
          <ReverseSense:False>
          <PrimaryMappings:>
          <SecondaryMappings:>
          <VolsValue:Zero>
          <DR:True> { see 4.5.7 }
  >
```

4.11.2 Counter Scheduler Changes

The changes to the counter scheduler reference data are described in the body of this document. Multiple releases of counter scheduler reference data are required to phase the implementation of non-desktop functionality during the migration.

4.11.2.1 Changes before Migration Point 20

Update the counter scheduler reference data to remove invocation of the LFS EOD Weekly stock statement function. This involves removing/disabling the following two persistent objects

Collection: SchedulerTimeEvents

Object: LFSStockStatement1
LFSStockStatement2

4.11.2.2 Changes at Migration Point 20

The counter schedule needs to be updated to invoke the new Maintain Office Variances module (see section 4.6.3) in the gateway node.

Twonew objects therefore need to be defined in SchedulerTimeEvents and SchedulerBusinessEvents.

```
<Collection:SchedulerBusinessEvents>
<ObjectName:OfficeVariances>
<Data:
    <Trigger:
        <Type:EndOfDay>
    >
    <Notify:
        <CmdLine:c:\Counters\bin\CASEPOSSImpact.exe>
        <CmdParams:maintainvariances>
    >
    <Depend:
        <After:
            <Name:EODMarkers>
            <SuccessRequired:0>
        >
    >
    <Exceptions:
        <Timeout:600>
        <RestartPercent:90>
    >
    <Counters:Gateway>
>
```

```
<Collection:SchedulerTimeEvents>
<ObjectName:OfficeVariances>
<Data:
    <Trigger:
        <Day:
            <DayName:Every>
            <StartTime:03:40:00>
            <EndTime:19:00:00>
        >
    >
    <Notify:
        <CmdLine:c:\Counters\bin\CASEPOSSImpact.exe>
        <CmdParams:maintainvariances /catchup>
    >
    <Exceptions:
        <Timeout:600>
        <RestartPercent:90>
    >
    <Counters:Gateway>
>
```

4.11.2.3 Changes between Migration Point 40 and 50

Update the counter scheduler reference data to remove the call to the EOD Daily Reconciliation module and to replace it with a call to the new EOD Daily Reconciliation (See section 4.6.4.1). This involves updating the following persistent objects:

Collection: SchedulerTimeEvents
Object: EPOSSDailyRecon

and

Collection: SchedulerBusinessEvents
Object: EPOSSDailyRecon

Update the Command Line (excluding parameters) to read:

```
    <CmdLine:c:\Counters\bin\CASEPOSSDailyRecon2.exe>
```

Update the counter scheduler reference data to remove invocation of the EPOSS Weekly Reconciliation function. This involves removing/disabling the following two persistent objects

```
    Collection: SchedulerTimeEvents
    Object: EPOSSWeeklyRecon
and
    Collection: SchedulerBusinessEvents
    Object: EPOSSWeeklyRecon
```

4.11.3 EPOSS Event Definitions

All of the new events that are introduced within this document need to be defined within reference data.

This Type-C reference data determines the Primary Mapping of the events and this in-turn dictates which reports the events are included within.

The primary mapping for all of the new events is therefore as follows:

Event Product	Event ID	Event Title	Primary Mapping		
			L2	L3	L4
6302	58	Excess Cash Removed	3109 (Stock Unit)	3102 Balancing	3108 Events
6303	59	Cash Shortage Made Good			
6304	60	Cash Variance Report Previewed	3193 (Report Production)	3106 (Reports)	
6305	61	Cash Variance Report Printed			
6306	62	Outstanding Transaction Correction Reminder Displayed	3192 (Report Confirmation)		
6307	63	Shared Stock Unit Variance Check Complete	3109 (Stock Unit)	3102 (Balancing)	
6308	64	Shared Stock Unit Variance Check Complete With Discrepancy			
6299	65	Trading Statement Created	3103 (Office)		
6300	66	Trading Statement Period Rolled			
6301	67	Trading Statement Period Roll Abandoned			

Note that the existing events 40 (Cash Acc Created), 41 (Office CAP rolled), and 42 (Office CAP Roll Abandoned) are replaced by events 65, 66, 67 respectively on reaching migration point 50.

New events are required for these on entering TP mode (rather than simply changing the text on existing events and messages as described in EA/HLD/005), because they are to be harvested by TPS and sent to POL MIS. (The TPS harvesting changes occur at migration point 10 (see EA/HLD/008), so reusing the existing events would cause them to be unexpectedly sent to the old TIP system from pre-point 50 branches).

To support the production of local transactions for cash adjustment events, as described in 4.1.10.5, the Type C data defining these events will be extended with a <PN:> attribute to identify the corresponding Horizon event product. (Other events will not be similarly extended because there is no need).

In relevant <T:> message texts, there is a new infill '%TILL%'. For individual stock units this will expand to an empty string. For shared stock units, it will expand to 'Till %DecId%', where %DecId% is the till declaration id.

In addition to the above new events, the existing till-level Declaration event message (ID=21) will be enhanced to include the declaration id (if any) in the text as described above, and parameterise it with the value and declaration id (if any). This should aid problem diagnosis, and in particular it will address PEAK incident PC0091605

4.11.3.1 Event 21 (Declaration Complete)

