


Fujitsu
Services

TPS Agents for BI3: High Level Design

Ref: AD/DES/041

COMPANY IN-CONFIDENCE

Version: 7.0
Date: 09/08/05

Document Title: TPS Agents for BI3: High Level Design**Document Type:** Design Specification**Release:** BI3 S90**Abstract:** This document is the High Level Design of the Transaction Processing System (TPS) agents up to and including Release BI3 S90. It provides an external description of the agents, their interfaces and the way in which they operate.**Document Status:** APPROVED**Author & Dept:** Rex Dixon (Tel:  /Development Unit – Design Team**Contributors:** Gareth Jenkins, Paul Callow, Paul MacNeill**Internal Distribution:** POA Library Agent Document Library
Reviewers**External Distribution:** None**Agent Team Ref:** TSC/AGT/074**WPre:** c:\agtdev\documents\tsc_agt_074_tps_agent_csr+_hld.doc**Approval Authorities:**

Name	Position	Signature	Date
Gareth Jenkins	RASD Design Authority		
Andy Kennedy	Development Unit – Design Authority		

0. DOCUMENT CONTROL

0.1 Document History

Version No.	Date	Reason for Issue	Associated CP/PinICL No.
0.1		This is the first issue for CSR+ and is essentially a copy of the CSR version without modification.	
0.2	08/09/00	All references to BES and BPS removed.	
0.3	28/09/00	<i>Output Table Format</i> and <i>Mapping of Riposte Messages to Output Tables</i> sections moved to AD/DES/047 - TPS Tables and Mappings for CSR+	
0.4		<i>Mapping of Riposte Messages to Output Tables</i> sections moved back from AD/DES/047 Algorithm for start of CAP year changed Logout events revised Harvesting of stock (cash and stamps) at lowest declared level TPS Reconciliation added TMS_RX_MIGRATION_OBJECT table added for Never-Polled Offices Report Deleted stock units found through DelStockUnits collection Description of chunk mechanism removed, as it is now generic and is described in [GEN] Description of EOD handling removed, as it is now generic and is described in [GEN]	PC/25140 CP/1957 PC/36983 CP/2096 CP/2186 CP/2684 PC/55745 PC/41948 -
1.0	10/01/01	Corrections and clarifications as a result of comments received. This version has been frozen for CSR+	
1.1	24/01/02	Network Banking enhancements	CP/3161
1.2	15/03/02	Note added to Section 2.8.3.9	
1.3	11/07/02	Additions for NBS and DCS EPOSS Transactions Harvesting of dates for OBCS Status transactions changed	CP/3263 PC/64962
1.4	17/04/03	Additions for Electronic Top-Ups Additions for Bureau de Change	CP/3448 CP/3394
2.0	02/05/03	Minor corrections as a result of review comments received	
2.1	06/06/03	Changes in harvesting reversal, especially for DCS	PC/91074 PC/91231

Fujitsu
Services

TPS Agents for BI3: High Level Design

Ref: AD/DES/041

COMPANY IN-CONFIDENCE

Version: 7.0
Date: 09/08/05

3.0	04/07/03	Approved version for S50	
3.1	28/11/03	Adding Chart of Account Messages	CP3585/3612
4.0	08/01/04	Approved version for S60	
4.1	18/10/04	Many minor changes for IMPACT Release 3	CP3716
5.0	17/01/05	Approved version for S80. Added section on T_LD_TC copied from approved document EA/HLD/010 version 1.0	
5.1	06/05/05	Addition of harvesting EPOSS Event 17 Other changes are documentary only and do not represent changed functionality at S90: <ul style="list-style-type: none"> – Removal of harvesting Stock Unit Rollover Trailers (March 2001) – Correction of inconsistency in event ids (S80) – Added background information from [EA/HLD/010] on IMPACT Release 3 changes at S80 – Tidying up presentation of Chart of Accounts harvesting (S60) 	CP3955 PC0064428
6.0	15/06/05	Approved version for S90 (later superseded), with minor editorial changes as a result of comments received	
6.1	23/06/05	Harvesting EPOSS Event 17 replaced by Event 68	PC0122229
6.2	11/07/05	Addition of default for missing transaction mode for APS and Mails transactions	PC0109219 & PC0114905
7.0	09/08/05	Approved version for S90, with minor change for S60 not previously documented: <ul style="list-style-type: none"> – Added “RISP”, “ROSP” & “RODP” transaction modes 	CP3612

0.2 Review Details

Review Comments by:

Review Comments to:

Rex Dixon

Mandatory Review Authority	Name
RASD Design Authority	Gareth Jenkins(*)
DU Design Authority	Andy Kennedy(*)
CS Network Service Manager	Alex Kemp(*)
CS Data Centre & Ops Service Manager	Peter Thompson(*)

CS Business Continuity Manager	Tony Wicks(*)
CS Infrastructure & Availability Manager	Carl Marx(*)

Optional Review / Issued for Information	
<i>(Optional Reviewer roles:)</i>	
DU S90 Release Manager	Mark Taylor
DU Test Designer	Peter J Robinson
<i>(Issued for information to):</i>	
Development Unit - Design Team	Roger Barnes, Phil Hemingway
Development Unit - Development Team	Peter Ambrose, Paul MacNeill, Keith Toh, Tim Trollope, Anne Mohan, John Rayner, David Harrison, Julie Havard

(*) = Reviewers that returned comments

0.3 Document Cross-References

Tag	Ref	Vers	Date	Title	Source
[COM]	AD/DES/042 (TSC/AGT/076)			High Level Design of Common Agents	PVCS/ agent lib
[CP3955]	CP/3955			Addition of bureau revaluation icon	PVCS
[E2E1HLD]	EA/HLD/002			End To End Release 1 – High Level Design	PVCS
[EAG]	EP/DES/002			EPOSS Attribute Grammar Catalogue	PVCS
[EODGEN]	TD/DES/124			End of Day Marker Generation HLD	PVCS
[EODHT]	TD/DES/018			EOD Harvest Trailer Generation HLD	PVCS
[EPOSSBAL]	EP/DES/021			EPOSS Balancing Service – High Level Design	PVCS
[EPOSSEOD]	EP/DES/025			EPOSS End of Day Service High Level Design	PVCS
[EPOSSMIG]	EP/DES/023			EPOSS Migration Services High Level Design	PVCS
[GEN]	AD/DES/039 (TSC/AGT/058)			Generic Agent Components for CSR+ High Level Design	PVCS/ agent lib
[GENTABS]	AD/SPE/006 (TSC/AGT/079)			Agents Generic Database Table Specifications for BI3	PVCS/ agent lib
[HRSAP_HLD]	EA/HLD/009			TPS HR SAP Summarisation & Transaction Corrections HLD	PVCS
[IMPACT3]	EA/HLD/010	2.0	24/02/05	IMPACT Release 3: Agents High Level Design	PVCS
[IMP3_STOCK]	EA/HLD/005			IMPACT Release 3 Counter Design for Balancing, Rollover and Stock Processing	PVCS

[IMP_S90]	EA/DPR/007			Impact Changes for S90 Design Proposal	PVCS
[MAESTRO]	AD/DES/032 (TSC/AGT/063)			Design of Agent Maestro Schedules for CSR+	PVCS/ agent lib
[NBSFLOWS]	NB/IFS/004			Network Banking Message Flows and Interfaces	PVCS
[OBCS]	AD/DES/043 (TSC/AGT/078)			OBCS Agents High Level Design for CSR+	PVCS/ agent lib
[RDBDC]	RD/IFS/034			Bureau de Change Phase 1 – Reference Data Interface to Counters	PVCS
[TED]	TD/ARC/001			Technical Environment Description	PVCS
[TPSTABS]	AD/DES/047 (TSC/AGT/075)			Pathway Agents: TPS Tables and Mappings for CSR+	PVCS/ agent lib
[TPSCSR+]	BP/DES/026			End to End Design for Enhancements to TPS at CSR+	PVCS

Where explicit versions are specified, these are the versions that have been consulted in the preparation of the current document.

0.4 Terminology

Term	Meaning
AMT	Agent Marker Table
AWT	Agent Work Table
BP	Balance Period
CAP	Cash Account Period
CofA	Chart of Accounts
DCS	Debit Card System (for EFTPoS)
EOD	End of Day
ETS	Electronic Top-Ups Service
NBS	Network Banking Service
POA	Post Office Account (Team)
TC	Transaction Correction
TMS	Transaction Management System. This is basically that part of Riposte that handles the messaging between the Post Offices and the Correspondence Servers.
TP	Trading Period
TPS	Transaction Processing System
Primary Harvesting	The harvesting of messages between the marker in one EOD Marker message and that in the next one.
Secondary Harvesting	The harvesting of messages between the EOD Marker message and the EOD Harvest Trailer message.

0.5 Changes in this Version

Changes	Associated CP/PinICL No.
See Document History section	

0.6 Changes Expected

Changes	Associated CP/PinICL No.
None	

0.7 Contents

0. DOCUMENT CONTROL	2
0.1 DOCUMENT HISTORY	2
0.2 REVIEW DETAILS	3
0.3 DOCUMENT CROSS-REFERENCES	4
0.4 TERMINOLOGY	5
0.5 CHANGES IN THIS VERSION	6
0.6 CHANGES EXPECTED	6
0.7 CONTENTS	6
1. GENERAL	8
1.1 INTRODUCTION	8
1.2 CONVENTIONS	8
2. TPS BULK HARVESTER (T_HV_ALL)	9
2.1 FUNCTIONAL DESCRIPTION	9
2.1.1 Use of Multiple Database Partitions	9
2.1.2 Organisational Units	10
2.1.3 Outlet Migration Complete Marker	10
2.1.4 Message Selection for Primary Harvesting	10
2.1.5 Message Selection for Secondary Harvesting	13
2.1.6 Handling event messages	14
2.1.7 Generating CAP and BP values	17
2.1.8 Handling (Stock Unit) Rollover Trailers	20
2.1.9 Handling CAP Trailers	20
2.1.10 Harvester exceptions	24
2.2 CONTROL INTERFACES	25
2.3 MAESTRO SCHEDULE	25
2.4 REGISTRY ENTRIES USED	25
2.5 EXCEPTION HANDLING THROUGH ORACLE	26
2.6 ORACLE TABLE FORMAT	26
2.6.1 TPS-Specific Database Tables	26
2.6.2 Generic Database Tables	27
2.7 RIPOSTE MESSAGE FORMAT	27
2.8 MAPPING OF RIPOSTE MESSAGES TO OUTPUT TABLES	28
2.8.1 The Output Tables	28
2.8.2 General Notes	30
2.8.3 Specific Notes	33

3. TPS TRANSACTION CORRECTIONS BULK LOADER (T_LD_TC)	38
3.1 FUNCTIONAL DESCRIPTION	38
3.2 CONTROL INTERFACES	38
3.3 MAESTRO SCHEDULE	39
3.4 REGISTRY ENTRIES USED	39
3.5 EXCEPTION HANDLING THROUGH ORACLE	39
3.6 ORACLE TABLE FORMAT	40
3.6.1 <i>TMS_TX_TPS_TC_DETAIL</i>	40
3.6.2 <i>Generic Database Tables</i>	40
3.7 RIPOSTE MESSAGE FORMAT	40
3.8 MAPPING OF ORACLE TABLES TO RIPOSTE MESSAGES	40
APPENDIX A. OUTLET BALANCING CONCEPTS	42

1. GENERAL

1.1 Introduction

This document is the high level design of the Transaction Processing System (TPS) agents produced for BI3. This version pertains to release BI3 S90.

