FUJ00171949

Document Title: **High Level Design of Common Agents**

Document Type: Design Specification

Release: BI3

Abstract: This document details the design of the Common Agents for BI3.

Document Status: Approved

Author & Dept: Les Andrew – [Agent Team]

Contributors: Les Andrew, Clare Keane, Terry O'Brien, John Rayner, Anne Mohan, Rex Dixon & Paul Callow

Internal Distribution: Document Management Agent Document Library

External Distribution:

Agent Team Ref: TSC/AGT/076

WPRef: c:\documents\tsc_agt_076_com_agent_csr+.doc

Approval Authorities

Name	Position	Signature	Date
Rex Dixon	Agent Team TDA		
Peter Ambrose	Agent Team Leader		

FUJ00171949

0. DOCUMENT CONTROL

0.1 Contents

FUJ00171949

FUJ00171949

0.2 Review Details

Review Comments By:

Review Comments To: Les Andrew

Mandatory Review Authority	Name
Agent Team TDA	Rex Dixon
Agent Team Leader	Peter Ambrose
SSC	Mik Peach
Integration & Test	Janusz Holender
Optional Review/Issued for Information	
James Stinchcombe	Simon Fawkes
John Rayner	Anne Mohan
Terry O'Brien	Paul Callow
Harjinder Hothi	Keith Toh
Glenn Stephens	Mark Taylor

0.3 Document History

Version No.	Date	Reason for Issue	Associated CP/PinICL No.
Issue 0.1		None. This is the first Issue for CSR+. It is based on TSC/AGT/038. The C_HV_AUDIT agent has changed and as a result the CheckpointWriter utility is no longer required. The persistent object archiver is no longer required. C_HV_POCHECK is a replacement for B_HV_NAS. New Acknowledgement agents are included.	CP2179, CP1836
Issue 0.2		C_HV_POCHECK is renamed back to B_HV_NAS to reduce costs. C_HV_POACK is now run on every counter.	CP2182
Issue 0.3		Description of StopDesktopTransfer agent clarified.	
Issue 0.4		Add description of Message Submission Loader (C_LD_MEMO)	CP2412/3
Issue 0.5		Add (incomplete) description of the EOD Harvester (C_HV_EOD). Tidied up some formatting.	CP1633
Issue 1.1		Reformatted to conform to Fujitsu Services P.O.A. document standard	
Issue 1.2		Updated C_HV_SDT (plus introducing UserLockRequest.dll), B_HV_NAS and C_HV_EOD.	CP2472, PC44967
Issue 2.0		Comments incorporated and changed to Approved	PC49521, CP2634, PC53142

FUJ00171949

Issue 2.1		Updated for M1R, S06 & S10 changes: C_HV_POSCH added B_HV_NAS and C_HV_MON removed. C_HV_EOD updated C_HV_CACK and C_HV_POACK updated	CP2764, CP2797, PC59332, PC59240, CP2840, CP2882, CP2916, CP2919 CP2956 PC59977 PC53800
Issue 2.2		Updated for comments and W_HV_ALL added	CP2182 and CP2354
Issue 2.3		C_HV_POSCH changes for retrying failed connections (incomplete replication) and MemoView not repeated after the counter bounce	PC66146 PC66472
Issue 3.0	14/02/02	Status changed to approved	
Issue 3.1	22/02/02	Changes for BI3. SLA Harvester moved to [OMDBAGT]	
Issue 3.2	26/02/02	Updated for Audit changes	
Issue 3.3	10/03/03	Includes C_HV_POSCH changes for interfacing with CNIM.	CP3161
Issue 4.0	27/05/03	Status changed to Approved	

0.4 Document Cross-References

Tag	Ref	Vers	Date	Title	Source
[ACKFS]	AD/FSP/001 (TSC/AGT/065)			Functional Specification for a Generic Acknowledgement Agent for CSR+	PVCS/ agent lib
[AUDHLD]	SD/DES/115			Audit Data Storage & Retrieval High Level Design (CSR+)	PVCS
[CNIM]	CNIM-6			Counter Network Infrastructure Manager (CNIM)	
[AUDREC]	SD/DES/116			Audit Data Extraction & Filter High Level Design Specification for CSR+	PVCS
[GEN]	AD/DES/039 (TSC/AGT/058)			Generic Agent Components for CSR+ High Level Design	PVCS/ agent lib
[GENTAB]	AD/SPE/006 (TSC/AGT/079)			Pathway Agents: Generic Database Table Specifications for CSR+	PVCS/ agent lib
[OMDBAGT]	AD/DES/062			OMDB Agents High Level Design for BI3	PVCS/ agent lib
[MSGSUB]	SD/IFS/013			RDMC Host to Message Broadcast Agent - AIS	PVCS
[SLAHLD]	AD/DES/044 (TSC/AGC/002)			High Level Design of the Agent Changes for Inbound Data File Delivery Performance Measures	PVCS/ agent lib

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

FUJ00171949

0.5 Terminology

Term	Meaning
cluster	a set of clone correspondence servers (for resilience) supporting one subset of Post Offices.
LookupService	provides details on the mapping of Post Offices to Clusters/Correspondence Servers in the MultiRiposte environment.
urgent message	a Riposte message that needs to be sent to the Correspondence Server very quickly as late arrival could affect an SLA.
EOD	End of day
CNIM	Counter Network Infrastructure Manager (see [CNIM])
nailed-up	In the context of CNIM a nailed-up connection is one that is to be held open for a specified period of time. This applies independently to the network and Riposte connections.
'fixed'	In the context of CNIM a 'fixed' connection is one that is nailed up for SLA reasons. The Riposte connection is to be nailed up as a consequence of a 'fixed' network connection.
FRIACO	Fixed Rate Internet Access Call Origination. It is a type of network connection.
POA	Post Office Account

0.6 Changes in this Version

Version	Changes
4.0	Status changed to Approved

0.7 Changes Expected

Changes
None

FUJ00171949

1. GENERAL

1.1 Introduction

This document describes the functionality of the Common Agents for BI3.

The list of Common Agents for this release are:

a) Audit Harvester (C_HV_AUDIT)
b) Ping Agent (C_HV_PING)
c) Audit recovery utilities
d) Stop desk transfer service (C_HV_SDT)
e) UserLockRequest library
f) Central Acknowledgement Agent (C_HV_CACK)
g) Counter Acknowledgement Agent (C_HV_POACK)
h) Message Submission Loader (C_LD_MEMO)
i) End of Day Harvester (C_HV_EOD)
j) Counter Call Scheduler (C_HV_POSCH)
k) The description of the SLA Harvester (W_HV_SLA now M_HV_SLA) has been moved to [OMDBAGT].

These are discussed further in section 2.

1.2 Outstanding Issues

None.

FUJ00171949

2. USER DESCRIPTION

This section describes the Fujitsu Services P.O.A. Agent Functionality and Interfaces for each of the agents.

2.1 Audit Harvester (C_HV_AUDIT)

2.1.1 Requirements

The requirements for this agent are summarised as follows :-

1) To capture all messages received on a CS into a series of Text Files which are available to the Audit server
2) A new set of Text Files should be started on each new day and also when the previous set of files contains a given number of messages, currently set at 1,600,000.
3) Each set of Text Files comprise eleven files, the first containing non-outlet data, and the other ten grouped on the third digit of the Group Id
4) Any recovery is to be by applying complete files. The files are allowed to duplicate messages.
5) Should the Audit Server fail, or filestore not be available, then the Agent can be stopped to prevent further files being generated, on the basis that it can catch up later. Nothing will be done to the agent in this area other than allowing it to be stopped and started!
6) Should a CS lose its entire message store, then we need a mechanism to continue auditing on a different CS without losing any messages. We are allowed to generate significant duplicate messages in this case but not more than a few days worth.

2.1.2 Functional description

This agent is required to capture all the Riposte messages and write them to flat files. It is an interactive service that uses a Riposte Message port with selection criteria of "".