```
<Data:
    <ID:21>
    <C:EVNT_DECSUCCESS>
    <Ca:Accounting>
    <T: %DECTYPE% Total £%Tot% For SU %SU% %TILL%>
    <Ti:Declaration Complete>
    <PM:
        <L1:>
        <L2:3109>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <P:
        <P1:%Tot%>
        <P2:%DecId%>
    >
>
```

4.11.3.2 Event 58 (Excess Cash Removed)

```
<Data:
    <ID:58>
    <C:EVNT_EXCESSCASH>
    <Ca:Accounting>
    <T:£%Value% Excess cash removed from SU %SU% %TILL%>
    <Ti:Excess Cash Removed>
    <PM:
        <L1:>
        <L2:3109>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <PN:6302>  { see 4.1.10.5 }
    <P:
        <P1:%Value%>
        <P2:%DecId%>
    >
>
```

4.11.3.3 Event 59 (Cash Made Good)

```
<Data:
    <ID:59>
    <C:EVNT_CASHSHORTAGE>
    <Ca:Accounting>
    <T:£%Value% Cash made good into SU %SU% %TILL%>
    <Ti:Cash Made Good>
    <PM:
        <L1:>
        <L2:3109>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <PN:6303>  { see 4.1.10.5 }
    <P:
        <P1:%Value%>
        <P2:%DecId%>
    >
>
```

4.11.3.4 Event 60 (Cash Variance Report Previewed)

```
<Data:
    <ID:60>
    <C:EVNT_VARIANCE_PREVIEW>
    <Ca:Reports>
    <T:Cash Variance report previewed>
    <Ti:Cash Variance Report Previewed>
    <PM:
        <L1:>
        <L2:3193>
        <L3:3106>
        <L4:3108>
    >
    <DEP:E>
    <P:>
>
```

4.11.3.5 Event 61 (Cash Variance Report Printed)

```
<Data:
    <ID:61>
    <C:EVNT_VARIANCE_PRINT>
    <Ca:Reports>
    <T:Cash Variance report printed>
    <Ti:Cash Variance Report Printed>
    <PM:
        <L1:>
        <L2:3193>
        <L3:3106>
        <L4:3108>
    >
    <DEP:E>
    <P:>
>
```

4.11.3.6 Event 62 (Outstanding TC Prompted)

```
<Data:
    <ID:62>
    <C:EVNT_OUTSTANDING_TC>
    <Ca:Accounting>
    <T:%Count% Outstanding TC(s) prompted>
    <Ti:Outstanding TC Prompted>
    <PM:
        <L1:>
        <L2:3192>
        <L3:3106>
        <L4:3108>
    >
    <DEP:E>
    <P:
        <P1:%Count%>
    >
>
```

4.11.3.7 Event 63 (Variance Check)

```
<Data:
    <ID:63>
    <C:EVNT_NO_VARIANCE>
    <Ca:Accounting>
    <T:Variance Check for SU %SU%>
    <Ti:Variance Check>
    <PM:
        <L1:>
        <L2:3109>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <P:>
>
```

4.11.3.8 Event 64 (Variance Check Discrepancy)