The additional functionality at S90 (over S80) concerns the harvesting of a Revaluation event for [CP3955]. The associated Design Proposal is [IMP_S90].

This high level design is based on the agent architecture (defined in [TED]). Much of the detailed design of the agents is generic to many agents and this generic agent design is covered in [GEN]. The document concentrates on those aspects of the agent design that are specific to the TPS business functionality.

Section 2 provides an external description of the TPS Bulk Harvester, its interfaces and the way in which it operates, and section 3 does likewise for the TPS Transaction Corrections Bulk Loader.

An associated document, [TPSTABS], provides the detailed mappings of Riposte Messages to Oracle Tables.

APPENDIX A provides the reader with a rough guide to some of the relevant concepts, such as stock units, cash account periods and rollovers.

As this document contains a detailed level of information, no lower level design documentation is considered necessary.

1.2 Conventions

The following conventions are used:

- a) Bits of attribute grammar marked with an asterisk (“*”) indicate that the field may be repeated a number of times.

This issue of the document has a few notes formatted like this. They act as an aside from the main flow of text, wherever this seems appropriate.

2. TPS BULK HARVESTER (T_HV_ALL)

This section describes the functionality and interfaces for the TPS Bulk Harvester agent for CSR+. The following topics are covered in different subsections:

- 1) Functional Description
- 2) Control Interfaces
- 3) Maestro Schedule
- 4) Registry Entries Used
- 5) Exception Handling through Oracle
- 6) Oracle Table Format (also see [TPSTABS])
- 7) Riposte Message Format
- 8) Mapping of Riposte Messages to Oracle Tables (also see [TPSTABS])

2.1 Functional Description

The TPS Bulk Harvester is a standard bulk harvester agent (see [GEN]). However, a number of special facilities have been added for the specific functionality required. This section describes these functions in the following areas:

- a) Use of Multiple Database Partitions
- b) Organisational Units
- c) Outlet Migration Complete Marker
- d) Message Selection during the Primary Scan
- e) Message Selection during the Secondary Scan
- f) Handling event messages
- g) Generating CAP and BP values
- h) Handling Rollover trailers
- i) Handling Cash Account trailers
- j) Harvester exceptions

These are described in the following sub-sections.

2.1.1 Use of Multiple Database Partitions

In order to provide the necessary performance in the TPS Host application, a number of identical sets of tables are required to hold subsets of the data harvested from Post Offices. This is done by having 64 separate partitions, each holding an instance of each of the TPS Transaction and TPS Reconciliation Tables described in 2.6.1. For each such table, there is one underlying table per partition, each with a different name. The TPS database provides 64 separate views for different users each of which maps onto one set of tables. Each instance of the TPS Bulk Harvester will use a different username (configured in the Registry using the *guise* passed to the agent to find the registry subentry) and thus will connect to a separate set of tables.

Note that there is a single instance of the TPS Control tables (such as TPS_OUTLET_EODS) and the generic database tables (such as TMS_AMT_TPS, see 2.6.2). These are not partitioned.

2.1.2 Organisational Units

The TPS Host requires that all records harvested are identified by the originating group's "organisational unit id", in column `org_unit_id`, as well as by the group id.

An error message output by the harvester wrongly refers to this entity as "originating unit" rather than "organisational unit".

The `org_unit_id` and its associated version number `org_unit_version_no` are obtained by looking up the group id in the `TPS_OUTLETS` table. This lookup is performed when starting to harvest the group and then cached until the group id changes. If the group id is not in the table, the group is not harvested and an error message is written to the NT Event Log.

2.1.3 Outlet Migration Complete Marker

The hook that selects the lower marker for the Primary Scan obtains the marker from the Agent Marker Table (see [GEN]). If this indicates that the outlet has never previously been harvested, the TPS Harvester attempts to read the Outlet Migration Complete Marker object (really an Indicator rather than a Marker) to determine whether MiMan migration of the outlet is complete. If migration is complete, a row is inserted into `TMS_RX_MIGRATION_OBJECT` (provided that it does not already exist).

The object is in collection `MiMAN` with an `ObjectName` of `##`. Migration is complete if (and only if) `<Data.MigStatus:>` has the value "C".

Note that this table does not have a column for `org_unit_id`. This is so that an outlet can appear on the Never-Polled Offices Report even if the outlet is not yet configured in `TPS_OUTLETS`.

2.1.4 Message Selection for Primary Harvesting

A number of different types of message are of interest during the Primary Scan. These are:

- a) Standard EPOSS Transaction message
- b) APS EPOSS Transaction message
- c) OBCS EPOSS Transaction message
- d) NBS EPOSS Transaction message
- e) ETS EPOSS Transaction message (from BI3 S50)
- f) DCS EPOSS Transaction message
- g) Bureau de Change EPOSS Transaction message (from BI3 S50)
- h) OBCS Order Book status messages (i.e. received, issued, impounded)
- n) EPOSS (and some non-EPOSS) Counter Events
- o) Stock Unit RolloverTrailer messages, to find their associated Revaluation messages
- p) CAPTrailer messages, in order to find their associated Stock Holding and Cash Account messages

An attempt is made to repair faulty EPOSS transactions – those for which the `<TxnData.Start:>` attribute is empty or absent – by using corresponding attributes from a later message. The later message used is the first one within the next 10 with the same value of `<TxnData.SessionId:>`

and that has the required <TxnData.Start.Date:> and <TxnData.End.Date:> attributes.
(PC/22496)

Since the TPS harvester needs to process most of the messages within Riposte, it has been decided that it is best to call **RiposteScanMessage** without any filtering during the primary harvesting and do all the filtering within the agent. The following pseudocode defines the selection criteria for the various types of messages to be processed:

```
// Ignore internal products (CP/1276 & CP/1578)
If ((<EPOSSTransaction.ProductNo:> exists)
    AND
    (<EPOSSTransaction.ProductNo:>
        between 10000 and 11299 OR between 11400 and 19999 OR
        equals 851))
    ignore the message
```

Product 851 is a BES Helpdesk payment, so is obsolete.

11400 should perhaps be changed to 11500 (see PC/82679). However, the products in this range are not passed to TIP, and from S50 the Data Warehouse feed ceases, so not changing it has no impact.

```
// EPOSS Events
If ((<EPOSSTransaction.TranType:E>)
    AND
    (<EPOSSTransaction.ID:> is NULL, 41, 9, 10, 7, 8, 6, 24, 65,
        66, 67, 58, 59, 60, 61, 62, 68))
    Process as an EPOSS Event (see section 2.1.6)

// APS EPOSS Transactions
If ((<EPOSSTransaction.TranType:S>)
    AND
    (<Application:APS>))
    Process as an APS Transaction (see section 2.8.3.1)

// OBCS EPOSS Transactions
If ((<EPOSSTransaction.TranType:S>)
    AND
    (<Application:OBCS>))
    Process as an OBCS Transaction (see section 2.8.3.11)

// NBS EPOSS Transactions and
// ETS EPOSS Transactions (from BI3 S50)
If ((<EPOSSTransaction.TranType:S>)
    AND
    ((<Application:NBA>) OR (<Application:ETA>))
    AND
    <EPOSSTransaction.AdditionalData.C2Mess.Ctrl.MsgType:C1>)
    Process as an NBS Transaction (see section 2.8.3.9)

// DCS EPOSS Transactions
If ((<EPOSSTransaction.TranType:S>)
    AND
    (<Application:DCA>))
```

```
AND
<EPOSSTransaction.AdditionalData.C2Mess.Ctrl.MsgType:C1>
    Process as a DCS Transaction (see section 2.8.3.6)

// Bureau de Change EPOSS Transactions (from BI3 S50)
If ((<EPOSSTransaction.TranType:S>)
    AND
    (<Application:> commences with "BDC")
    Process as a Bureau de Change Transaction (see section 2.8.3.2)

// Any other EPOSSTransaction message
If ((<EPOSSTransaction.TranType:S>)
    Process as an EPOSS Transaction (see section 2.8.3.8)

// OBCS Status messages
If ((<TranType:ADMIN>)
    AND
    (<Application:OBCS>)
    AND
    (<ObjectName:>) // null
    AND
    (NOT (<Data.State:5><Data.Result:1><Data.Vouchers:0>)))
    Process as an OBCS Status message (see section 2.8.3.10)

// Stock Unit Rollover Trailer
If ((<Data.TranType:RolloverTrailer>)
    AND
    (<Data.Container:> not ##))
    Process as a Rollover Trailer (see section 2.1.8)

// CAP Trailer
If (<Data.TranType:CAPTrailer>)
    Process as a CAP Trailer (see section 2.1.9)

// Forced Logout events (M1 counters)
If ((<Application:UserAccessControl>)
    AND
    (<Type:InactivityLogout>) OR (<Type:AdminLogout>))
    Process as an EPOSS Event (see section 2.1.6)

// Forced Logout events (pre-M1 counters)
If ((<Application:UserAccessControl>)
    AND
    (<Type:> does not exist)
    AND
    (<LogoutAuthority:> exists))
    Process as an EPOSS Event (see section 2.1.6)
```

<i>This change was introduced at CI4 M1 (PC/36983)</i>
--

```
// Other Logout events
If (((<SecurityEvent.EventName:FailedLogon>)
    OR
    (<Logon:> exists)
    OR
    (<Logout:> exists))
    Process as an EPOSS Event (see section 2.1.6)

Else
    We don't want this message so ignore it.
```

2.1.5 Message Selection for Secondary Harvesting

The generic harvesting mechanism defined in [GEN] includes a secondary harvesting scan of all the messages on the Gateway Counter (node 1) from the EOD Marker message to the EOD Harvest Trailer message inclusive. The TPS Harvester employs such a Secondary Scan.