Its functionality is as follows :-

1) The agent connects to the local Riposte service and normally starts harvesting the Riposte messages from the "C_HV_AUDIT_*CheckpointDomain*_CHKPT" named checkpoint (where CheckpointDomain is either supplied in the registry or defaulted to the Correspondence Server's NodeId). It is this checkpoint that is updated as the output files are created. The service can be forced to start from another checkpoint (configured in the registry via OVERRIDECHECKPOINT) in cases where it is necessary to repeat some of the audit. In such cases the registry value is unset once C_HV_AUDIT_*CheckpointDomain*_CHKPT is updated. It is an error if an OVERRIDECHECKPOINT is specified but the checkpoint does not exist. If the normal checkpoint does not exist it tries the original CSR checkpoint name (which will no longer exist as checkpoints are not migrated from Riposte 5.4 to 6.1). If neither exists the agent fails. Any missing daily checkpoints are written (which may happen if the service has not been run for a while), otherwise an attempt to use yesterday's checkpoint may end up being 8 days ago.
2) There is no filter set up on the message port, so all messages are read.
3) Write any missing daily checkpoints based on the details in the daily persistent object.
4) A set of files are created in a working directory (base_name/Doing where base_name is configurable). If a file already exists it is overwritten as the same messages are about to be re-harvested. These files must permit overwriting and deletion.

FUJ00171949

5) Messages are read from the message port and appended to the appropriate text file (each separated by a new line). Occasionally a message includes a new line character and so the message occupies 2 lines in the file.

6) Whenever a configurable number of messages is read, or the time passes midnight, the files are saved and moved into an output directory of base_name/Done (where the audit server can pick them up from). The filename is configurable and the base_name is as used for the working directory. The files must allow deletion.

7) After saving the old files the message port is checkpointed using the agent's checkpoint name (C_HV_AUDIT_*CheckpointDomain*_CHKPT).

8) If the file change is because of a change of day (uses UTC time) a daily checkpoint is written (see section 2.1.3). A daily persistent object is also updated to record details on the last daily checkpoint written. This is in case the audit agent does not run over midnight for one or more nights and allows the missing daily checkpoints to be written on startup.

9) To create new files and continue auditing go back to step 4

10) Should the Service be stopped at any time, then the current files should be closed and moved to the output directory and the message port should be checkpointed before the code exits. Tivoli should be configured to monitor the service and restart it whenever it fails.

2.1.3 Daily Checkpoints and Checkpoint Override

In normal use the agent will use and maintain its own checkpoint and this will ensure that all messages are written to the audit files and messages are not duplicated. However there need to be contingencies for when there are problems such as

a) one or more of the audit files are corrupt or lost
b) the correspondence server on which the audit service is running fails and it is moved to another one.

2.1.3.1 Moving Correspondence server

When moving the audit agent from one Correspondence Server to another, either the new Correspondence server is a replacement and retains the same NodeId or it has a different NodeId.

In the case where the NodeId is unchanged, the audit agent can be started without any configuration change. However one likely reason for such a change is that the original one crashed, in which case an output file may be missing and recovery is as in section 2.1.3.2.

Where the new NodeId is different there are 2 choices

1) configure the OVERRIDECHECKPOINT registry setting to be the checkpoint name used by the old Correspondence Server i.e. C_HV_AUDIT_*OldNodeId*_CHKPT and let it continue using its own default checkpoint name

2) configure the CheckpointDomain registry setting to be the NodeId of the old Correspondence Server (or what was configured as the CheckpointDomain registry setting for the audit agent on the failed Correspondence server) so the new agent continues to use the original set of checkpoints.

2.1.3.2 Recreating Lost Files

The Audit agent sets up checkpoints at the start of every day using a daily cycle of names of the form *day_CheckpointDomain_CHCKPT*, where day is one of Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or Sunday (e.g. Monday_Wigan_CHCKPT.). This allows the audit agent to be reset to use any day's checkpoint up to one week old.

FUJ00171949

To recover a file we have to go back to a known daily checkpoint that is before the lost data. The required checkpoint name is then supplied as an OVERIDECHECKPOINT setting in the registry. This checkpoint is used to reset the audit agents normal checkpoint and then OVERIDECHECKPOINT is automatically unset (i.e. set to "").

The value in OVERIDECHECKPOINT can either be the appropriate day's checkpoint name or a number of days ago as "-1" (yesterday) to "-6". In this case, if there is no checkpoint for the appropriate *day_CheckpointDomain_CHCKPT*, an attempt is made using just the *day* as the name of the checkpoint (as used at CI3, but which will fail as checkpoints are not migrated from Riposte 5.4 to 6.1).

Note: At CSR+ the CheckpointWriter agent is not required to write a separate daily checkpoint to each host because the daily checkpoints written by C_HV_AUDIT is replicated by Riposte. The daily checkpoints are written by C_HV_AUDIT so that they are co-ordinated with the audit files. At CSR there was a remote possibility that the daily checkpoint written by the CheckpointWriter agent could be ahead of the audit agent.

2.1.4 Control Interfaces

This agent runs as an NT service with one service instance per locale per cluster. The service name is defined as:

TMSAudit<**s**><**n**>

where

'<**s**>' is the locale (B for Bootle and W for Wigan)

'<**n**>' is the cluster id

e.g. TMSAuditB2, TMSAuditW1

The underlying executable is:

C_HV_AUDIT.exe

2.1.5 Registry Entries Used

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

The registry entry values utilised are listed in [GEN]. The following registry entry values of particular interest are reproduced here for the convenience of the reader:

FUJ00171949

Value Name	Type	Max Size	Opt?	Usage
OUTFILE	EXPAND_SZ	256	Y	Full or relative path name of the base directory used for audit files. (See note 1 below).
FILENAME	EXPAND_SZ	256	Y	Format of the filenames to be used for the audit files. (See note 2 below).
CHECKPOINTDOMAIN	SZ	256	Y	Used to allow an agent instance to have distinct checkpoint. This distinguishes between Bootle and Wigan. If omitted the Correspondence Server's NodeId is used.
OVERRIDECHECKPOINT	SZ	256	Y	Omitted unless recovery is involved. Supplied as either an existing named checkpoint, "-1" to "-6" (for days ago) or "-INFINITE" (for RIPOSTE_BASE)
AGENTS_DATA_OFFICE	DWORD		N	A group to be used for containing the persistent object for identifying the latest daily checkpoint (see Note3)
PULSE_INTERVAL	DWORD		Y	number of messages read before writing audit file
MSGPORT_QUEUE_SIZE	DWORD		Y	Queue size used when creating a message port.
CLOSECHECKINTERVAL	DWORD		Y	checks for a closedown request every N milliseconds so that an audit file is written before the service closes.

Note 1: Two directories are created under this directory - "Doing" for temporary working files, and "Done" for the completed audit files.

Note 2: The format of the archive filenames that allows insertion of the date and time into the filename. This should be of the form

FN01_<AuditPointId>_<AuditPointSubId>%s_~_V001.arc (see [AUDHLD])

Note: this parameter must contain one, and only one, '%' which must be followed by 's'.

Note: this parameter must contain one '~' which is replaced by a digit 0-9 and 'X' to make a set of files

Note 3: group 999993 is allocated for this use on the live system. This group is explicitly not configured for access by the Lookup service as Audit uses it directly from each cluster.

Note 4: The registry settings for this agent are as follows:- PULSE_INTERVAL is 1,600,000 CLOSECHECKINTERVAL is 1 minute and MSGPORT_QUEUE_SIZE is 2048 (should the agent slip this far behind Riposte then Riposte has extra overheads managing the messages. No messages are lost).

2.1.6 Output Table Format

Not applicable. This agent does not access a database.

FUJ00171949

2.1.7 Riposte Messages

The daily persistent object is written to record details on the last daily checkpoint taken. On start-up it is used to detect missing checkpoints. It has the following format.

```
<Collection:AgentDaily>
<ObjectName:C_HV_AUDIT_CheckpointDomain>
<Data:
   <Day:>
   <Date:>
>
```

This agent reads all Riposte messages, no matter what their format (including checkpoints and attachments).

FUJ00171949

2.2 Ping Agent (C_HV_PING)

For supporting an engineer when checking that the counter is communicating with the correspondence server.

2.2.1 Functional description

This agent waits for messages including "<Application:EngineerApp><Data.TranType:CounterMsg>" and responds to the same Post Office with a priority message containing a TranType of CorrespondenceMsg.