```
<Data:
    <ID:64>
    <C:EVNT_VARIANCE>
    <Ca:Accounting>
    <T:Variance Check for SU %SU% with £%Dis% Discrepancy>
    <Ti:Variance Check Discrepancy>
    <PM:
        <L1:>
        <L2:3109>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <P:
        <P1:%Dis%>
    >
>
```

4.11.3.9 Event 65 (Trading Statement Created)

```
<Data:
    <ID:65>
    <C:EVNT_TRADINGSTATEMENT>
    <Ca:Accounting>
    <T:Trading Statement created for TP %CAP%>
    <Ti:Trading Statement Created>
    <PM:
        <L1:>
        <L2:3103>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <P:
        <P1:%CAP%>
    >
>
```

4.11.3.10 Event 66 (Branch TP Rolled)

```
<Data:
    <ID:66>
    <C:EVNT_BRANCH_ROLLOVER>
    <Ca:Accounting>
    <T:Office rolled from TP %OldCAP% to TP %CAP%>
    <Ti:Branch TP Rolled>
    <PM:
        <L1:>
        <L2:3103>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <P:
        <P1:%CAP%>
    >
>
```

4.11.3.11 Event 67 (Branch TP Roll Abandoned)

```
<Data:
    <ID:67>
    <C: EVNT_BRANCH_ROLL_ABNDN>
    <Ca:Accounting>
    <T:TP %CAP% rollover abandoned>
    <Ti:Branch TP Roll Abandoned>
    <PM:
        <L1:>
        <L2:3103>
        <L3:3102>
        <L4:3108>
    >
    <DEP:E>
    <P:
        <P1:%CAP%>
    >
>
```

4.11.4 Global Objects

The GlobalObjects.dat file contains the report layout definitions. Since these will be changing at S80, then the file will need to be re-distributed with the counter software changes. The details of each change to the Global Objects file are dictated by the report layouts as described in the User Interface Design Proposals and these details are not discussed in this document.

4.11.5 Desktop Buttons

Most changes to the Desktop Buttons are described in the Menu Hierarchy document SD/SPE/016. However, there are additional changes to the content of desktop buttons that do not visually affect the desktop interface.

4.11.5.1 APS Transactions Report Button

The APS Report criteria needs to change to ensure that all APS transactions are included in the selection criteria. This change is documented in section 4.9.2.

4.11.5.2 Transaction Log Report Button

The Transaction Log Report criteria needs to change to ensure that Cash adjustment pseudo-transactions (EPOSSTransaction.TranType=L) are excluded (see section 4.1.10.5).

Since this change has no effect on pre-S80 systems, it can be provided as a simple amendment to the existing button rather than introduce a whole new one via softlaunch.

```
<Collection:DesktopButtonsIItem01206>
<ObjectName:IItem02683>
<RData:<Data:<InterfaceName:<EPOSSReport:
    ...
    <SpecificCriteria:
            <Op1:EPOSSTransaction.OpeningFiguresId>
            <Comp:NOT EXISTS>
    >
    <SpecificCriteria:
            <Op1:EPOSSTransaction.TranType>
            <Comp:EQ>
            <Op2:S>
    >
    ...>>...>>
```

4.11.5.3 Sales Report Button

The sales report includes a new action button as described in section 4.9.1.

Since this new action button should only appear at migration point 20 when S80 counter code is activated, the revised desktop button needs to be defined in a new report menu button revealed by softlaunch :

```
<Collection:DesktopButtonsIItem02755>
<ObjectName:IItemnnnnn>
<RData:<Data:<InterfaceName:<EPOSSReport:
    ...
    <ShowActionButton:True>
    <ActionButtonPicFile:>
    <ActionButtonImageIndex:>
    <ActionButtonHelpText:Touch this button define a date range>
    <ActionButtonCaption:Date Range>
    <ActionButtonInterface:
    <EPOSSReport:
            <Cmd:EnterDateRange>
    >
    ...>>...>>
```

4.11.6 Other Collections

4.11.6.1 WatchDogIntUpdates

WatchDogIntUpdates is an existing Riposte collection containing temporal Type C Reference data objects, and is used by EPOSSWatchdog to determine the actions to be taken on detecting that nodes have become disconnected or reconnected.

Existing definitions ensure that the 'Stock Balancing' and 'Reports' buttons on the root menu are disabled if the node is disconnected, andthis is enough to prevent access to buttons on those menus such as 'Declare Cash' – even if the user has entered the menu before the disconnection occurred.

No such general restriction applies to the Housekeeping menu, so a new definition must be added in order to disable the new Transaction Correction Processing button on the Housekeeping menu if the node becomes disconnected.

4.11.6.2 EPOSSTxnSelection

EPOSSTxnSelection is an existing Riposte collection containing Type C reference data that controls the end-of-day EPOSSReconciliation task.

4.11.6.2.1 EPOSSTxnSelection.Criteria01

The Criteria01 object defines Riposte selection criteria for harvesting relevant messages for analysis by EPOSSReconciliation. This criteria will need to be extended to ignore 'TranType=L' local transaction messages as described in 4.1.10.5

The revised criteria will therefore be