The EOD Marker is included as the first message in the scan so that it itself can be harvested, as an EPOSS Event (see section 2.1.6).

A number of different types of message are of interest during the Secondary Scan. These are selected by supplying a filter to **RiposteScanMessage** (where *dd-Mon-yyyy* is the EOD date being harvested).

- a) The EOD Marker message itself, recognised by:
<Application:EOD> and
<EODMark.TranType:EODMark> and
<EODMark.EODDate:dd-Mon-yyyy>
- b) The TPS Daily Reconciliation Trailer, if any, recognised by:
<Application:EPOSSDailyReconTrailer> and
<EPOSSTransaction.TranType:DailyReconTrailer> and
<EPOSSTransaction.EODDate:dd-Mon-yyyy>
- c) The TPS Weekly Reconciliation Trailer, if any, recognised by:
<Application:EPOSSWeeklyReconTrailer> and
<EPOSSTransaction.TranType:WeeklyReconTrailer> and
<EPOSSTransaction.EODDate:dd-Mon-yyyy>
- d) The Chart of Accounts Summary Trailer, if any, recognised by:
<Application:CofASummary> and
<EPOSSTransaction.TranType:CofATrailer> and
<EPOSSTransaction.EODDate:dd-Mon-yyyy>

2.1.5.1 Harvesting the Daily and Weekly Reconciliation Trailers

A tertiary scan of the Gateway Counter (node 1) is initiated to find the TPS Daily/Weekly Reconciliation message(s) associated with the TPS Daily/Weekly Reconciliation Trailer. The low marker for this scan is the EOD Marker message itself and the high marker is the Reconciliation Trailer.

The Reconciliation messages are recognised by having the same values for:
<Application:> and
<EPOSSTransaction.ReconciliationTrailerId:> and

<EPOSSTransaction.EODDate:>
as the Reconciliation Trailer.

The Reconciliation messages to be harvested are further recognised by the value of
<EPOSSTransaction.TranType:>:

<EPOSSTransaction.TranType:>	Harvested as
DailyTxnCT	Daily Reconciliation Transaction totals (see 2.8.3.14)
DailyCAErr	Daily Reconciliation Cash Account counter errors (see 2.8.3.15)
WeeklyCAErr	Weekly Reconciliation Cash Account counter errors (see 2.8.3.16)

2.1.5.2 Harvesting Chart of Accounts Summary Messages

From S60 onwards, a tertiary scan of the Gateway Counter (node 1) is initiated to find the Summary message(s) associated with the CofA Summary Trailer. The low marker for this scan is the EOD Marker message itself and the high marker is the CofA Summary Trailer.

The Summary messages are recognised by having the same values for:
<Application:> and
<EPOSSTransaction.CofASummaryTrailerId:> and
<EPOSSTransaction.EODDate:>
as the CofA Summary Trailer.

The Summary messages to be harvested are further recognised by the value of
<EPOSSTransaction.TranType:>:

<EPOSSTransaction.TranType:>	Harvested as
CofASummary	Chart of Accounts Summary record (see 2.8.3.4)
CofATransaction	Chart of Accounts Transaction detail record (see 2.8.3.5)

2.1.6 Handling event messages

There are a number of different types of Event that need to map onto rows in TMS_RX_EPOSS_EVENTS. These are:

- EPOSS Transactions with <EPOSSTransaction.TranType:E> (i.e. EPOSS Events) and <EPOSSTransaction.Id> with values of NULL, 41, 9, 10, 7, 8, 6, 24, 65, 66, 67, 58, 59, 60, 61, 62, 68
- InactivityLogout messages
- AdminLogout messages
- LogoutAuthority messages
- FailedLogon messages
- Logon messages
- Logout messages

h) EOD Marker messages

EPOSS Events contain zero or more “parameter” attributes (<EPOSSTransaction.P.Pn:> where ‘n’ is 1, 2 or 3), which need to be included in the TMS_RX_EPOSS_EVENTS depending on the value of <EPOSSTransaction.Id>. The columns that are affected by this mapping are:

- a) event_id
- b) cash_account_week
- c) balance_period
- d) changed_container
- e) changed_status
- f) changed_user
- g) adjustment_amount
- h) txn_correction_amount

1) event_id is mapped as follows:

EPOSS Event?	<EPOSSTransaction.Id>	event_id	Comment
Y	41	914	
Y	9	915	
Y	10	916	
Y	7	917	
Y	6	919	
Y	24	933	
Y	65	6299	Trading Statement Created
Y	66	6300	Trading Statement Period rolled
Y	67	6301	Trading Statement Period Roll Abandoned
Y	58	6302	Excess Cash Removed
Y	59	6303	Cash Shortage Made Good
Y	60	6304	Cash Variance Report Previewed
Y	61	6305	Cash Variance Report Printed
Y	62	6306	Outstanding Transaction Correction Reminder Displayed
Y	68	7156	Revaluation
Y	other (i.e. 8 or null)	0	
InactivityLogout		938	Forced Logout
AdminLogout		933	Manager Forced Logout

LogoutAuthority: Desktop		938	Forced Logout
LogoutAuthority: other		933	Manager Forced Logout
FailedLogon		932	Failed Logon
Logon		930	Logon
Logout		931	Logout
EOD Marker		923	End of Day

*InactivityLogout and AdminLogout were introduced at
CI4 M1 (PC/36983)*

*Events of type 913 are also sent to TIP but they
are derived by the TPS host from
TMS_RX_OBCS_STATUSES instead of from
TMS_RX_EPOSS_EVENTS.*

- 2) cash_account_week is mapped as follows:

```
if (event_id = 914)
  cash_account_week = value of <EPOSSTransaction.P.P1:>
else if (event_id = 919)
  cash_account_week = value of <EPOSSTransaction.P.P2:>
else if (event_id = 915 or 916)
  cash_account_week is derived as for CAP attribute
fi
```

- 3) balance_period is mapped as follows:

```
if (event_id = 919)
  balance_period = value of <EPOSSTransaction.P.P3:>
fi
```

- 4) changed_container is mapped as follows:

```
if (event_id = 916 or 919)
  changed_container = value of <EPOSSTransaction.P.P1:>
else if (event_id = 915 or 917)
  changed_container = value of <EPOSSTransaction.P.P2:>
fi
```

- 5) changed_status is mapped as follows:

```
if (event_id = 915)
  changed_status = value of <EPOSSTransaction.P.P1:>
fi
```

- 6) changed_user is mapped as follows:

```
if (event_id = 917)
  changed_user = value of <EPOSSTransaction.P.P1:>
```

```
fi
```

- 7) adjustment_amount is mapped as follows:

```
if (event_id = that for EPOSS Event 58 or 59)
    adjustment_amount = value of <EPOSSTransaction.P.Pl:>
fi
```

- 8) txn_correction_count is mapped as follows:

```
if (event_id = that for EPOSS Event 62)
    txn_correction_count = value of <EPOSSTransaction.P.Pl:>
fi
```

2.1.7 Generating CAP and BP values

Many of the records passed to TPS need to include the identifier of the Cash Account Period (CAP) and the Balance Period (BP) at which it was originally written. This information is obtained by looking at the markers that are written to the Riposte Message store at the time that CAP and/or BP change (i.e. BP can change without CAP changing, but not vice versa). There are a number of these messages which link together in a fairly complex structure, which between them contain the current Riposte marker and the identifier of the CAP or BP that is just starting.

With the new accounting regime introduced by IMPACT 3 at S80, the concepts of CAP and BP disappear and are replaced by a new concept of Trading Period (TP). Individual stock units will migrate from CAP/BP to TP. As defined in [IMP3_STOCK], the StockUnit Object will contain a <Data.TP:1> attribute when the stock unit has migrated to operate in *Branch Trading Period* mode. The TPS Harvester retains the functionality for calculating CAP and BP for stock units that have not migrated. If the <Data.TP:1> attribute is present in the StockUnit Object, the CAP and BP are harvested as NULL. (The NOT NULL constraints on these two columns are removed at the appropriate point during the migration.)

There are two cases to be considered:

- a) Normal EPOSS Transactions or Events occurring in Normal Stock Units
- b) Non-EPOSS Events or Events occurring in the Default Stock Unit

These are defined in the following sub-sections.

2.1.7.1 Normal EPOSSTransactions or Events occurring in Normal Stock Units

The following diagram is copied from [EAG].

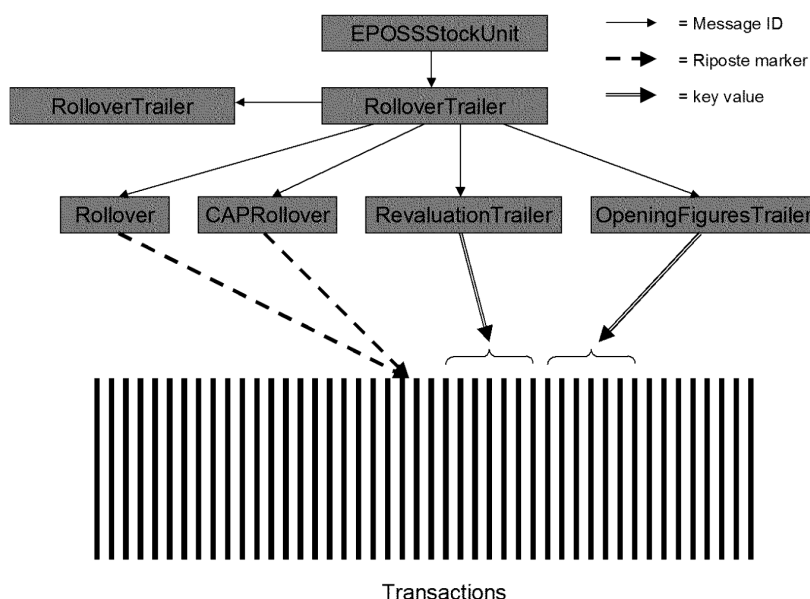


Figure 1 – Stock Unit Account Structure

The following outlines the logic required to calculate the CAP and BP for any EPOSS Transaction found in the Message Store.