2.2.2 Control Interfaces

This agent runs as an NT service with one service instance per cluster. The service name is defined as:

TMSPingAgent<n>

where '<n>' is the cluster id, omitted for cluster 1.

Thus the service names for two instances of the agent processing clusters 1 and 2 respectively will be **TMSPingAgent** and **TMSPingAgent2**.

The underlying executable is:

C_HV_PING.exe

2.2.3 Registry Entries Used

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

The registry entry values utilised are listed in [GEN]. The following registry entry values of particular interest are reproduced here for the convenience of the reader:

Value Name	Type	Max Size	Opt?	Usage
RIPOSTE	REG_SZ	15	N	Locale.
CLUSTERID	REG_DWORD		N	Cluster id for 'single' riposte connect.
PMEXPIRYTIME	REG_DWORD		Y	Riposte message Expiry[1]
PMINTERVALOP	REG_DWORD		Y	Interval operator defined by Riposte which controls how PMDISCINTERVAL is used. Default value is 0.
PMDISCONNECTINTERVAL	REG_DWORD		Y	Disconnect interval in milliseconds for Riposte priority messages. Default value is 0

[1] PMEXPIRYTIME is set to 1 day for this agent

FUJ00171949

2.2.4 Output Table Format

Not applicable. This agent does not access a database.

2.2.5 Riposte Message Format

Input Message:

```
<Application:EngineerApp>
<Data:
   <TranType:CounterMsg>
   <MessageID:messageid>
   ...
>
```

Output Message:

```
<Application:EngineerApp>
<Data:
   <TranType:CorrespondenceMsg>
   <MessageID:messageid>
>
```

where the Data.MessageId value on the output message is taken from the Data.MessageId value on the input message.

2.3 Audit Recovery Utilities

2.3.1 Introduction

An auditor (a real person) will need to retrieve and examine data from the audit files produced by C_HV_AUDIT for specified ranges of dates and Post Office FAD codes as described in [AUDREC]. The audit file names will contain the date, cluster id and a site identifier. Files will have been archived to tape. Utilities are required from the agent development team to:

- Establish the cluster ID for each Post Office specified to help identify the tape and files on the tape to be retrieved.
- Load a selection of messages from retrieved audit files into an empty Riposte message store.

All other stages of the audit retrieval and data inspection process are outside the scope of the design responsibility of the agent team.

2.3.2 Further Background

The files may have been archived onto tapes that are now at a remote warehouse. The process of retrieving these tapes, once identified, may therefore take more than a day. This means that the process may be split into 2 phases by a period of time:

1. Auditor specifies data to be retrieved; tapes are identified and requested.
2. Tapes are restored; data is imported to Riposte and the auditor inspects the data.

2.3.3 Assumptions

FUJ00171949

1. The reason for doing an audit is something like the fact that POCL are suspicious that a Post Master is fiddling the system and want to inspect the data for his PO.
2. The reason for auditing implies that the number of PO codes that the auditor wishes to inspect will be small, typically 1. (The number of times that this sort of audit has been required in R1C to December 1998 is about 5 for the 200 POs.)
3. We can limit the number of PO codes in the auditors list to a hard coded limit. (See below for limit chosen.)
4. The existing message store does not need to be saved. It can be deleted.
5. The original assumption that 'moving of Post Offices from one cluster to another is not supported' is no longer true. The mapping of FAD codes to cluster ids at the time of audit will not necessarily match the mapping at the time the audit data was archived. This is being addressed under CP2625 by allowing access to the RODB. Until this CP is implemented the current Agent_Get_ClusterId facility is to be used.
6. The volume of data to be loaded into the Riposte message store will be small (only a few Post Offices) therefore, Riposte will be configured for a single volume. This will also limit the number of nodes supported to ~5000.
7. The production and delivery of the initial message file used by Reset_Riposte_Message_Store is outside the scope of the agent team.

2.3.4 Getting the Cluster Id

2.3.4.1 Introduction

The Correspondence Server message store files written by C_HV_AUDIT are archived by Cluster, e.g. the two files (one at each site) gathered from Cluster 1 OFS 1 will have names in the format FN01_TMS_Cluster1B_B_~_DDMMCCYY_HHMMSS_V001.arc & FN01_TMS_Cluster1W_W_~_DDMMCCYY_HHMMSS_V001.arc.

What does OFS 1 mean?

But the Internal Auditor will, in general, be given requests for Audit Information by Outlet.

The retrieval process requires the Internal Auditor to use Legato Networker User to specify the files to be retrieved from tape. The retrieval process is defined in [AUDHLD]. Because of the volumes of TMS files, it is a requirement to only retrieve those files relevant to the Outlet. This necessitates being able to establish which Cluster an Outlet's messages will have been created on and hence archived from. This utility is required only to provide the correct Correspondence Server cluster number.

2.3.4.2 Requirements

The following requirements were supplied in an email from Michele Myles on 27/11/98 and updated by a meeting attended by Gareth Jenkins on 2/12/98.

- The utility must have a user interface and be able to be invoked from the desktop
- The utility must be able to run on an Audit Workstation (and RDMS Administrator workstation). At meetings held on 18/1/00 and 20/1/00 it was agreed that this only had to run on the Audit server.
- The outlet identify will be provided as input in FAD Code format (six or seven characters acceptable)
- The utility must provide the Correspondence Server cluster (1-4)

FUJ00171949

- The FAD Code entry field must allow for entry of multiple FAD Codes

At meetings held on 18/1/00 and 20/1/00 it was agreed that the Audit team would provide the GUI facility to run on the Audit workstations. This will run the Agent_Get_ClusterId, Reset_Message_store and Agent_Load_Audit_Data programs on an Audit Server. This GUI uses DCOM to a service on the Audit Server that is running under a priviledged user (LAuditors) and it is this service that runs the agent programs. The GUI expects the agent programs to exit with 0 and if one exits with anything else then the GUI displays the names of the files (on the Audit Server) that contain the program's output to stderr and stdout.

2.3.4.3 Agent_Get_ClusterId

2.3.4.3.1 Syntax

Agent_Get_ClusterId [filename]

2.3.4.3.2 Description

A list of FAD codes is read from the file specified (or standard input be default). A single list of the set of cluster identifiers of all Post Office codes is produced. The special value "ALL" is not supported.

Cluster Identifiers: <cluster_id_list>

where cluster_id_list is a comma separated list of cluster ids.

For example:

Cluster Identifiers: 1, 3, 4

Filename — filename is the full or relative name a file containing the list of PO FAD codes to lookup. If not supplied standard input is read. Standard input is terminated from a DOS prompt with ^Z followed by RETURN.

It is expected that the data will normally be supplied in a file.

2.3.4.3.3 FAD Code File Format

The FAD codes are supplied in the file formatted defined for Agent_Load_Audit_Data.

2.3.4.3.4 Messages

Warning and error messages are sent to standard error output.

No error, warning or progress messages are sent to standard output.

The NT event log is not used.

A message resource will not be used. Message text will be hard coded into the executable and internationalisation will not be supported.

FUJ00171949

2.3.4.3.5 Manageability

There are no manageability requirements.

This tool is not required to integrate with Tivoli or Maestro or any similar tools.

As stated above the NT event log is not used.

2.3.4.3.6 Permissions

No special user permissions.

2.3.4.3.7 Exit codes

0 on success
1 on error

2.3.4.3.8 Configuration

None.

2.3.4.3.9 Implementation Notes

The core of the utility can be provided by simply calling existing APIs in the dynamic link library LookupService.dll.

The following skeleton pseudo code illustrates the calls to be made to the cluster lookup service. Error handling and reporting the results of the lookup are omitted from the pseudo code as are the parameters to the calls.

```
Parse command line parameters

Read registry values. // RIPOSTE is mandatory

Open filename
                    Read file up to MAX_AUDIT_FAD_CODES FAD codes

                    if more than MAX_FAD_CODES in file
                                        generate an error and exit non-zero
Close filename

LookupConnect(…);
   For each FAD code
      GetClusterForGroup(…)
   end for each group id
LookupDisconnect()


Output list of cluster ids
```

2.3.5 Loading the Riposte Message Store

2.3.5.1 Introduction

FUJ00171949

The files will have been retrieved from tape and placed into a directory on the Archive Server. At this release all files (TMS and Non-TMS files are placed into the same directory). The retrieval from tape has been developed as part of the Retrieval component of the Audit Server. This utility must extract the appropriate messages from the TMS files in this directory. The messages to be extracted will be based on the extraction criteria as input by the user. The extracted messages will be used to populate an empty message store on the Audit Server.