```
NOT (
        (TxnData.Container EQ "") OR
        (EPOSSTransaction.PM.L5 NE "3017") OR
        (Exists (EPOSSTransaction.SM.L4) AND EPOSSTransaction.SM.L4 EQ "3021") OR
        (Exists (EPOSSTransaction.TranType) AND EPOSSTransaction.TranType EQ "L") OR
        (Exists(EPOSSTransaction.OpeningFiguresId))
)
```

4.11.6.3 EPOSSStockUnit

EPOSSStockUnit is an existing Riposte collection containing Type C Reference data objects such as EPOSSStockUnit.Parameters. It will be extended as follows

4.11.6.3.1 EPOSSStockUnit.TCParams

To support the Transaction Correction Processing logic, we introduce a new temporal reference data configuration object 'EPOSSStockUnit.TCParams'

```
<Collection:EPOSSStockUnit>
<ObjectName:TCParams>
<StartDate:01-JAN-2004 00:00:00>
<EndDate:>
<RData:
          <Data:
             <Roles:
                <Role:role>
                repeated for each applicable role
             >
```

```
                <MakeGoodOptions:
                    <Option:option>
                >
                <SecondaryMapping:secondarymapping>
            >
>
```

where

role

The repeated values of this attribute define the Riposte roles that are allowed to access Transaction Correction Processing.

This is currently defined to be either 'Manager' or 'Supervisor'

option

Each option defines a method of settling a 'Make Good' transaction correction from a cash-equivalent.

For S80, there will be just two options – Cash and Cheque. See 4.5.3

In future, there may be other options such as Debit Card

The purpose of the option grammar string is to define the displayable text to be presented in the picklist entry, plus the interface definition by which the chosen option can be transacted onto the EPOSSCore stack (as if launched from a desktop button).

secondarymapping

To support production of the Branch Trading Statement (see also EA/HLD/005), Transaction Correction transactions must contain a secondary mapping that allows them to be counted under a separate part of the balancing tree as part of stock unit rollover. This avoids rollover having to do a separate scan for corrections. See 4.5.4 for details.

The mapping will be of the form

```
<L1:><L2:><L3:981><L4:980><L5:3017>
```

4.11.6.3.2 EPOSSStockUnit.OfficeVariances

To support the Maintain Office Variances logic, we introduce a new temporal reference data configuration object 'EPOSSStockUnit.OfficeVariances'

```
<Collection:EPOSSStockUnit>
<ObjectName:OfficeVariances>
<StartDate:01-JAN-2004 00:00:00>
<EndDate:>
<RData:
    <Data:
        <Queries:
            <Traded:Traded>
            <Discrepancies:Discrepancies>
```

```
                <OfficeSuspense:OfficeSuspense>
                <AllSuspense:AllSuspense>
                <SUSuspense:SUSuspense>
                <SULocalSuspense:SULocalSuspense>
                <Adjustment:Adjustment>
                <TCReq:TxnCorrRequests>
                <TCProc:TxnCorrProcessed>
            >
        >
    >
where
```

Traded

A Riposte search criteria to select any transactions that would affect the cash position, and thereby invalidate an existing cash declaration. This will be of the form

```
     EPOSSTransaction.PM.L5 EQ "3017"
AND EPOSSTransaction.ProductNo EQ “1”
AND TxnData.Container EQ “sss”
AND NOT(Exists(EPOSSTransaction.OpeningFiguresId))
```

where *“sss”* is the stock unit id to be filled in before use. See 4.1.8.2, 4.1.10.7.1

Discrepancies

A Riposte search criteria to select brought forward figures and movements for committed discrepancies in a stock unit. This will be of the form