Note that the implementation caches information in memory to save having to repeat the search through Riposte for every EPOSSTransaction message retrieved.

- 1) Find the StockUnit Object associated with the EPOSSTransaction
 - a. Obtain the stock unit name from the EPOSSTransaction (attribute <TxnData.Container:>).
 - b. Retrieve stock unit object using **AgtRiposteGetObject** (collection 'StockUnits' objectname 'extracted from EPOSSTransaction'.
 - c. If this object does not exist, then the stock unit must have since been deleted. So retrieve stock unit object for a deleted stock unit using **AgtRiposteGetObject** (collection 'DelStockUnits' objectname 'extracted from EPOSSTransaction'.
 - d. If the stock unit object contains the attribute <Data.TP:> with a value of 1, the CAP and BP are not needed and this calculation therefore terminates.
- 2) Find the latest RolloverTrailer message.
 - a. Obtain its message identifier from the StockUnit object (attribute <Data.RolloverTrailer:>).
 - b. Retrieve RolloverTrailer message using **AgtRiposteGetMessage**.
- 3) Find the corresponding CAP and BP from this RolloverTrailer message
 - a. Extract attributes <Data.CAP:> and <Data.BP:> from the RolloverTrailer message.

-
- 4) Find the marker that delimits the start of this Balance Period
 - a. Obtain message identifier of Rollover message from RolloverTrailer message (attribute <Data.Rollover:>).
 - b. Retrieve the Rollover message using **AgtRiposteGetMessage**.
 - c. Extract the delimiting marker (held as a string) from the RollOver message (attribute <Data.Mark:>).
 - 5) Compare the Id of the current EPOSSTransaction message with this marker
 - a. Convert the marker string extracted from the Rollover message into a Riposte marker using **RiposteMarkerFromString**.
 - b. Compare message number of the EPOSSTransaction with the entry for the relevant Node in the Riposte marker (obtained in the previous step).
 - 6) If the message number is greater than the corresponding marker entry, then the CAP and BP from the RolloverTrailer (extracted in step 3) are used. We have now finished with this EPOSSTransaction message.
 - 7) Otherwise we need to go back to the previous RolloverTrailer
 - a. Obtain the message identifier of the previous RolloverTrailer message in the chain (attribute <Data.Previous:> of the current RolloverTrailer message). If the current RolloverTrailer message is the first such message in the journal, the attribute <Data.Previous:> will not exist. This situation should not occur in real life while attempting to identify a given transaction's balance and cash account period.

However the code should allow for it and it should also allow for the case of 'message unobtainable' (i.e. archived). In either case an event should be logged and the EPOSSTransaction should be treated as an exception.

- b. Retrieve the previous RolloverTrailer message using **AgtRiposteGetMessage**.
 - c. Having found this message, continue from step 3 above.
- 8) In all the above, where an attribute is described as containing a message identifier, that Attribute has the following structure:

```
<Attribute-name:
  <GroupId: >
  <Id: >
  <Num: >
>
```

Within the implementation of 'Give CAP and BP' we should maintain cached data for each Container (ie StockUnit) that we have encountered. Most Post Offices will have a fairly low number of StockUnits and so a serial search of known StockUnits should be sufficient.

For each Container, we will need to store the Container Name and also to hold details of all the RolloverTrailer messages we have encountered chaining from that container (CAP, BP, message identifier and 'Previous' message identifier) and the marker from the corresponding RollOver message (held as a MARKER structure). Note that there is a special container known as '##'. All messages for this container are part of the Office Rollover process and are ignored during normal harvesting.

This should ensure that StockUnit objects and RolloverTrailer and RollOver messages are not read from Riposte more than once, but also are not read unless they need to be. The cached data should be cleared from store whenever the outlet changes.

2.1.7.2 Non-EPOSS Events or Events occurring in the Default Stock Unit

The mechanism described above does not work if the message being harvested doesn't have a <TxnData.Container:> attribute, or if there is no RolloverTrailer Object for the StockUnit (as is the case for the 'DEF' Container). In these cases we need to make an estimate as to what the CAP should be using the following algorithm.

- 1) Obtain the Start of the Cash Account Year. By default, the date of the Thursday before the last Sunday in March (i.e. the fourth Thursday in March) is used (PC/25140). However, this can be overridden by an entry in the <CAP_NewYear_Dates_Data> section of the .rip file.

Such a section is optional, but if it is provided it would have the following structure:

```
<CAP_NewYear_Dates_Data>
!1996,28-Mar-1996
!1997,27-Mar-1997
!1998,26-Mar-1998
!1999,25-Mar-1999
!2000,23-Mar-2000
!2001,22-Mar-2001
!2002,28-Mar-2002
!2003,27-Mar-2003
```

This algorithm has been confirmed by POCL, so there will never be any need to use the .rip file.

- 2) Subtract this date (i.e. Start of the Cash Account Year) from the current date and divide by 7 to get the current week number. Add 1 so that week numbers start at 1 not 0.

2.1.8 Handling (Stock Unit) Rollover Trailers

Harvesting of (Stock Unit) RolloverTrailer messages was removed in March 2001 (PC/ 64428).

2.1.9 Handling CAP Trailers

The CAPTrailer message identifies the Cash Account Rollover of the entire outlet. A pseudo stock unit is used to represent the entire outlet. This has a somewhat different structure to a Normal Stock Unit.

As defined in [IMP3_STOCK], the EPOSSCap Object will contain a <Data.TP:1> attribute when the entire Office has migrated to operate in Branch Trading Period mode. Once this has happened, messages associated with the CAP Trailer (now really a Trading Period Trailer) are no longer harvested.

When a CAPTrailer message is found, then it is necessary to decide if this is linked into an EPOSSCap object with name 'Office'. If it is, then the related Cash Account messages and Opening Figures messages need to be retrieved and harvested. Note that such messages may be before the "start marker" for the current harvest, but will always be after the Start marker for the CAP associated with the RolloverTrailer pointed to by the CAPTrailer.

The following diagram is copied from [EAG]. Note that it has not been updated to include the Office Declarations – for these, see Figure 3.

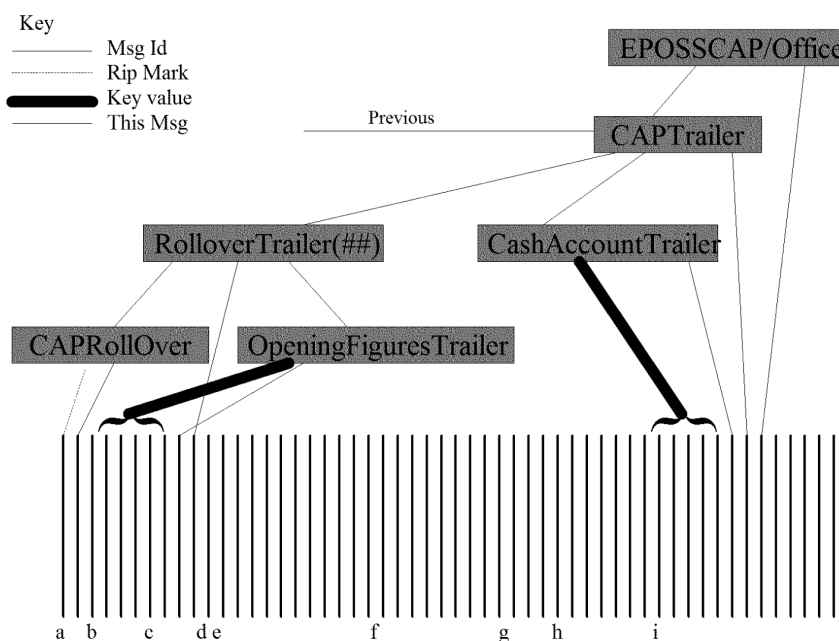


Figure 2 - Office 'Stock Unit' Account Structure

2.1.9.1 Checking a CAPTrailer Message is Valid

The processing is as follows. In the following description “current message” denotes the CAPTrailer message that has been found during the normal scanning of the message store as opposed to any other CAPTrailer message that we find during the navigation process that is described.

- 1) Find the EPOSSCap Object associated with the “current message”
 - a. Retrieve the stock unit object using **AgtRiposteGetObject** (Collection ‘EPOSSCap’ ObjectName ‘Office’).
- 2) Find the latest CAPTrailer message.
 - a. Obtain its message identifier from the EPOSSCap Object (attribute <CAPTrailer:>).
 - b. Retrieve the latest CAPTrailer message using **AgtRiposteGetMessage**.
- 3) Is this the “current message”?
 - a. This is done by comparing the MessageId of the “current message” with that of the CAPTrailer message found by navigation. If it is the same the “current message” is linked in and so we need to harvest the relevant data as described below (section 2.1.9.2).

Since CAP Rollovers normally only take place once per week, then it is not worth putting in checks to see if we’ve gone back too far. It is easier in the unlikely event of an “unlinked CAPTrailer” to just carry on back until the “Previous message” is “unobtainable” (i.e. archived) and give up at that point. NB: in 99.9% of cases the CAPTrailer will be found directly linked to the EPOSSCap Object.

Whenever a CAPTrailer is ignored an Event should be logged.

- 4) Otherwise we need to go back to the previous CAPTrailer

- a. Obtain the message identifier of the previous CAP trailer from the current CAP trailer in the chain (i.e. the one navigated to via the stock unit, not the “current message”), (attribute <Data.Previous:>)
- b. Retrieve the previous CAP trailer message using **AgtRiposteGetMessage** and continue from step 3) above.