2.3.5.2 Requirements

The following requirements were supplied in an email from Michele Myles on 27/11/98 updated following a meeting on 2/12/98 and further email exchanges with Michele.

- The utility must have a user interface and be able to be invoked from the desktop
- The utility must be able to run on an **Audit Server**
- The utility must process all TMS files in a specified directory
- The utility must accept the extraction criteria:

  Outlet:

  in FAD Code Format, multiple Outlets must be allowed, ALL must be an option

  OR

  FAD Code list

  Date Range:

  DATE FROM & DATE TO,
  to allow any message whose creation date falls on or between DATE FROM and DATE TO to be extracted (given that other extraction criteria are satisfied)

- The utility will extract the messages from the specified files based on the extraction criteria and populate the empty message store on the **Audit Server**
- The extraction functionality will be carried out on the **Audit Server**
- The utility will clear down the **Audit Server** Message Store before reloading the extracted messages
- A warning should be given to the User when Extraction parameters are accepted, to notify the fact that the message store will be cleared of existing messages.
- Successful or unsuccessful completion of the extraction and reload needs to be notified to the User.
- The extraction and reload should be in background mode, to allow the Audit Workstation to be used for other audit activities.

The following additional requirements for CP/1658 (archive recovery) have been derived by informal conversations with Simon Fawkes:

1. On exit the Audit_Load_Audit_Data should generate an event in the NT event log indicating whether loading was successful or failed using the appropriate severity (error or informational).

2. If loading failed, there should be error messages describing the failure in the NT event log.

2.3.5.3 Restore_Audit_Data

2.3.5.3.1 Syntax

Restore_Audit_Data -d date_range [-f filename] [-r ripostename] [-b initialmessagestore] [-c ripostemessagestore] [-p path_for_data_files] [-g filename_format]

FUJ00171949

2.3.5.3.2 Description

This DOS batch script will:

Get user parameters and perform minimal validation of the values supplied necessary to pass the parameters to other utilities.
Call Reset_Riposte_Message_Store
Call Agent_Load_Audit_Data

The utilities called must be on the PATH.

2.3.5.3.3 Messages

Error and warning messages generated by this script will be written to standard error output.

Messages generated by the other utilities called will not be redirected.

A message resource will not be used. Message text will be hard coded into the executable and internationalisation will not be supported.

2.3.5.3.4 Manageability

There are no manageability requirements.

This tools is not required to integrate with Tivoli or Maestro or any similar tools.

The NT event log is not used.

2.3.5.3.5 Permissions

The permissions required by the utilities called.

2.3.5.3.6 Exit Codes

0 on success
non-zero on error

2.3.5.3.7 Configuration

PATH must be configured so that the utilities called are on the PATH. (i.e. C:\Program Files\PathwayAgents should be on the PATH).

2.3.5.4 Reset_Riposte_Message_Store

2.3.5.4.1 Syntax

FUJ00171949

Reset_Riposte_Message_Store [-r ripostename] [-b initialmessagestore] [-c ripostemessagestore]

2.3.5.4.2 Description

This console application will:

1. Stop the Riposte service running on the machine specified in the registry (or the local machine if no name configured).
2. Wait for the Riposte service to terminate (with a configurable timeout).
3. Reset the message store (see issue below)
4. Start the Riposte service.
5. Wait for the service to enter "RUNNING" state (with a configurable timeout). (This does not necessarily mean the service is ready to process calls from clients.)

The location of the Riposte message store will be in a path configured through the registry or passed in by parameter.

The message store is reset by replacing any existing message store with a copy of the one identified in the parameter or in the registry value INITIAL_MESSAGE_STORE. This initial message store will be a minimal message store necessary for Riposte to work (e.g. it will contain the Riposte license). The existing message store will not be saved.

2.3.5.4.3 Messages:

Errors and warnings will be reported to standard error output starting with the keyword "Warning:" or "Error:".

Success will not generate any messages to standard output or standard error output.

2.3.5.4.4 Exit codes

On exit the process will return:

0	on success
1	on error (or timeout)

2.3.5.4.5 Configuration

The registry key for this is **Reset_Riposte_Message_Store**.

The registry entry values utilised are listed in [GEN]. The following registry entry values of particular interest are reproduced here for the convenience of the reader:

Value Name	Type	Size	Opt?	Meaning
RIPOSTE	SZ	15	Y	To define the machine on which the Riposte service is running. If the value is not supplied the local machine is assumed. Value may alternatively be supplied by parameter.
INITIAL_MESSAGE_STORE	EXPAND_SZ	Unlimited	Y	The path name of the source file to be copied into the Riposte message store location. This might be a remotely mounted share on the audit workstation from the audit server. Value may alternatively be supplied by parameter.
RIPOSTE_MESSAGE_STORE	EXPAND_SZ	Unlimited	Y	The path name of the Riposte message store. This might be a remotely mounted

FUJ00171949

				share on the audit workstation from the audit server. Value may alternatively be supplied by parameter.
SERVICE_CONTROL_TIMEOUT	DWORD	-	Y	A timeout in milliseconds for waiting for the service to stop or start. Default 120 seconds.
ALLOWED_RIPOSTES	SZ	Unlimited	Y	A comma-separated list of Riposte service names. When the Riposte service name is supplied by parameter, the name must be present in this list.

2.3.5.4.6 Implementation Notes

This utility is implemented in C.

2.3.5.4.7 Service control

See the service routines: OpenSCManager, OpenService, StartService, QueryServiceStatus, ControlService.

2.3.5.5 Agent_Load_Audit_Data

2.3.5.5.1 Syntax

Agent_Load_Audit_Data -d date_range [-f filename] [-r ripostename] [-p path_for_data_files] [-g filename_format]

2.3.5.5.2 Description

This console utility will read all the files in the path directory which have the file name format given by the filename format. The files will be processed in the order in which they were generated. This means sorting on a combination of <Date> and <Date_Sub_Index> in the file format:

FN01_<Audit_Point_Id>_<Audit_Point_Sub_Id>_<Date>_<Date_Sub_Index>_V001

Files will not be filtered on date and time since transactions within the specified date range may have been written to an audit file at a time which lies outside the specified range .

For each file the utility will load all the data in the files into Riposte filtering on the PO FAD codes and date range specified.

The message will be filtered if the Riposte <GroupId:> attribute in the message does not match a FAD code in filename.

The message will be filtered if the Riposte <Date:> and <Time:> attributes in the message is not ins the range of dates specified.

The utility connects to the local Riposte service by default using the Riposte client library (not multi riposte). The utility must assume that the Riposte service may still be initialising and not yet ready to service client requests. If this is the case it will wait until the service is ready (There will be no timeout on this wait).

FUJ00171949

Messages are inserted into Riposte using the Riposte interface RiposteImportMessage(). Messages are not synchronously committed. If a message fails as being invalid and does not end with a ">" the next line from the input file is appended. This is because Riposte messages may include new line characters.

-f filename — The full or relative pathname of the file containing the list of PO FAD codes to filter on. If not supplied the list will be read from standard input.

-d date_range — This must be a range of dates in the format:
start_date[:end_date]

where start_date and end_date are in the format:
CCYYMMDD

as used in the audit file name.

CCYY > 1971
MM 1 -12
DD 1-31

By default the utility will connect to the local Riposte service using the Riposte client dll (not the MultiRiposte dll). If the RIPOSTE registry value is set it will be used.

2.3.5.5.3 FAD Code File Format

Each FAD code is supplied on a separate line.

A FAD code is a 6 digit decimal number with a optional 7th check digit. The 7th character may also be a space. The check digit may be any character.

The special value "ALL" may be used to indicate all PO FAD codes. In this case no other FAD code may be supplied.

Lines are terminated with "CR LF".

White space (including blank lines) is allowed anywhere in the file except within the FAD code.

2.3.5.5.4 Messages

Errors and warnings will be reported to standard error output starting with the keyword "Warning:" or "Error:".