```
      EPOSSTransaction.PM.L5 EQ "3017"
AND   EPOSSTransaction.PM.L3 EQ "3110"
AND   TxnData.Container EQ “sss”
AND  (EPOSSTransaction.OpeningFiguresId EQ “fff” OR
      NOT(Exists(EPOSSTransaction.OpeningFiguresId)))
```

where *“sss”* is the stock unit id, and *“fff”* is the opening figures id to be filled in before use. See 4.1.8.3

OfficeSuspense

A Riposte search criteria to select the brought forward suspense position for the office. This will be of the form

```
      EPOSSTransaction.PM.L5 EQ "3017"
AND  (EPOSSTransaction.PM.L2 EQ "740" OR
      EPOSSTransaction.PM.L2 EQ "490")
AND   TxnData.Container EQ “##”
AND   EPOSSTransaction.OpeningFiguresId EQ "fff"
```

where *“fff”* is the opening figures id to be filled in before use. See 4.1.8.4.1, and Suspense Report in EA/HLD/005

AllSuspense

A Riposte search criteria to select all suspense movements (including local suspense) for all stock units. This will be of the form

```
      EPOSSTransaction.PM.L5 EQ "3017"
AND  (EPOSSTransaction.PM.L2 EQ "740" OR
      EPOSSTransaction.PM.L2 EQ "490" OR
      EPOSSTransaction.PM.L4 EQ "960")
AND  (NOT Exists(EPOSSTransaction.OpeningFiguresId))>
```

See Suspense Report in EA/HLD/005

SUSuspense

A Riposte search criteria to select the suspense movements (not including local suspense) for a particular stock unit. This will be of the form

```
       EPOSSTransaction.PM.L5 EQ "3017"
   AND (EPOSSTransaction.PM.L2 EQ "740" OR
       EPOSSTransaction.PM.L2 EQ "490")
   AND  TxnData.Container EQ “sss”
   AND  NOT(Exists(EPOSSTransaction.OpeningFiguresId))
```

where *“sss”* is the stock unit id to be filled in before use. See 4.1.8.4.1

SULocalSuspense

A Riposte search criteria to select the local suspense movements for a particular stock unit.This will be of the form

```
       EPOSSTransaction.PM.L5 EQ "3017"
   AND  EPOSSTransaction.PM.L4 EQ "960"
   AND  TxnData.Container EQ “sss”
   AND  NOT(Exists(EPOSSTransaction.OpeningFiguresId))
```

See 4.1.8.4.2.

Adjustment

A Riposte search criteria to select local adjustment event transactions (see 4.1.10.5). This will be of the form

```
       EPOSSTransaction.PM.L5 EQ "3017"
   AND  EPOSSTransaction.PM.L4 EQ "970"
   AND  EPOSSTransaction.TranType EQ "L"
```

See 4.1.8.4.3

TxnCorrRequests

A Riposte search criteria to select all transaction correction requests (see 4.10.4.1, 4.1.8.4.5)

```
WAIndex.LFSFlag EQ “TC”
```

Note that since all such requests are received from Correspondence Servers, the message store scan can be optimised by using start/end marks that restrict the search to messages originating from node 32 or above.

The query is further optimised by use of the ‘WAIndex.LFSFlag’ attribute, which is indexed by Riposte for fast access..

TxnCorrProcessed

A Riposte search criteria to select all processed transaction corrections (see 4.10.4.2, 4.5.9.2). This will be of the form

```
WAIndex.LFSFlag EQ “TP”
```

Note that since no such requests are received from Correspondence Servers, the message store scan can be optimised by using start/end marks that restrict the search to messages originating from node 31 or below.

The query is further optimised by use of the ‘WAIndex.LFSFlag’ attribute, which is indexed by Riposte for fast access.

4.11.6.4 MandatorySummaries

MandatorySummaries is an existing Riposte collection containing temporal Type C Reference data objects, and is used by EPOSSStockUnit to determine whether particular reports need to be produced prior to stock unit rollover, and how to invoke that report production. Each object in the collection corresponds to a particular report.

The syntax is now extended to support optional ‘SpecificCriteria’ if the transactions relevant to the report are not simply those under the specified root node. See also 4.5.6 and 4.9.2.1.

Note that the use of ‘SpecificCriteria’ incurs a performance penalty, since the message store must be searched explicitly.