5) In all the above, where an attribute is described as containing a MessageId, that Attribute has the following structure:

```
<attribute-name:
  <GroupId: >
  <Id: >
  <Num: >
>
```

2.1.9.2 Harvesting messages associated with a CAPTrailer

Having found that the “current message” is a valid CAPTrailer (i.e. one that is linked to current EPOSSCap Object), there are a number of associated messages that need to be harvested into various tables. These are:

Messages	Significance	Table
Opening Figures messages	Stock Holdings (“accounting” level)	TMS_RX_STOCK_HOLDINGS
Opening Figures Trailer	Stock Holding Total Lines (“accounting” level)	TMS_RX_SH_TOTAL_LINES
Cash Account messages	Cash Account Lines	TMS_RX_CASH_ACCOUNTS
Cash Account Trailer	Cash Account Total Lines	TMS_RX_CA_TOTAL_LINES
Declaration messages	Stock Holdings (“inventory” level)	TMS_RX_STOCK_HOLDINGS
Declaration Trailer	Stock Holding Total Lines (“inventory” level)	TMS_RX_SH_TOTAL_LINES

This is done as follows:

- 1) Find the corresponding (Office) RolloverTrailer message
 - a. Obtain the message identifier of the RolloverTrailer from the “current message” (attribute <Data.RolloverTrailer:>).
 - b. Retrieve the RolloverTrailer trailer using **AgtRiposteGetMessage**.
- 2) Find and harvest the Stock Holding messages at lowest declared level (i.e. at “inventory” level). See 2.1.9.3 for details.
- 3) Find the corresponding CAPRollover message
 - a. Obtain the message identifier of the CAPRollover message from the RolloverTrailer obtained in the previous step (attribute <Data.CAPRollover:>).
 - b. Retrieve the CAPRollover message using **AgtRiposteGetMessage**.
- 4) Find the corresponding OpeningFiguresTrailer message
 - a. Obtain the message identifier of the OpeningFiguresTrailer from the RolloverTrailer obtained in the previous step (attribute <Data.OpeningFigures:>).
 - b. Retrieve the OpeningFiguresTrailer trailer using **AgtRiposteGetMessage**.

-
- 5) Find the corresponding OpeningFigures messages
- Obtain the corresponding Opening Figures Identifier from the OpeningFiguresTrailer (attribute `<Data.OpeningFiguresId:>`).
 - Scan between two markers for messages with the attribute `<EPOSSTransaction.OpeningFiguresId:OpeningFiguresId>` and `<EPOSSTransaction.ProductNo:>` in ranges 1-9999, 11300-11399, and greater than 20000, using **AgtRiposteScanMessage** (*OpeningFiguresId* was obtained in previous step). The low marker is the delimiting marker for the Cash Account Period of the CAPRollover found in step 3. The high marker is the same with the entry for the relevant node set to the message number of the OpeningFiguresTrailer found in step 4.
- 6) Each Opening Figures message is harvested as a “standard” Stock Holdings message, i.e. at “accounting” level (as described in section 2.8.3.12). There may be no messages to harvest within this range; this is not an error.
- 7) The OpeningFiguresTrailer, itself found during the scan, is harvested, for reconciliation purposes, as a “standard” Stock Holding Totals Line, i.e. at “accounting” level (as described in section 2.8.3.18).
- 8) Find the corresponding CashAccountTrailer message
- Obtain the message identifier of the CashAccountTrailer from the “current message” (attribute `<Data.CashAccountId:>`).
 - Retrieve the CashAccountTrailer using **AgtRiposteGetMessage**.
- 9) Find the corresponding Cash Account messages
- Obtain the corresponding Cash Account Identifier from the CashAccountTrailer (attribute `<Data.CashAccountId:>`).
 - Scan between two markers for messages with the attribute `<CashAccountId:CashAccountId>` using **AgtRiposteScanMessage** (*CashAccountId* was obtained in previous step). The low marker is the delimiting marker for the Cash Account period of the CAPRollover found step 3. The high marker is the same with the entry for the relevant node set to the message number of the CashAccountTrailer found in step 8.
- 10) Each Cash Account message is harvested as described in section 2.8.3.3. There may be no messages to harvest within this range; this is not an error.
- 11) The CashAccountTrailer, itself found during the scan, is harvested, for reconciliation purposes, as a Cash Account Totals Line (as described in section 2.8.3.17).

2.1.9.3 Harvesting Stock Holdings at lowest declared level

The “standard” Stock and Cash messages harvested as described in the previous section are at “accounting” level (e.g. stamps are aggregated and not broken down by denomination). CP/2096 requires that Stock and Cash are also harvested at lowest declared level (i.e. “inventory” level, whereby, for example, stamps are broken down by denomination).

Note that any errors detected during this phase are logged as Events, but harvesting of the group continues regardless.

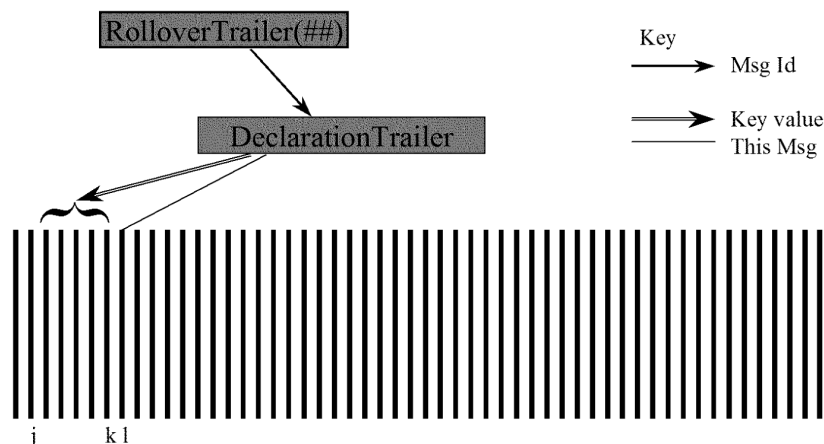


Figure 3 - Office Declaration Structure

- 1) Find the corresponding (Office) DeclarationTrailer message
 - a. Obtain the message identifier of the DeclarationTrailer from the RolloverTrailer (attribute <Data.DeclarationTrailer:>).
 - b. Retrieve the DeclarationTrailer trailer using **AgtRiposteGetMessage**.
- 2) Establish the lower bound for the corresponding Declaration messages
 - a. Obtain the corresponding Declaration Identifier from the DeclarationTrailer (attribute <Data.DeclareId:>).
 - b. Obtain the message identifier of the lower bound message by parsing the Declaration Identifier. It is not a “pointer”, but can be parsed as *group_id_num*.
- 3) Find the corresponding Declaration messages
 - a. Scan between two markers for messages with the attribute <EPOSSTransaction.OpeningFiguresId:DeclareId> and <EPOSSTransaction.ProductNo:> in ranges 1-9999, 11300-11399, and greater than 20000, using **AgtRiposteScanMessage** (DeclareId was obtained in previous step). The low marker is the lower bound found in the previous step. The high marker is the DeclarationTrailer found in step 1. Only the one node is scanned.

Note that the attributes containing the Declaration Identifier key have a different name in the Declaration messages from that in the DeclarationTrailer. This is unlike the standard practice for keying similar sets of messages delimited by a trailer.

- 4) Each Declaration message is harvested as a “lowest declared level” Stock Holdings message, i.e. at “inventory” level (as described in section 2.8.3.12). There may be no messages to harvest within this range; this is not an error.
- 5) The DeclarationTrailer, itself found during the scan, is harvested, for reconciliation purposes, as a “standard” Stock Holding Totals Line, i.e. at “accounting” level (as described in section 2.8.3.18). The DeclarationTrailer is harvested only if <Data.MessageCount:> exists and is non-zero.

2.1.10 Harvester exceptions

Whenever the agent is unable to write the data to be harvested as a row in one of the database tables, it not only records this as an error in the NT Event Log, but it also captures the details it just failed to write in the Harvester Exceptions Table, TMS_HARVESTER_EXCEPTIONS. No further error action is required by the agent, so the harvesting of the group for the trading day concerned is still considered to have been successful.

This generic behaviour is described in [GEN], and the table's format is given in [GENTABS].

The harvester exceptions may then be 'repaired' using a TPS Host repair tool.

2.2 Control Interfaces

This agent runs as a command line task. It is invoked by the following command line:

T_HV_ALL.exe [-i *instance_id*] [-g *guise*] [-s *stop_time*] [-b *business_day*]

where

Parameter	Meaning
-i <i>instance_id</i>	An identifier, up to 3 characters long, with a default of “001”, set up in the Maestro schedules to ensure uniqueness of all instances of this agent.
-g <i>guise</i>	An identifier used as the final part of the registry key hierarchy. It defaults to the value used for the <i>instance_id</i> .
-s <i>stop_time</i>	Stop date and time, in the format “dd-Mon-yyyy hh:mi” where the space between date and time is optional, and the time uses the 24-hour clock. If omitted, there is no stop time.
-b <i>business_day</i>	A date, in the format “dd-Mon-yyyy”, which controls the termination of harvesting each Post Office. It defaults to ‘today’, as given by the system date.

2.3 Maestro Schedule

Use sample 3 with a *chunk_type* of ‘p’, as described in [GENTABS] and [MAESTRO], as a basis.

The following items are as described in [GENTABS]:

- Agent Work Table **TMS_AWT_TPS**
- AWT Sequence **TMS_SEQ_AWT_TPS**
- Generic Schedule Sequence **TMS_SCHEDULE_SEQ**

2.4 Registry Entries Used

The registry key for this agent is **T_HV_ALL**.

As stated in 2.1.1, each instance of the TPS Bulk Harvester is configured with different registry values for USERNAME and PASSWORD. These value names are held at a lower level, under the keys **T_HV_ALL\guise**, (in practice, under **T_HV_ALL\1** to **T_HV_ALL\64**).

Details of registry value names are contained in [GEN]. The following registry entries are of particular interest.

Value Name	Value	Comment
DBLOCATION	Explicitly configured	Location of the Oracle database (the Host string used during connection). Specified by PIT, typically “TPS”
FIELD_CODES	“C:\Program Files\PathwayAgents\tps_field_names.rip” [Explicitly configured]	Pathname of RIP file.
PASSWORD	Explicitly configured	Oracle password. Specified by PIT.

PERFMONFILE	"C:\AgentStats\T_HV_ALL_Perf.mon" [Explicitly configured]	Name of memory mapped file.
RIPOSTE	Explicitly configured	The local Locale. Specified by PIT, typically "BOOTLE" or "WIGAN".
TOTALCONNECTI ONTIMEOUT	1200000 (20 mins) [Explicitly configured]	In milliseconds.
USERNAME	Explicitly configured	Oracle user name. Specified by PIT, typically "TPS_AGENT_guise".

The following registry values names are also of relevance to this agent:
CHUNK_DELAY_PERIOD, CHUNK_POLL_PERIOD, CONNECTPOLLTIME, DBSCANTIME,
FAILBIAS, INITIALFAILSTATUS, REPLICATIONDELAY, REPORTFAILDELAY,
REPORTFAILPERIOD, TRACECALLS, TRACELEVEL, WRONGBIASDELAY.