Success will not generate any messages to standard error output.

Progress may be reported to standard output. (i.e. the name of each file successfully processed)

The generic agent message resource, AgentEL.dll, will be used.

It is known that Riposte will only support data for a finite number of PO groups. This limit when reached will cause the processes to report an error and exit. (The limit is the number of markers that can be fitted into a 64K buffer. This is ~5000 nodes.)

Bad message formats will not be reported and will be silently filtered as not matching the selection criteria. (i.e. Corrupt audit files are not reported.)

FUJ00171949

To meet the archiving requirement for logging messages in the NT event log, all messages sent to standard output and standard error output will also be logged to the NT event log. In addition to the above messages, the deliverable and library version numbers will be logged to the NT event log.

Riposte return code	Description	Comment
MSG_RIPOSTE_IMPORT_SELF_MESSAGE 0xC10200A8L	An attempt was made to import a message created by the local message processor. Importation of self-originated messages is not supported.	Treated as a fatal error. The process will exit immediately. The Riposte service must be configured with a nodeid other than the one used to write any of the data to be loaded. If such a clash is encountered an error will be generated and the process will exit. Riposte this return code if the wrong node id is used.
MSG_MS_MESSAGE_ALREADY_PRESENT 0xC105008AL MSG_RIPOSTE_MESSAGE_ALREADY_PRESENT 0xC10200A9L	The specified message could not be inserted because it is already present in the message store.	This will be treated as a warning. This means that if the utility is re-run without resetting the Riposte message store this message will be generated for all previously entered messages.
MSG_RIPOSTE_INVALID_NUM 0xC1020012L	Invalid message sequence number.	This is treated as a warning. This is produced if the messages are imported out of order. Gaps are allowed but can not be filled in later. i.e. The sequence: 1,2,3,4,5,6,7,8,9,10,15 is allowed but 1,2,3,4,5,6,7,8,9,10,15,11,12,13,14 is not allowed by Riposte. Gaps may be present due to archiving but messages should never occur out of order on the tapes.
All other values	-	Treated as fatal error. The process will exit immediately.

2.3.5.5.5 Manageability

There are no manageability requirements.

This tools is not required to integrate with Tivoli or Maestro or any similar tools.

2.3.5.5.6 Permissions

The caller must have the permissions indicated in the RiposteImportMessage() manual page.

2.3.5.5.7 Exit codes

On exit the process will return:

0	on success
1	if any warnings were reported
2	if any errors were reported
3	if warnings and errors were reported

2.3.5.5.8 Configuration

The registry key for this is **Agent_Import_Audit_Data**.

The registry entry values utilised are listed in [GEN]. The following registry entry values of particular interest are reproduced here for the convenience of the reader:

FUJ00171949

Value Name	Type	Size	Opt?	Meaning
MAX_ERRORS_REPORTED	DWORD	-	y	The limit on the number of error or warning messages generated for a particular problem. Default 10.
FILENAME	EXPAND_SZ	unlimited	Y	The format of file names to be read as used by C_HV_AUDIT. . Value may alternatively be supplied by parameter.
INFILE	EXPAND_SZ	unlimited	Y	The directory path in which FILENAME will be found. . Value may alternatively be supplied by parameter.
CONNECTPOLLTIME			Y	
REPORTFAILDELAY			Y	
REPORTFAILPERIOD			Y	
RIPOSTE	SZ	15	Y	To define the machine on which the Riposte service is running. If the value is not supplied the local machine is assumed. . Value may alternatively be supplied by parameter.
ALLOWED_RIPOSTES	SZ	Unlimited	Y	A comma-separated list of Riposte service names. When the Riposte service name is supplied by parameter, the name must be present in this list.

2.3.5.1.9 Limits

The C type "time_t" is used to store time. The utility will therefore implement the "UNIX date bug" in 2039.

The full path name of files read are limited in length to _MAX_FNAME (256) characters.

2.3.5.1.10 Implementation Notes

None.

FUJ00171949

2.3.6 Integration

The following integration is left to Fujitsu Services P.O.A. (e.g. PIT):

1. The production and delivery of the initial message file used by Reset_Riposte_Message_Store. This should be the minimal messages store necessary to run Riposte (e.g. it should contain a license object).

2. The choice of the Riposte node id of the CS running on the Audit Server should be different to the nodeid used on any of the live system CS.

3. Registry configuration of the utilities document here using the sample .INI file supplied.

4. Riposte should have the configuration parameter DisableArchiving set to 1 so that expired messages can be viewed.

5. Riposte configuration parameter “EnableMessageImport” should be set to 1.

6. RestoreAuditData will require the permissions defined in RiposteImportMessage() manual page on the Riposte web page.

7. The cluster lookup service on the Audit Workstation or Server should be installed and configured exactly as it is on Agent Servers.

FUJ00171949

2.4 Stop Desktop Transfer Service (C_HV_SDT)

The Stop Desktop Transfer agent is provided as an NT service running on every counter to fix a timing problem when a desktop session transfer is requested. **The Stop Desktop Transfer code no longer works because of Escher changes to their underlying code. However these Escher changes seem to have fixed the original problem so effectively the service is redundant.**

2.4.1 Functional description

The Stop Desktop Transfer service has been introduced to fix a timing problem with the desktop.

This problem occurred if a user was logged onto a counter that was currently locked (because it was in an operation that could not be transferred), the counter desktop was working flat out, and the user attempted to log onto another counter to transfer the session. The user was able to log onto the new counter instead of being rejected, as the original counter desktop could not respond immediately to say it was locked and the second counter therefore assumed that it was ok to log the user on.

The service uses a message port to select any Transflush messages (i.e. those containing "<Application:$TransFlush>"). These are generated as a result of a request to transfer a user session. It always starts from 'current' as any replies to missed messages will be too late to be of any use anyway. If the request is against this counter (attribute FlushNodeId is this counter's NodeId) and the user is currently locked, respond with a TransferFailed message (the same message that the counter originally issued to inhibit the transfer).

Originally persistent objects were used to lock and unlock the user's session, which

1) had performance implications because of the number of object updates
2) could result in locks being left outstanding if a counter went down.

The current version uses interfaces in the UserLockRequest.dll to manage these locks

2.4.2 Message Sequence

A counter is locked for a user by the desktop calling LockUser specifying the user and transfer message e.g. calling LockUser("ALDINI", "Cannot perform session transfer while displaying picklist") on counter 1.

It is unlocked by calling UnlockUser for the user

When the user tries to log on to another counter, a TransFlush message is issued from this second counter; for example for user ALDINI on counter 2:

```
<Message:
    <GroupId:123456>
    <Id:2>
    <Num:16492>
    <Date:21-Jan-1999>
    <Time:16:40:49>
    <Expiry:1>
    <TellerID:ALDINI>
    <UserName:ALDINI>
    <FlushNodeId:1>
    <Application:$TransFlush>
    <CRC:F712763A>
```

In this example, StopDeskTransfer on counter 1 should find it is locked on calling CheckUserLock and respond by posting the TransferFailed message:

FUJ00171949

<Message:
<GroupId:123456>
<Id:1>
<Num:16810>
<Date:21-Jan-1999>
<Time:16:41:31>
<User:ALDINI>
<Expiry:10>
<Collection:Sessions>
<ObjectName:ALDINI>
<TellerID:ALDINI>
<SessionReceived:1>
<TransferFailed:Cannot perform session transfer while displaying picklist>
<Version:109>
<CRC:7A56CF15>

2.4.3 Control Interfaces

This agent runs as an NT service with one service instance per counter. The service name is defined as:

StopDeskTransfer

The underlying executable is:

C_HV_SDT.exe

2.4.4 Registry Entries Used

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

The registry entry values utilised are listed in [GEN].

2.4.5 Riposte Message Format

See section 2.4.2 above.

FUJ00171949

2.5 UserLockRequest Library

2.5.1 Functional Description

UserLockRequest.dll, is provided in order to isolate the mechanism for managing the Locks from the Desktop code and the StopDeskTransfer agent code (see Stop Desktop Transfer Service (C_HV_SDT)). This library will contain the following interfaces for managing the locks:

a) LockUser
b) UnlockUser
c) CheckUserLock

The StopDeskTransfer agent is no longer required so these interfaces are only required to satisfy historical calls within the agent and desktop code.