```
<Collection:_MandatorySummaries>
<ObjectName:NN>
<StartDate:01-Jan-1996 00:00:30>
<EndDate:>
<RData:
   <Data:
      <Description:Description>
      <CutOffID:CutOffId>
      <NodeID:RootNode>
      <Enforce:Enforce>
      <SpecificCriteria:Criteria>
      <ReportButton:
         <Collection:DesktopButtonsIItemNNNNN>
         <ObjectName:IItemNNNNN>
      >
   >
>
```

where

Description A description of the report, for display in a picklist during stock unit rollover if the report should be produced

CutOffId The id of the cutoff to be checked

RootNode The node in the DataServer tree containing the tidemark to be checked against the cutoff.

Enforce True if the report must be produced before rollover when the cutoff check indicates existing report is out of date or missing.

Criteria A Report broker search criteria to select the required transactions rather than use the entire current DataServer tree. The syntax is used to build up a Riposte query. e.g.

```
<Op1:Application>
<Comp:EQ>
<Op2:APS>
```

DesktopButtonsIItemNNNNN The collection holding the report launch button

IItemNNNNN

The report launch button to be used to invoke the report. Note that this button may or may not be visible on the desktop. If a button is to be hidden/revealed by softlaunch, the same softlaunch definition should specify a Manadatory Summaries override that replaces this with the button being revealed. See 4.9.2.1

4.12 Distributed Application Services

No changes are expected in this area.

4.13 Information Management

No changes are expected in this area.

4.14 Networking Services

These will be provided by standard Riposte messaging. No changes are expected in this area.

4.15 Platforms

There are no changes to platform design or configuration except for the addition/removal of binaries and reference data as described in other sections within this document.

4.15.1 Live Counters

Impact Release 3 will be supported on live counters.

4.15.2 Training Mode

Training mode will not be supported (see CP 3842).

4.15.3 Training Counters

Impact Release 3 will be supported on training counters.

5.0 Systems Management

The system is primarily controlled through reference data.

5.1 Reference Data

The additional reference data that is required to support the design described within the scope of this document is described in section 4.11.

5.2 Receipts and Reports

These are controlled through a Global Objects mechanism whereby Riposte Message reference data is delivered to the counter in a 'DAT' file as part of the Counter upgrade. Changes to the Global Objects DAT file covered in the scope of this document are described in section 4.11.

5.3 Screens and Dialogue Flows

For screens and dialogues, see documents EA/IFS/011, EA/IFS/012 and EA/IFS/013. Particularly, EA/IFS/012 covers the majority of this document.

6.0 Application Development

The development will follow the current development standards, as described in DE/PRO/003

Each development will be covered by a low-level design document. The particular documents for this development are indicated by the compliance matrix in Appendix B – Low Level Design Cross Reference

During development, VB6 will be used, and source will be held in VSS. PVCS will be used for delivery.

All source code will be documented by inline comments as described in NB/STD/001.

7.0 System Qualities

The primary aim of the solution adopted for Impact Release 3 Balancing and handling stock by volume is to ensure the risks associated applying the required changes are mitigated as far as possible. Additionally the following System Qualities are considered.

7.1 Availability

Changes arising from Impact Release 3 have no affect on the availability of the Counter systems.

7.2 Usability

The changes to the counter for Impact Release 3 are primarily designed to make the system more useable from a back-office perspective. The increase in the duration of the trading period means that balancing does not have to occur as frequently. In addition, cash may be controlled in a tighter manner via changes to the Cash Declaration process and the reporting of cash variances. Finally, any error generated by branches as a result mistakes made in pervious periods can be corrected automatically via the introduction of Transaction Corrections.

7.3 Supportability