2.5 Exception Handling through Oracle

Harvester exceptions will be logged to the Harvester Exceptions table (TMS_HARVESTER_EXCEPTIONS) using generic harvester code (see [GEN] and [GENTABS]).

2.6 Oracle Table Format

2.6.1 TPS-Specific Database Tables

All the TPS-specific tables are defined in [TPSTABS]. In summary, they are as follows:

1) TPS Transaction Tables

This has details of all the Data tables into which data is to be harvested. These tables are:

- a. TMS_RX_APS_TRANSACTIONS
- b. TMS_RX_BDC_TRANSACTIONS
- c. TMS_RX_CASH_ACCOUNTS
- d. TMS_RX_EFT_TRANSACTIONS
- e. TMS_RX_EPOSS_EVENTS
- f. TMS_RX_EPOSS_TRANSACTIONS
- g. TMS_RX_NWB_TRANSACTIONS
- h. TMS_RX_OBCS_STATUSES
- i. TMS_RX_OBCS_TRANSACTIONS
- j. TMS_RX_STOCK_HOLDINGS
- k. TMS_RX_MIGRATION_OBJECT

2) TPS Reconciliation and Chart of Accounts Tables

This has details of the following Reconciliation tables into which reconciliation data is to be harvested. These tables are:

- a. TMS_RX_TRANSACTION_TOTALS
- b. TMS_RX_COUNTER_TRAN_ERRORS
- c. TMS_RX_CA_CT_EXCEPTIONS
- d. TMS_RX_CA_TOTAL_LINES
- e. TMS_RX_SH_TOTAL_LINES
- f. TMS_RX_COFA_TRANSACTIONS
- g. TMS_RX_COFA_SUMMARIES

3) TPS Control Tables

- This has details of the following Control tables used during harvesting. These tables are:
- a. TPS_OUTLETS (input only – see 2.1.2)
 - b. TPS_OUTLET_EODS (output only)

TPS_OUTLET_EODS is an output-only table.

2.6.2 Generic Database Tables

See [GENTABS] for the usage and format of the following generic database tables and sequences:

- a) Agent Marker Table **TMS_AMT_TPS**
- b) Agent Work Table **TMS_AWT_TPS**
- c) AWT Sequence **TMS_SEQ_AWT_TPS**
- d) Harvester Exceptions Table **TMS_HARVESTER_EXCEPTIONS**

2.7 Riposte Message Format

The Riposte messages that are to be harvested are described in separate documents as follows:

- 1) [EPOSSBAL] defines the following messages:
 - a. EPOSSCAP collection for Office
 - b. CAP Trailer message
 - c. Rollover Trailer message
 - d. Opening Figures Trailer message
 - e. Opening Figures message (aka Opening Balance Transaction message or EPOSS Stock Holding message)
 - f. StockUnits and DelStockUnits collections
- 2) [EPOSSMIG] defines the following messages:
 - a. Outlet Migration Complete Marker object

As that document contains an error, the corrected attribute grammar is reproduced here for convenience (PC/55745).

```
<Collection:MiMAN>
<ObjectName:##>
<Data:
  <MigStatus:C>           // present if and only if migration complete
>
```

- 3) [EPOSSEOD] defines the following messages:
 - a. Daily Reconciliation Transaction Totals message
 - b. Daily Reconciliation Cash Account Counter Error message
 - c. Daily Reconciliation Trailer message
 - d. Weekly Reconciliation Cash Account Counter Error message
 - e. Weekly Reconciliation Trailer message
- 4) [TPSCSR+] defines the following messages:
 - a. Office Declaration messages for cash, stamps and stock (aka Statements per Outlet)
 - b. Office Declaration Trailer message

The three classes of Office Declaration message have <EPOSSTransaction.TranType:> set to SUCashStatement, SUSTampStatement and SUSTockStatement respectively. The TPS Harvester, however, does not distinguish between them.

- 5) [EAG] is an historic document that defines the following messages:
 - a. Standard EPOSS Transaction line message
 - b. APS EPOSS Transaction line message
 - c. OBCS EPOSS Transaction line message
 - d. EPOSS Counter Event message
 - e. Various Event messages
 - f. EPOSS Cash Account message
- 6) [EODGEN] defines the following messages:
 - a. (Public) EOD Marker message
- 7) [EODHT] defines the following messages:
 - a. EOD Harvest Trailer message
- 8) [OBCS] defines the following messages:
 - a. OBCS Order Book status messages (i.e. received, issued, impounded)
- 9) [NBSFLOWS] defines the following messages:
 - a. NBS, ETS & DCS EPOSS Transaction [C1] messages
- 10) [RDBDC] is expected to contain the definitions of the following messages:
 - a. Bureau de Change EPOSS Transaction messages
- 11) [E2E1HLD] defines the following messages:
 - a. Chart of Accounts Summary Trailer messages
 - b. Chart of Accounts Summary messages
 - c. Chart of Accounts Transaction messages

2.8 Mapping of Riposte Messages to Output Tables

2.8.1 The Output Tables

The following tables identify which messages are harvested into which tables.

The mappings are defined in the Excel spreadsheet in [TPSTABS].

Section 2.8.2 provides some general notes that apply to many or all of the interface tables (identified in the spreadsheet with a General Note Number). Section 2.8.3 provides specific notes that apply to a particular table (identified in the spreadsheet with a Specific Note Number).

- 1) TPS Transaction Tables

The Data tables into which data is to be harvested are:

Table	Messages harvested
-------	--------------------

TMS_RX_APS_TRANSACTIONS	APS EPOSS Transactions
TMS_RX_BDC_TRANSACTIONS	Bureau de Change Transactions
TMS_RX_CASH_ACCOUNTS	Cash Account messages from a CAP Rollover
TMS_RX_EFT_TRANSACTIONS	DCS Transactions (for EFTPoS)
TMS_RX_EPOSS_EVENTS	EPOSS and other events
TMS_RX_EPOSS_TRANSACTIONS	EPOSS Transactions other than APS and OBCS
TMS_RX_NWB_TRANSACTIONS	Network Banking (NBS) Transactions and Electronic Top-Up (ETS) Transactions
TMS_RX_OBCS_STATUSES	OBCS Order Book status messages
TMS_RX_OBCS_TRANSACTIONS	OBCS EPOSS Transactions
TMS_RX_STOCK_HOLDINGS	Stock Holding messages from a CAP Rollover, at both accounting and inventory levels
TMS_RX_MIGRATION_OBJECT	Outlet Migration Complete Marker object
TMS_RX_COFA_SUMMARIES	Chart of Accounts Summary messages
TMS_RX_COFA_TRANSACTIONS	Chart of Accounts Transaction messages

2) TPS Reconciliation Tables

The Reconciliation tables into which reconciliation data is to be harvested are:

Table	Messages harvested
TMS_RX_TRANSACTION_TOTALS	Daily Reconciliation Transaction totals (DailyTxnCT)
TMS_RX_COUNTER_TRAN_ERRORS	Daily Reconciliation Cash Account counter errors (DailyCAErr)
TMS_RX_CA_CT_EXCEPTIONS	Weekly Reconciliation Cash Account counter errors (WeeklyCAErr)
TMS_RX_CA_TOTAL_LINES	Cash Account Total Lines from CAP Rollover
TMS_RX_SH_TOTAL_LINES	Stock Holding Total Lines from CAP Rollover, at both accounting and inventory levels

3) TPS Control Tables

The Control tables into which control data is to be harvested are:

Table	Messages harvested
TPS_OUTLET_EODS	Latest EOD

2.8.2 General Notes

- 1) `org_unit_id` is obtained by looking up `<GroupId>` in the `TPS_OUTLETS` table as described in 2.1.2.

- 2) `org_unit_version_no` is obtained from the `TPS_OUTLETS` table as described in 2.1.2.
- 3) `balance_period` is calculated as described in 2.1.7
- 4) `cash_account_period` is calculated as described in 2.1.7
- 5) `trading_date` is the value of the `<EODMark.EODDate:>` attribute of the EOD Marker that we are currently harvesting up to.
- 6) The Oracle DATE fields are formed by concatenating the appropriate Riposte Date and Time fields. Note that all Dates and Times are Universal Time (not Local Time).
- 7) *No longer referenced.*
- 8) `fallback_mode` is derived from the `<TxnData.Mode:>` attribute as follows:

Details for this are found in section `<FallbackMode_Data>` of the .rip file.

<code><TxnData.Mode:></code>	<code>fallback_mode</code>
REC	Y
any other value	N

- 9) *No longer referenced. Replaced by Specific Notes.*
- 10) `transaction_mode_id` is derived from the `<EPOSSTransaction.CrossReference.Omode:>` attribute if it exists and is non-empty. Otherwise it is derived from the `<TxnData.Mode:>` attribute. (PC/37312)

Note that the `<EPOSSTransaction.CrossReference.Omode:>` is populated only if `<TxnData.Mode:>` is "ER".

Details of the following lookup are found in section `<TransactionModeId_Data>` of the .rip file. Note that there is no longer any Default in this section, so a null mode generally causes an exception (PC/33593); however, for APS transactions, and for EPOSS transactions relating to Mails (identified by either `<Application:Mails>` or `<ItemName:Mails>`), a null mode is defaulted to "SC", mapping to "1" (PC/109219 & PC/114905).

Omode or Mode	transaction_mode_id	Comment
SC	1	Serve Customer
ER	1	Existing Reversal
RV	1	New Reversal
RISD	2	Remit In - Supplies Division
ROSD	3	Remit Out - Supplies Division
RU	4	Revaluation - Uprating
RD	5	Revaluation - Downrating
TI	7	Transfer In - Between Stock Units

RIOP	8	Remit In - Other Offices
ROOP	9	Remit Out - Other Offices
RICL	10	Remit In - Client
ROCL	11	Remit Out - Client
RODC	12	Remit Out - DPC
TO	13	Transfer Out - Between Stock Units
REC	14	Bulk Input
HK	15	Housekeeping
SAP	16	Stock Adjustment - Positive
DDP	17	Declaration Discrepancy - Positive
SAN	18	Stock Adjustment - Negative
DDN	19	Declaration Discrepancy - Negative
PT	22	Parcel Traffic
NAD	23	Non Accounting Data
RIAD	24	Remit In Value Stock from ADC
ROAD	25	Remit Out to ADC
ROSP	26	Remit Out Suspense Pouch
RISP	27	Remit In Suspense Pouch
RODP	28	Remit Out Dispatch Pouch
MG	29	Make Good
HD	30	Plead Hardship
EV	31	Request Evidence
WO	32	Write Off
AN	33	Assign Nominee
SW	34	"Stock" Write Off

11) <TxnData.SessionId:> has the form:

- a. *gggggg-nn-mmmmmmmmm* (obsolescent), or
- b. *ss-gggggg-nn-mmmmmmmmm-uu*

where

ss is 44, to identify Horizon
gggggg is the GroupId (up to 6 digits)
nn is the NodeId (up to 2 digits)
mmmmmmmm is the Num of the last message committed to the message store before the start of the session
uu is a uniqueness factor, starting at 1, but could be more than 1 digit.

session_sequence_number is calculated as follows:

$$((mmmmmmmm * 10) + uu) \text{ modulo } 1,000,000$$

12) `transaction_sequence_number` is generated such that it is different for each transaction within a session, and increasing. It is calculated as follows:

$$(\text{Num} - \text{mmmmmmmm}) + 10,000,000,001 \text{ modulo } 10,000$$

where

`Num` is the value of `<Num:>`

`mmmmmmmm` is from `<TxnData.SessionId:>`

The addition of a large number is to cater for MiECCO transactions where the `mmmmmmmm` component in `<TxnData.SessionId:>` bears no relationship to the message's `Num` (PC/30329).