An additional interface of GetULRErrorString is provided for accessing the text associated with a return code.

These interfaces are available for calling from both C and Visual Basic.

An associated header file, **UserLockRequestAPI.h**, is included for use in the calling 'C' source code. The file defines at least the following:

ULR_MAX_MESSAGE_SIZE	256
ULR_STATUS_UNLOCKED	0
ULR_STATUS_LOCKED_THIS_USER	1
ULR_STATUS_LOCKED_DIFFERENT_USER	2

2.5.1.1 Response codes

All responses returned by these functions are error codes in the standard NT format. The error codes introduced for this dll are defined in UL_Msgs.h. When using Visual Basic the following statement should be included for the standard error responses:-

```
Private Enum ULRResults
ULR_LOCK_IN_USE = &HC4280010
ULR_USER_ALREADY_HOLDS_LOCK = &H44280011
ULR_LOCK_TRANSFERED_FROM_ANOTHER_USER = &H84280012
ULR_UNLOCK_WHEN_NOT_LOCKED = &H44280013
ULR_LOCKED_TO_DIFFERENT_USER = &HC4280014
End Enum
```

FUJ00171949

2.5.1.2 LockUser

The **LockUser** function allocates a Lock to a user on the local counter. Note that both 0 and ULR_LOCK_TRANSFERED_FROM_ANOTHER_USER can be returned on successfully allocating a lock.

LockUser creates a Lock for the named user and associates the "message" with the Lock. If the Lock is already assigned to the user, the new message supersedes the previously associated message and the function returns success.

If the Lock is already assigned to a different user, the Lock is reassigned to the specified user. However, as this represents an error of usage, ULR_LOCK_TRANSFERED_FROM_ANOTHER_USER is returned.

EPOSS should decide whether to output this event (which is categorised as a Warning) and/or ULR_LOCK_IN_USE to the Event Log.

The Lock created by this function can be released explicitly by calling **UnlockUser** (see 2.5.1.3). It is released implicitly whenever the process acquiring the Lock terminates.

2.5.1.2.1 Synopsis

In 'C':

```
unsigned long _stdcall LockUser (
          unsigned char *szUser,
          unsigned char *szTransferMessage);
```

In Visual Basic:

[Public|Private] Declare Function LockUser Lib "UserLockRequest.dll" Alias "_LockUser@8" (ByVal *szUser* As String, ByVal *szTransferMessage* As String) As Long

2.5.1.2.2 Parameters

szUser	A null-terminated string containing the name of the user to whom the Lock is to be assigned.
szTransferMessage	A null-terminated string containing the "message" to be associated with the Lock. If necessary this will be truncated to ULR_MAX_MESSAGE_SIZE

This "message" is typically of the form "Cannot transfer session while …" and will be displayed on the Desktop when a Lock prevents a session from being transferred.

2.5.1.2.3 Return values

The function returns 0 if successful. Otherwise it returns a failure code (in the form of an NT event number), which may be an NT failure code, or it may be specific as follows:

FUJ00171949

ULR_LOCK_IN_USE	Error: the Lock is already assigned elsewhere. This will not happen if there is only one desktop process.
ULR_LOCK_TRANSFERED_FROM_ANOTHER_USER	Warning: this process already has the Lock assigned to a different user. The Lock has been successfully reassigned to the specified user.

2.5.1.3 UnlockUser

The **UnlockUser** function releases a Lock assigned to a user on the local counter. This lock must be held by the process requesting the unlock.

UnlockUser release a Lock previously assigned by this process to the named user. No message is retained when the lock is released.

We have decided to fail an unlock request when the wrong user is specified

2.5.1.3.1 Synopsis

In 'C':

```
unsigned long _stdcall UnlockUser (
          unsigned char *szUser);
```

In Visual Basic:

[Public|Private] Declare Function UnlockUser Lib "UserLockRequest.dll" Alias "_UnlockUser@4" (ByVal *szUser* As String) As Long

2.5.1.3.2 Parameters

szUser	A null-terminated string containing the name of the user to whom the Lock is currently assigned.

2.5.1.3.3 Return values

The function returns 0 if successful. Otherwise it returns a failure code (in the form of an NT event number), which may be an NT failure code, or it may be specific as follows:

ULR_LOCKED_TO_DIFFERENT_USER	Cannot unlock as the Lock is assigned to a different user.
ULR_UNLOCK_WHEN_NOT_LOCKED	Cannot unlock as not locked

FUJ00171949

2.5.1.4 CheckUserLock

The **CheckUserLock** function checks the status of the Lock on behalf of a user on the local counter. This function is to be used by StopDeskTransfer when checking locks.

CheckUserLock checks the Lock on behalf of the named user and returns its status in pdwLockStatus:

a) If the Lock is unassigned, ULR_STATUS_UNLOCKED is returned.
b) If the Lock is assigned to the named user, ULR_STATUS_LOCKED_THIS_USER is returned.
c) If the Lock is assigned to a different user, ULR_STATUS_LOCKED_DIFFERENT_USER is returned.

2.5.1.4.1 Synopsis

In 'C':

```
unsigned long _stdcall CheckUserLock (
          unsigned char *szUser,
          unsigned long *pdwLockStatus,
          unsigned char szTransferMessage[ULR_MAX_MESSAGE_SIZE + 1]);
```

In Visual Basic:

[Public|Private] Declare Function CheckUserLock Lib "UserLockRequest.dll" Alias "_CheckUserLock@12" (ByVal *szUser* As String, *pdwLockStatus* As Long, ByVal *szTransferMessage* As String) As Long

2.5.1.4.2 Parameters

szUser	A null-terminated string containing the name of the user for whom the status of the Lock is to be checked.
pdwLockStatus	A pointer to the variable which will receive the status of the Lock: ULR_STATUS_UNLOCKED ULR_ STATUS_LOCKED_THIS_USER ULR_ STATUS_LOCKED_DIFFERENT_USER
szTransferMessage	A pointer to a buffer which will receive a null-terminated string containing the "message" associated with the Lock (if ULR_STATUS_LOCKED_THIS_USER or ULR_STATUS_LOCKED_DIFFERENT_USER).

2.5.1.4.3 Return values

The function returns 0 if successful. Otherwise it returns a failure code (in the form of an NT event number), which may be an NT failure code.

2.5.1.5 GetULRErrorString

Returns the text associated with a response from one of the other functions.

2.5.1.5.1 Synopsis

FUJ00171949

In 'C':

unsigned char* _stdcall GetULRErrorString (unsigned long lEventNo, char *szBuf, int wSize);

In Visual Basic:

[Public|Private] Declare Function GetULRErrorString Lib "UserLockRequest.dll" Alias "_GetULRErrorString@12" (ByVal lEventNo As Long, ByVal szBuf As String, ByVal wSize As Long) As String

2.5.1.5.2 Parameters

lEventNo	An error response value.
szBuf	A buffer for containing the returned text associated with lEventNo
wSize	the size of szBuf

2.5.1.5.3 Return values

The function returns the pointer for szBuf (allows the direct passing of the return from this call as a parameter to another function e.g. printf)

2.5.2 Registry Entries Used

The default registry configuration of this dll is expected to be acceptable. Should there be any need to use a different configuration the Registry key for this dll will be **UserLockRequest** (under SOFTWARE\ICL\PathwayAgents).

The following registry entry values are utilised (see [GEN]):

Value Name	Type	Max Size	Opt?	Usage
FILENAME	EXPAND_SZ	2048	Y	The full pathname of the well-known file. Defaults to c:\counters\userlocks\AgtUserLocks.txt
TRACE_LEVEL	DWORD		Y	Causes trace messages to be written to the event log
TESTENVIRONMENT	DWORD		Y	A setting of 0x01000000 causes Riposte messages to be written for locks and unlocks, but only if Riposte is connected to the current thread. No other bits should be set without prior authorisation from the agent team.

2.5.2.1 Supplement for TESTENVIRONMENT

When the 0x01000000 bit is set in TESTENVIRONMENT then Riposte messages are written every time a lock is successfully acquired or released. However if Riposte is not connected on the first access to the library then this trace is switched off. Riposte messages are used for this trace so that the actions of the StopDeskTransfer agent can be correlated with the locks (since the agent checks the locks in response to Riposte Messages).