No changes to system supportability are perceived as a result of Impact Release 3.

However, additional events will be reported to the event log to enhance available diagnostic information. These additional events are described in section 4.11.

7.4 Security

No changes to system security are perceived as a result of Impact Release 3. However, the End of Day task that provides Protection Against Loss of Data (4.6.5) must run with Administrator privileges in order to modify the registry settings for the Riposte service. To minimize the amount of code that executes with privilege, this task is run as a distinct binary with very specific and limited functionality.

7.5 Potential for Change

There is a significant potential for change in all areas of this design. At the time of writing there are numerous change requests and 'soft' change requests that have not been considered.

The design herein will not change until such change requests have been approved.

8.0 Solution Implementation Strategy

8.1 Migration

Migration of counter software to the functionality described in this document is described in the Migration High Level Design EA/HLD/008. References are made throughout this document to the various migration points that are stipulated in EA/HLD/008.

All migration issues are covered in the body text of this document.

9.0 Costs, Risks and Timescales

Costs are dealt with in the plans.

Timescales are documented in the plans.

Consideration to the design from the outset has identified two areas of attention requiring analysis for risk:

- Technical Risks
- Management Risks

Every effort has been made in the production of this high level design to mitigate against the unnecessary overheads that could be caused by these risks in consideration of the design.

9.1 Technical Risks

The following technical risks have been identified in consideration to the design.

Firstly the full implications of merging Cash Declaration with ONCH have taken a while to become clear, and have resulted in a change to the life-cycle of Cash Declarations in order to support the Cash Variance Report. It is possible that that this change will have unexpected consequences for normal declaration and rollover.

Secondly the changes to the counter applications, and in particular EPOSS, are invasive. The EPOSS Application does not lend itself easily to invasive change. Its component elements are few in number but a single element may have multiple uses in the functioning counter, hence a change can have an impact in areas that are not required for change in a business sense. There is hence a very real risk of regression in any change made to EPOSS under the changes required for Impact Release 3.

These risks will be addressed by ongoing liaison between implementors and designers, to ensure that specific changes are not made in isolation without regard to impact elsewhere. In some cases, the risk can be reduced by ‘clone and tweak’ rather than modifying the existing implementation in-line. This at least provides a regression option.

9.2 Management Risks

The following management risks have been identified in consideration of the design.

Firstly the timescales for delivery of S80 are recognised to be tight and if at all possible it is required that development should commence before the design is brought to final approval. There is an obvious risk that changes to the design can then require development rework. Changes to the design in this document (particularly the Cash Variance Report and supporting mechanisms) have already overlapped with initial implementation.

Secondly, the high level design is one of two complementary HLD for changes to the Counter in response to Impact Release 3. Ideally the two documents hence should be levelled to the same degree and as stated, complement each other. There is inevitably a certain amount of overlap, which means that implementors need to be familiar with both documents (at least in outline) to ensure that design overlap between the two does not result in duplicated or even conflicting implementation.

FUJ00091090

10.0 Appendix A – Design Proposal Cross Reference

In conjunction with the equivalent table in EA/HLD/005, the following compliance matrix resolves each reference provided in the Design Proposal EA/DPR/004 to ensure all Counter requirements are addressed by the two Counter High Level Designs.