There is a remote chance of non-uniqueness should a single session generate more than 10,000 messages.

Note that `transaction_sequence_number` is not calculated from `<TxnData.TxnId:>`

13) *No longer referenced.*

14) *No longer referenced.*

15) *No longer referenced.*

16) These are indicator fields, which in Riposte will be signified by the attribute being present with a value of 1 as TRUE and any other value or absent as FALSE. In Oracle, TRUE maps onto 'Y' and FALSE maps onto 'N'.

17) These are "reverse" indicator fields, which in Riposte will be signified by the attribute being present with a value of 1 as TRUE and any other value or absent as FALSE. In Oracle, TRUE maps onto 'N' and FALSE maps onto 'Y'.

18) *No longer referenced.*

19) *No longer referenced.*

20) These fields are all ignored by the harvester and so no values will ever be inserted into them.

21) A repair is attempted for those EPOSS Transactions for which the `<TxnData.Start:>` attribute is empty or absent (see section 2.1.2).

22) `transaction_mode_version_no` is taken from the `<EPOSSTransaction.CrossReference.OmodeV:>` attribute if the `<EPOSSTransaction.CrossReference.Omode:>` exists and is non-empty. Otherwise it is taken from the `<EPOSSTransaction.BlackBoxData.V:>` attribute. (PC/37312)

Note that the `<EPOSSTransaction.CrossReference.Omode:>` is populated only if `<TxnData.Mode:>` is "ER".

23) `user_name` is taken from `<EPOSSTransaction.AdditionalData.POClerkID:>` if it exists, otherwise from `<User:>`.

This rule was originally invented for the BES Help Desk, but has been retained for specialist use by third line support.

2.8.3 Specific Notes

2.8.3.1 TMS_RX_APS_TRANSACTIONS

Details for this are found in section <APS_EPOSS_Data> of the .rip file.

- 1) reversal is derived from <EPOSSTransaction.BlackBoxData.TransactionMode:> (PC/47285 & PC/51205):

<EPOSSTransaction.BlackBoxData.TransactionMode:>	reversal	Comment
2	1 (Existing reversal)	
0	0 (Normal)	
null	null	Will cause an exception
any other value	"Unexpected Mode <i>n</i> "	Will cause an exception

- 2) transaction_mode_id is always 1, but the TPS Harvester does not police this

2.8.3.2 TMS_RX_BDC_TRANSACTIONS

Details for this are found in section <BDC_EPOSS_Data> of the .rip file.

- 1) reversal is derived as for the TMS_RX_EPOSS_TRANSACTIONS table, using the mapping that is dependent on <EPOSSTransaction.CrossReference.Num:>
- 2) transaction_sequence_number is derived as for the TMS_RX_EPOSS_TRANSACTIONS table.

2.8.3.3 TMS_RX_CASH_ACCOUNTS

Details for this are found in section <CASH_ACCOUNT_Data> of the .rip file.

- 1) amount is set to the <EPOSSTransaction.SaleValue:> attribute if it is not NULL, otherwise it is set to the <EPOSSTransaction.Qty:> attribute, with a default of 0 if both are NULL.
- 2) The <EPOSSTransaction.LineNo:> attribute is of the form 'nnnnxxxx', where either 'nnnn' or 'xxxx' are '9999'. The line_number column will take whichever of 'nnnn' or 'xxxx' which is not '9999'. (If both are '9999', then '9999' should be supplied. If neither are '9999', then either can be supplied. However, I don't believe that either of these cases should ever happen.)

2.8.3.4 TMS_RX_COFA_SUMMARIES

Details for this are found in section <COFA_SUMMARY_TRAILER_Data> of the .rip file.

2.8.3.5 TMS_RX_COFA_TRANSACTIONS

Details for this are found in section <COFA_TRANSACTIONS_Data> of the .rip file.

2.8.3.6 TMS_RX_EFT_TRANSACTIONS

Details for this are found in section <EFT_EPOSS_Data> of the .rip file.

- 1) reversal is derived from the <TxnData.Mode:> attribute as follows (PC/90174, PC/91231):

<TxnData.Mode:>	reversal
ER	2 (New reversal)
RV	2 (New reversal)
any other value	0 (Normal)

This mapping specifically takes account of the fact that DCS EPOSS Transactions are always settlement transactions, for which reversals must always generate a "2"

- 2) transaction_sequence_number is derived as for the TMS_RX_EPOSS_TRANSACTIONS table.

2.8.3.7 TMS_RX_EPOSS_EVENTS

Details for this are found in the following sections of the .rip file:

Input Message	.rip file section
EPOSS Event	<EPOSS_EVENT_Data>
InactivityLogout	<FORCED_LO_IL_Data>
AdminLogout	<FORCED_LO_AL_Data>
LogoutAuthority: Desktop	<FORCED_LOGOUT_Data>
LogoutAuthority: other	<MANAGER_FORCED_LOGOUT_Data>
FailedLogon	<FAILED_LOGON_Data>
Logon	<LOGON_Data>
Logout	<LOGOUT_Data>
EOD Marker	<EOD_EVENT_Data>

- 1) See 2.1.6.

2.8.3.8 TMS_RX_EPOSS_TRANSACTIONS

Details for this are found in section <EPOSS_Data> of the .rip file.

- 1) If <Application:> is not EPOSSAppMain, reversal is derived from the <TxnData.Mode:> attribute as follows:

Details for this are found in section <Reversal_Data> of the .rip file.

<TxnData.Mode:>	reversal
ER	1 (Existing reversal)
RV	2 (New reversal)
any other value	0 (Normal)

If <Application:> is EPOSSAppMain, reversal is further dependent on <EPOSSTransaction.CrossReference.Num:> as follows (PC/27578):

<TxnData.Mode:>	<EPOSSTransaction.CrossReference.Num:>	reversal
ER	not null	1 (Existing reversal)
ER	null	2 (New reversal)
RV		2 (New reversal)
any other value		0 (Normal)

2) transaction_sequence_number is harvested from <EPOSSTransaction.AdditionalData.Num:> if it is not null, otherwise it is generated as per General Note 12 above.

2.8.3.9 TMS_RX_NWB_TRANSACTIONS

Details for this are found in section <NWB_EPOSS_Data> of the .rip file. (Note that this table is used for both NBS and ETS EPOSS Transactions.)

1) reversal is derived as for the TMS_RX_EPOSS_TRANSACTIONS table, using the mapping that is dependent on <EPOSSTransaction.CrossReference.Num:>

Note that this mapping could have been simplified, as NBS transactions cannot be reversed and ETS transactions always have a Mode of 'RV'.

2) transaction_sequence_number is derived as for the TMS_RX_EPOSS_TRANSACTIONS table.

2.8.3.10 TMS_RX_OBCS_STATUSES

Details for this are found in section <OBCS_STATUS_Data> of the .rip file.

1) reversal is derived from the <TxnData.Mode:> attribute as follows:

Details for this are found in section <Reversal_Data> of the .rip file.

<TxnData.Mode:>	reversal
ER	1 (Existing reversal)
RV	2 (New reversal)

any other value	0 (Normal)
-----------------	------------

2) The date component of `end_date` is harvested from `<Data.EndDate:>` if it is not null, otherwise it is harvested from `<Date:>`. Similarly, the date component of `start_date` is harvested from `<Data.StartDate:>` if it is not null, otherwise it is harvested from `<Date:>`. (PC/64962)

2.8.3.11 TMS_RX_OBCS_TRANSACTIONS

Details for this are found in section `<OBCS_EPOSS_Data>` of the .rip file.

2.8.3.12 TMS_RX_STOCK_HOLDINGS

Details for this are found in the following sections of the .rip file:

Input Message	.rip file section
Opening Figures message	<code><STOCK_HOLDING_Data></code>
Office Declaration message	<code><DECLARATION_SH_Data></code>

- 1) BP cannot be calculated in the normal way for Stock Unit ##, so it is hard-coded to 1.
- 2) CAP cannot be calculated in the normal way for Stock Unit ##, so it is derived as described in 2.1.7.2
- 3) `stock_type` indicates the nature of the Stock Holding entry as follows:

Nature	Associated with	stock_type
Accounting level	Opening Figures message	A
Inventory (i.e. lowest-declared) level	Office Declaration message	L

2.8.3.13 TMS_RX_MIGRATION_OBJECT

See 2.1.3.

2.8.3.14 TMS_RX_TRANSACTION_TOTALS

Details for this are found in section `<REC_DAILY_TXNS_Data>` of the .rip file.

2.8.3.15 TMS_RX_COUNTER_TRAN_ERRORS

Details for this are found in section `<REC_TXN_ERRS_Data>` of the .rip file.

2.8.3.16 TMS_RX_CA_CT_EXCEPTIONS

Details for this are found in section `<REC_CA_ERRS_Data>` of the .rip file.

2.8.3.17 TMS_RX_CA_TOTAL_LINES

Details for this are found in section `<REC_WEEKLY_CA_Data>` of the .rip file.

"Weekly" refers to the fact that the CAP Rollover for an Office typically occurs once a week every week.

- 1) `total_abs_ca_line_amounts` is a total of both Quantities and Amounts as only Quantity OR Amount appears on a Cash Account Line and TPS Host cannot distinguish.