The trace message for locks is

FUJ00171949

<Application:UserLockRequest><Data:<Action:Lock><UserName:...><TransferMessage:...><PID:...
>>

The trace message for unlocks is
<Application:UserLockRequest><Data:<Action:Unlock><UserName:...>>

2.5.3 Implementation Notes

The **UserLockRequest** library uses file locking (see registry setting for FILENAME) to control the user locking. If necessary it will create this file and up to one level of directory. A process will acquire a user lock by opening this file with write access and will release the lock by closing the file.

2.6 Central Acknowledgement Harvester (C_HV_CACK)

This agent is provided as a tool to be used in the monitoring of SLAs. It is not a solution in itself but it can be configured via Riposte objects to

a) respond to particular messages as they arrive
b) check for a late response to a message.

NOTE: In order to measure an SLA a user will also have to

1) write a message to be acknowledged from another agent
2) configure the acknowledgement agent or agents to handle this message (may require dll for optional hook)
3) harvest messages from the acknowledgement agent via another agent.

Although originally intended as a tool for monitoring SLAs there is no reason why its capabilities cannot be used just to acknowledge the receipt of particular messages.

In cases where the handling of particular acknowledgements may have an adverse affect on other acknowledgements they may be handled by a separate instantiation of the agent.

2.6.1 Functional description

The functional specification of the generic Acknowledgement Agent is described in [ACKFS]. The agent only knows which messages to process and how from its configuration via the CSAckDefinition objects available in the virtual office (possibly suffixed if DEFINITION_SET is specified in the registry). It assumes that these objects have been tested by the unit providing them and does not include any further validation. However if a particular entry fails consistently it is switched off to avoid impacting the other acknowledgements.

The agent uses 4 message port threads; 2 where only current messages are to be processed and 2 where all messages are to be processed (using named checkpoints). In each case there is one thread for responding to messages and one when checking for late responses. The threads for responding to messages can each handle up to 16 configuration entries and the threads for checking late responses can each handle 8 configuration entries (since each entry requires a pair of filters).

The checkpoint names used are:

FUJ00171949

Checkpoint Name	Comments
AGT_C_HV_CACK_<*Domain*><*ClusterId*>	Handles acknowledgement message types.
AGT_C_HV_CACK_<*Domain*>TIMEOUT_<*ClusterId*>	Handles timeout message types

where <*Domain*> is substituted by the CHECKPOINTDOMAIN registry setting and is omitted by default.

2.6.2 Recovery requirements

Same as for any other interactive harvester.

2.6.3 Sizing & Performance

Same as for any other interactive harvester

2.6.4 Control Interfaces

This agent runs as an NT service with one service instance per cluster. There may be several variants of the acknowledgement agent (distinguished by different DEFINITION_SET and CHECKPOINTDOMAIN registry values), each with their own set of services. Currently there is only one set and its service name is defined as:

TMSCAcknowledge<n>

where '<n>' is the cluster id, omitted for cluster 1.

Thus the service names for two instances of the agent processing clusters 1 and 2 respectively will be **TMSCAcknowledge** and **TMSCAcknowledge2**.

The underlying executable is:

C_HV_CACK.exe

2.6.5 Maestro schedule

N/A. This agent runs all the time and is under Tivoli control.

2.6.6 Registry Entries Used

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

The registry entry values utilised are listed in [GEN]. The following registry entry values of particular interest are reproduced here for the convenience of the reader:

FUJ00171949

Value Name	Type	Max Size	Opt?	Usage
RIPOSTE	SZ	15	N	To define the locale used for Riposte services.
CLUSTERID	DWORD		N	Cluster id for 'single' riposte connect.
VIRTUALOFFICE	DWORD		N	The group id of the virtual office.
PULSE_INTERVAL	DWORD		Y	Pulse interval parameter supplied to RiposteCreateCheckpointedMessagePort. It is the count of messages processed by Riposte before a Riposte pulse is generated, which allows the agent to write a checkpoint.[2]
DEFINITION_SET	SZ			For this agent it supplies a suffix to the collection name used for the configuration details i.e..**CSAckDefinition***suffix* Default ""
CHECKPOINTDOMAIN	SZ			For this agent it modifies the set of checkpoint names used by the agent. It is normally used in conjunction with DEFINITION_SET when multiple service variants are required. Default ""

See also CLOSECHECKINTERVAL, CONNECTPOLLTIME and TOTALCONNECTIONTIMEOUT.

2.6.7 Output Table Format

Not applicable. This agent does not access a database.

2.6.8 Riposte Message Format

This agent is configured from the **CSAckDefinition***DEFINITION_SET* (i.e. **CSAckDefinition** suffixed by the registry value for DEFINITION_SET) objects in the virtual office as defined in the FS [ACKFS].

[2] PULSE_INTERVAL is set to 1010000 for this agent. The agent uses 2 checkpoints and skews this value slightly for one of them.

FUJ00171949

2.6.9 Miscellaneous Information

Both the Central and Counter Acknowledgement Harvesters share common source files. This agent's source is selected by the absence of a definition of COUNTER_AGENT.

The differences between the counter and central agents are:

a) Riposte connections. The counter agent connects to the local host and the centre agent uses LUC to connect to the appropriate Riposte service. The central agent additionally connects to MultiRiposte but this is disconnected after the thread initialisation is complete.
b) Configuration object names. The counter agent is configured by persistent objects with a Collection name starting WSAckDefinition and the central agent by objects with a Collection name starting CSAckDefinition. The former accesses objects from the local host and the latter from the Virtual Office.
c) Executable, Service and Checkpoint names.
d) Registry values used.

The purpose of each of the four work threads is as follows:

a) To listen on a checkpointed message port for a Request acknowledgement message and to generate a Result acknowledgement message
b) To listen for a current Request acknowledgement message and to generate a Result acknowledgement message
c) To listen on a checkpointed message port for Request and Result acknowledgement messages and to generate a Timeout message if necessary
d) To listen for current Request and Result acknowledgement messages and to generate a Timeout message if necessary

The threads need to be dynamically initialised with the data from persistent objects that specify what messages to listen for and give details on how to generate the acknowledgement or timeout message. Each thread calls the same initialisation routine and a critical section is used to allow one thread to read all the persistent objects and set up the data in the appropriate thread.

Each thread is dynamically allocated space for a table of 16 pointers, 16 being the maximum number of filters allowed in one work thread. The two acknowledgement threads can be configured with up to 16 message types but the two timeout checking threads have a maximum of 8 message types because two filters are required for each message type.

The data from persistent objects which configures the timeout threads is pointed to by alternate entries in the pointer table in order to maintain a consistent index into the filter and message handler arrays and into the pointer table array. I.E. for a message type the filter and message handler indexes for a Request acknowledgement message are n the filter and message handler indexes for a Result acknowledgement message are n+1 and the index into the pointer table is n+1.

The filter for a thread is set to a dummy filter <Dummy:99> if there is no persistent object to configure a thread. This causes that particular thread to simply wait on the message port for the duration of the run of the agent.

On receipt of a Request message for timeout checking a new entry is included on timeout chain in timeout order. Whenever this timeout is inserted at the head of the chain the next interrupt time is reset.

When a Result message is received this thread looks for the matching Request message on the timeout chain and, if found, removes it. When removing the first entry on the timeout chain the next interrupt time is also reset

FUJ00171949

Should an entry on the timeout chain exceed its timeout time an interrupt will occur. These timeouts for any late responses are handled in a separate thread. This timeout thread checks through the chain and generates a timeout message for each entry that has timed out. The timeout thread then removes those entries from the chain and resets the next interrupt.

All external libraries named in the persistent objects are dynamically loaded when the agent is configured. At the same time the addresses of the external functions are mapped and saved. If either of these actions fail then the persistent object is not configured into the agent. If an Initialise option (i.e. <Initialise:1>) is included for a ViaFunction then the function is called on startup with a dummy input message of "INIT". If the function needs to wait for a resource to become available it should return RC_2_RETRY (i.e. 2) and it will be retried after a short interval. Returning in this way allows the agent to handle closedown requests and timeouts. To fail the service it returns RC_4_UNEXPECT (i.e. 4).