Design Proposal Reference		HLD References
2.5.1.1	**Branch Transactions**	
2.5.1.1.2.3	Stock Revaluation	Revaluation (4.8)
2.5.1.1.7	Reversal Control	Transaction Correction Reversal Control (4.5.7)
2.5.1.2	**Changes to Cash/Stock Declarations and Handling of variances**	
2.5.1.2.1	Changes to Cash Declarations	Cash Declaration (4.1.5)
2.5.1.2.1.1	Check for Variances Function	Check for Variances (4.1.6)
2.5.1.2.1.2	Declaration Events	Check Variance Event Recording (4.1.6.4)
2.5.1.2.1.3	Variance Persistent Objects	Variance History Update (4.1.6.3)
2.5.1.2.2	Reporting of Cash Variances	Cash Variance Report (4.1.9)
2.5.1.2.3	Processing of Cash Variances	Add/Remove Cash (4.1.10)
2.5.1.2.4	Stock Declarations and Variances	Stock Declaration (4.2), Rollover Discrepancy Checking (4.4)
2.5.1.2.5	Non-Value Stock Declarations	Non-Value Stock Declaration and Reporting (4.3)
2.5.1.3	**Reporting**	
2.5.1.3.1	Remuneration Reporting	Remuneration Reporting (4.9.1)
2.5.1.3.5	APS Transactions Report	APS Transactions Report (4.9.2)
2.5.1.5	**EOD**	
2.5.1.5.1	Removal of LFS Weekly Stock Reporting functions	Weekly Stock Statements (4.6.1.1)
2.5.1.5.2	POL FS Summarisation at counter	POL FS Summarisation (4.6.2)
2.5.1.5.3	Maintenance of Office Variances Persistent Object	Maintain Office Variances (4.1.8)
2.5.1.5.4	LFS EOD functionality changes to handle changes in Cash Declarations	LFS Cash Statement Generation (4.1.3)
2.5.1.5.5	Simplification of EPOSS Reconciliation	EPOSS Reconciliation (4.6.4)
2.5.1.5.6	Protection against lost data	Protection Against Loss of Data (4.6.5)
2.5.1.6	**Logon Checks**	
2.5.1.6.1	ONCH run for "yesterday"	Cash Declaration Check (4.7.2)
2.5.1.6.2	Stock Unit in correct Trading Period	Trading Period Check (4.7.3)
2.5.1.6.3	Outstanding Transaction Corrections	Outstanding Transaction Corrections (4.7.4)
2.5.1.6.4	Protection against Data Loss	Rollover Warning (4.7.1)
2.5.1.7	**Process Transaction Correction**	Transaction Correction Processing (4.5)
3.3	**Internal Interfaces**	
3.3.2	TMS to Branch: Transaction Corrections	Transaction Correction Messages (4.10.4)

11.0 Appendix B – Low Level Design Cross Reference

The following compliance matrix identifies which Low Level Design documents are responsible for addressing the elements in this High Level design.

HLD Reference		Responsible LLD
4.1	**Cash Declarations and Variance Reporting**	
4.1.1	Validity of Cash Declarations	EP/LLD/013 (EPOSSDeclare Low Level Design)
4.1.2	Removal of ONCH Declarations	
4.1.3	LFS Cash Statement Generation	
4.1.4	Changes to Cash Discrepancy Processing	
4.1.5	Cash Declaration	
4.1.6	Check for Variances	
4.1.7	Retrospective Cash Declaration at Logon	
4.1.8	Maintain Office Variances	EA/LLD/007 (Maintain Office Variances Low Level Design)
4.1.9	Cash Variance Report	EA/LLD/015 (Impact Release 3 Receipts & Reports)
4.1.10	Add/Remove Cash	EP/LLD/013 (EPOSSDeclare Low Level Design)
4.1.11	Cash Auto-Declaration on Inactive Rollover	
4.1.12	Pruning of Cash Declaration Objects	
4.1.13	Pruning of Variance History Objects	
4.2	**Stock Declaration**	
4.3	Non-Value Stock Declaration and Reporting	EA/LLD/004 (Counter Balancing Functional Low Level Design)
4.4	Rollover Discrepancy Checking	
4.5	Transaction Correction Processing	EA/LLD/008 (Impact Release 3 Transaction Correction Application), EP/LLD/012 (EPOSSCore Low Level Design)
4.6	**Changes to End-of-Day**	
4.6.1	LFS End of Day	EP/LLD/013 (EPOSSDeclare Low Level Design)
4.6.2	POL FS Summarisation	N/A
4.6.3	Maintain Office Variances	EA/LLD/007 (Maintain Office Variances Low Level Design)
4.6.4	EPOSS Reconciliation	EA/LLD/004 (Counter Balancing Functional Low Level Design)
4.6.5	Protection Against Loss of Data	**Error! Reference source not found.** (Data Protection Low Level Design)
4.7	**Changes to Log-On Checks**	
4.7.1	Rollover Warning	EA/LLD/019 (Data Protection Low Level Design)
4.7.2	Cash Declaration Check	EP/LLD/013 (EPOSSDeclare Low Level Design)
4.7.3	Trading Period Check	EA/LLD/006 (Impact Release 3 – Trading Period Rollovers Low Level Design)
4.7.4	Outstanding Transaction Corrections	EA/LLD/008 (Impact Release 3 Transaction Correction Application)
4.8	Revaluation	EA/LLD/004 (Counter Balancing Functional Low Level Design)
4.9	Other Reporting	EA/LLD/015 (Impact Release 3 Receipts & Reports)