2.1.1.18 TMS_RX_SH_TOTAL_LINES

Details for this are found in the following sections of the .rip file:

Input Message	.rip file section
OpeningFiguresTrailer	<REC_WEEKLY_SH_Data>
OfficeDeclarationTrailer	<REC_DECLARE_SH_Data>

- 1) `stock_type` indicates the nature of the Stock Holding entry as follows:

Nature	Associated with	stock_type
Accounting level	Opening Figures Trailer	A
Inventory (i.e. lowest-declared) level	Office Declaration Trailer	L

2.1.1.19 TPS_OUTLET_EODS

One row in this table is created/updated per outlet, to record the latest EOD Date harvested.

- 1) These fields were added for handling the migration phase for TPS Reconciliation, and so are now obsolete.
- 2) `eposs_weekly_recon_ignore` was introduced to indicate whether the very first EPOSS Weekly Reconciliation necessarily contained errors and hence should be ignored. The attribute to be harvested was never, in practice, generated.

3. TPS TRANSACTION CORRECTIONS BULK LOADER (T_LD_TC)

The **TPS Transaction Corrections Bulk Loader** is described using the following structure:

- 1) Functional Description
- 2) Control Interfaces
- 3) Maestro Schedule
- 4) Registry Entries Used
- 5) Exception Handling through Oracle
- 6) Oracle Table Format
- 7) Riposte Message Format
- 8) Mapping of Oracle Tables to Riposte Messages

3.1 Functional Description

The TPS Transaction Corrections Bulk Loader allows the TPS Host to pass a number of transaction corrections to specified Branches. It is expected to be run once per day, during the evening or night.

The Loader fetches rows from the TMS_TX_TPS_TC_DETAILS table. For each row it creates a corresponding Riposte message in the group identified by `group_id`. The message is given a long expiry period controlled by a registry parameter, defaulted to 42 days. The `actioned_ind` flag is set to <null> to indicate success.

If the group is not known to LUC, the record is written to the Loader Exceptions table TMS_EXCPTNS. The `actioned_ind` flag is set to 'F' to indicate failure.

3.2 Control Interfaces

This agent runs as a command line task. It is invoked by the following command line:

T_LD_TC.exe [-i *instance_id*] [-g *guise*] [-s *stop_time*]

where

Parameter	Meaning
-i <i>instance_id</i>	An identifier, up to 3 characters long, with a default of "001", set up in the Maestro schedules to ensure uniqueness of all instances of this agent.
-g <i>guise</i>	An identifier used as the final part of the registry key hierarchy. It defaults to the value used for the <i>instance_id</i> .
-s <i>stop_time</i>	Stop date and time, in the format "dd-Mon-yyyy hh:mi" where the space between date and time is optional, and the time uses the 24-hour clock. If omitted, there is no stop time.

3.3 Maestro Schedule

This bulk loader agent should be scheduled to run once overnight following the host processing that populates the interface table. The agent schedule may be run even if the interface table is empty.

Use sample 1 with a *chunk_type* of 's' (i.e. a single work chunk), as described in [GENTABS] and [MAESTRO], as a basis.

The following items are as described in [GENTABS]:

- Agent Work Table **TMS_AWT_TPS_TC_DETAIL**
- AWT Sequence **TMS_SEQ_AWT_TPS_TC_DETAIL**
- Generic Schedule Sequence **TMS_SCHEDULE_SEQ**

3.4 Registry Entries Used

The registry key for this agent is **T_LD_TC**.

Details of registry value names are contained in [GEN]. The following registry entries are of particular interest.

Value Name	Value	Comment
DBLOCATION	Explicitly configured	Location of the Oracle database (the Host string used during connection). Specified by PIT, typically "TPS"
PASSWORD	Explicitly configured	Oracle password. Specified by PIT.
PERFMONFILE	"C:\AgentStats\T_LD_TC_Perf.mon" [Explicitly configured]	Name of memory mapped file.
RIPOSTE	Explicitly configured	The local Locale. Specified by PIT, typically "BOOTLE" or "WIGAN".
TOTALCONNECTI ONTIMEOUT	1200000 (20 mins) [Explicitly configured]	In milliseconds.
USERNAME	Explicitly configured	Oracle user name. Specified by PIT.
MSG_EXPIRY	42 [Defaulted]	Expiry period to be given to Riposte messages created.

The following registry values names are also of relevance to this agent:

CHUNK_DELAY_PERIOD, CHUNK_POLL_PERIOD, CONNECTPOLLTIME, DBSCANTIME, REPORTFAILDELAY, REPORTFAILPERIOD, TRACECALLS, TRACELEVEL.

3.5 Exception Handling through Oracle

Exceptions will be logged to the exceptions table (TMS_EXCPTNS) using generic loader code (see [GEN] and [GENTABS]).

3.6 Oracle Table Format

3.6.1 TMS_TX_TPS_TC_DETAIL

This table contains a row for every valid transaction correction received in the current day. Its formal definition is in [HRSAP_HLD]. Its estimated size is 1200 rows.

3.6.2 Generic Database Tables

See [GENTABS] for the usage and format of the following generic database tables and sequences:

- Agent Work Table **TMS_AWT_TPS_TC_DETAIL**
- AWT Sequence **TMS_SEQ_AWT_TPS_TC_DETAIL**
- Exceptions Table **TMS_EXCPTNS**

3.7 Riposte Message Format

The message format is formally defined here.

```
<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>
    <3:ModeOption3>
  >
  <Text:MessageText>
  <ClientRef:ClientReference>
>
```

3.8 Mapping of Oracle Tables to Riposte Messages

The mapping is given in the table below. All fields are regarded as optional. No validation is performed on their values.

Riposte attribute	Oracle column	Comment
<Ref:>	sap_reference_id	The transaction correction reference number
<Iter:>	iteration_flag	Contains a value of 'N' (New) or 'E' (Evidence Provided)

<Article:>	article	Horizon Product Id of the product that will be transacted as a result of processing the Transaction Correction
<Instruction:>	instruction	The product to which the Transaction Correction will be settled
<AccountingSense:>	accounting_sense	Containing the value 'TCINV' if the Article is sold or 'TCCRM' if the article is returned
<Value:>	value	The Value to be transacted as part of the Correction. If Value is non-zero, then Quantity will be zero
<Qty:>	quantity	The Quantity to be transacted as part of the Correction. If Quantity is non-zero, then Value is irrelevant (not to be used)
<AllowedModesId:>	allowed_modes	The Mode Id that is passed from the POL FS System and from which the ModeOptions are derived within the TPS System
<1:>	mode_1	The possible modes in which the Correction may be transacted Note: mode_2 and mode_3 may be NULL, in which case the corresponding Riposte attributes will be empty
<2:>	mode_2	
<3:>	mode_3	
<Text:>	message	Up to 500 characters of message text. Carriage returns are depicted by the presence of the vertical bar character ' '
<ClientRef:>	client_reference_id	A client reference number

APPENDIX A. OUTLET BALANCING CONCEPTS

This Appendix is included to assist the reader in understanding the relationships between some of the 'balancing' operations carried out by the staff at a Post Office and the various Riposte messages produced on the counters. *It should only be considered to be a rough guide.* The reader should refer to other documents, such as [EPOSSBAL], for a more complete understanding.

The cash account year is divided into **cash account periods**. Each cash account period (**CAP**) lasts a week, generally from Thursday to the following Wednesday. The CAP is identified as a week number from 1 to 52 (or 53) and the start of the year is the Thursday before the last Sunday in March. Exceptionally, a CAP may last for two or even three weeks, in which case the CAP number jumps accordingly.

A **stock unit** (or **container**) corresponds to a counter drawer. Conceptually a stock unit can be moved from one counter to another. It may be assigned to an individual or it may be shared. If it is shared then it may be for the whole Post Office or just a subset. When a user (or clerk) logs on they do so as an individual using a particular stock unit.

Each stock unit has to be periodically **balanced**. Before a stock unit can be balanced, certain **declarations** of cash and stock have to be performed by the user (and any discrepancies confirmed). If the user is satisfied with the trial balance, **the stock unit may be rolled over**. The user must elect whether to roll over the stock unit into the next CAP, or into the next **balance period (BP)** within the same CAP. Each balance period is identified by a value from 1 to 99 within the CAP. There are no longer any messages of interest associated with a **stock unit rollover**.

Once all the stock units have been balanced and rolled over into the next CAP, then the office as a whole is balanced and rolled over into the next CAP. (A stock unit of '##' refers to the whole office.) The messages of interest for an **office rollover** include (see 2.1.9.2):

- a) **Opening Figures messages**, which give the stock holdings at 'accounting' level for the new CAP,
- b) **Declarations at lowest declared level**, which give stock holdings at 'inventory' level, and
- c) **Cash Account messages**, which contain accounting data posted for the cash account.

Both the Opening Figures messages and Declaration messages are simply copied from those for each stock unit rollover and are not aggregated across the stock units. At accounting level, cash is a single product as are (denominated) stamps. At inventory level, cash is broken down into 1p coins and £5 notes, etc, and stamps are broken down into 1p stamps and £5 stamps, etc.

The user may abandon the stock unit rollover at any time before finally committing it, e.g. because he is not happy with the trial balance. Some of the messages associated with the rollover, including perhaps the stock unit **RolloverTrailer message** itself, may have been written to the message store before the rollover was abandoned. A RolloverTrailer message is 'valid' (i.e. corresponds to a committed rollover) only if it is properly linked into the chain of RolloverTrailer messages for that stock unit. The head of the chain is pointed to by a persistent object – this structure is shown in Figure 1.

The office rollover may likewise be abandoned before finally committing it. A **CAPTrailer message** has to be 'validated' in a similar manner before it is harvested (see 2.1.9.1). The data structure is shown in Figure 2.

An EPOSS transaction is performed within a stock unit. The EPOSSTransaction message records the stock unit (in the <Container:> attribute), but not the stock unit's CAP or BP current at the time of the transaction. However, the data to be harvested for an EPOSSTransaction includes the CAP and

**Fujitsu
Services**

TPS Agents for BI3: High Level Design

Ref: AD/DES/041

COMPANY IN-CONFIDENCE

Version: 7.0

Date: 09/08/05

BP, which the TPS Harvester, therefore, has to determine. It does this by following back down the chain of RolloverTrailer messages for the stock unit until it finds the one that occurs **before** the EPOSTransaction, and that one gives the required information – this is described in 2.1.7.1.