A *try-except* block is used around the calls to these external functions in order to increment a count of any exceptions that occur. There is a count for each external function and when it exceeds a certain limit, currently set at 5, the InError flag is set and the message type subsequently ignored.

FUJ00171949

2.7 Counter Acknowledgement Harvester (C_HV_POACK)

This agent is provided as a tool to be used in the monitoring of SLAs. It is not a solution in itself (see [SLAHLD]) but it can be configured via Riposte objects to

a) respond to particular messages as they arrive
b) check for a late response to a message.

NOTE: In order to measure an SLA a user will also have to

a) write a message to be acknowledged from another agent
b) configure the acknowledgement agent or agents to handle this message (may require dll for optional hook)
c) harvest messages from the acknowledgement agent via another agent (see SLA Harvester (W_HV_SLA)).

In cases where the handling of particular acknowledgements may have an adverse affect on other acknowledgements they may be handled by a separate instantiation of the agent. This is provided for consistency with C_HV_CACK but, as it requires registry configuration, it is unlikely to be used on the counters and is not fully covered here.

2.7.1 Functional description

The functional specification of the generic Acknowledgement Agent is described in [ACKFS]. The agent only knows which messages to process and how from its configuration via the WSAckDefinition objects available at that Post Office. It assumes that these objects have been tested by the unit providing them and does not include any further validation. However if a particular entry fails consistently it is switched off to avoid impacting the other acknowledgements.

A copy of this agent runs on every counter and by default there will be an acknowledgement result generated from each counter.

The agent uses 4 message port threads; 2 where only current messages are to be processed and 2 where all messages are to be processed (using named checkpoints). In each case there is one thread for responding to messages and one when checking for late responses. The threads for responding to messages can each handle up to 16 configuration entries and the threads for checking late responses can each handle 8 configuration entries (since each entry requires a pair of filters).

The checkpoint names used are:

Checkpoint Name	Comments
AGT_C_HV_POACK_%d%N	Handles acknowledgement message types.
AGT_C_HV_POACK_%dTIMEOUT_%N	Handles timeout message types

where %N is the NodeId of the counter and %d is "".

2.7.2 Recovery requirements

Same as for any other interactive harvester.

2.7.3 Sizing & Performance

Same as for any other interactive harvester

FUJ00171949

2.7.4 Control Interfaces

This agent runs as an NT service with one service instance per counter. The service name is defined as:

TMSPOAcknowledge

The underlying executable is:

C_HV_POACK.exe

2.7.5 Maestro schedule

N/A. This agent runs all the time.

2.7.6 Registry Entries Used

This agent should not require any registry configuration for normal running. When values are being supplied then the registry key for this agent is **C_HV_POACK**.

The registry entry values utilised are listed in [GEN].

The following registry entry values of particular interest are reproduced here for the convenience of the reader:

Value Name	Type	Max Size	Opt?	Usage
TESTENVIRONMENT	DWORD		Y	Used for agent team testing

See also DEFINITION_SET and CHECKPOINTDOMAIN.

2.7.7 Output Table Format

Not applicable.

2.7.8 Riposte Message Format

This agent is configured from the WSAckDefinition objects in the local office as defined in the FS [ACKFS].

2.7.9 Miscellaneous Information

Both the Central and Counter Acknowledgement Harvesters share common source files. This agent's source is selected by the inclusion of a definition for COUNTER_AGENT. See the Central Acknowledgement Harvester for further details.

FUJ00171949

2.8 Message Submission Loader (C_LD_MEMO)

The Message Submission Loader agent detects messages written to the RDMC database by the Message Submission Application [MSGSUB] and loads them into the specified PO counters.

2.8.1 Functional description

The Message Submission Loader agent detects messages stored in the 'desktop_memo' table by scanning the table either periodically or when it receives an alert. It formats the messages as described in the tables below, and writes them to Riposte to be sent to the PO groups specified in the corresponding 'desktop_memo_distribution' table entries. These two tables are in the RDMC database.

2.8.2 Control Interfaces

This agent runs as an NT service. As with other interactive loaders only one instance of this service should be run at a time. The service name is defined as:

TMSMessageSubmission

The underlying executable is:

C_LD_MEMO.exe

2.8.3 Maestro schedule

N/A . This agent should run all the time.

2.8.4 Registry Entries Used

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

The registry entry values utilised are listed in [GEN].

2.8.5 Oracle Table Format

The Message Submission Application [MSGSUB] creates a row in the desktop_memo table for each message/memo to be sent. This must only be done after (or in the same commit unit as) all the corresponding entries desktop_memo_distribution table have been created. The definitive definition of these tables is in [MSGSUB] and are reproduced here for convenience.

2.8.5.1 DESKTOP_MEMO Table

The DBALERT TMSALERTMESSAGESUBMISSION is used to trigger the agent when new rows are inserted into this table.

FUJ00171949

Attribute	Type	Max Size	NULL?	Comments
version	number	2	No	Identifies the format of the row in the interface table. It is included to enable the format of the table to be changed at future release of the software. The value at initial release will be 1. The agent will ignore any rows which do not have this field set to 1
memo_id	number	10	No	The desktop_memo unique identifier. Identifies the corresponding 'desktop_memo_distribution' entries.
message_id	varchar2	50	No	Consists of the caption, date & time the message was created and the Workstation name.
caption	varchar2	18	No	Identifies the memo title and caption used on the invoking button at the counter.
authority	varchar2	20	No	Identifies the department or organisation of the sender of the message: Current expected value is: 'CustServ' but may be POCL in the future.
text	varchar2	1700	No	The free format text supplied by the message sender. [MSGSUB] imposes restrictions that are more rigorous than those for a Riposte value.
expiry	number	3	No	Number of days the memo is retained by riposte.
priority	varchar2	4	No	Indicates the priority of the message that influences the sequence in which the memo is presented at the Counter. Permitted values are 'High' & 'Low'.
user_group	varchar2	15	No	The user role to which the memo is to be made available at the counter.
workstation_info	varchar2	100	No	Identifies the NT user, NT Domain and Workstation name of the creator of the memo.
actioned_ind	varchar2	1	Yes	Agent action indicator. Values are: 'N' = New row added by the Message Submission Application. Null = The agent has completed processing for this memo. 'F' – The agent cannot write the memo to the message store due to a problem occurring prior to distribution.
time_actioned_ind_ updated	date		No	The date & time the actioned_ind was last updated.

2.8.5.2 DESKTOP_MEMO_DISTRIBUTION Table

The Message Submission Application creates a row in the desktop_memo_distribution table for each post_office that a message/memo is to be sent to.

FUJ00171949

Attribute	Type	Max Size	NULL?	Comments
memo_id	number	10	No	The desktop_memo unique identifier. Identifies the corresponding 'desktop_memo' entry.
group_id	number	10	No	The Post Office unique identifier.
actioned_ind	varchar2	1	Yes	Agent action indicator. Values are: 'N' = New row added by the Message Submission Application. Null = The agent has successfully written the memo defined by memo_id to the Correspondence Server for the Post Office specified by the group_id 'F' – The agent has failed to write the memo defined by memo_id to the Correspondence Server for Post Office specified by the group_id.
time_actioned_ind_ updated	date		No	The date & time the actioned_ind was last updated.

2.8.5.3 Generic Database Tables

This agent uses no database tables or sequences of the generic types specified in [GENTAB].

2.8.6 Riposte Message Format

```
<GroupId:>
<Expiry:>
<Application:MemoView>
<Sender:C_LD_MEMO>
<MessageID:>
<MsgVersion:
   <Major:>                    // Fujitsu Services P.O.A. Release
   <Minor:>                    // Application Version
>
<Data:
   <Text:>
   <Caption:>
   <UserGroup:>
   <Brand:>
   <Priority:>
>
```

.

2.8.7 Riposte Message Mapping

FUJ00171949

Riposte field	desktop_memo field(s)	Comments
GroupId	group_id	From desktop_memo_distribution table field
Expiry	expiry	
Application		Hard-coded value: 'MemoView'
Sender		Hard-coded value: 'C_LD_MEMO'
MessageID	message_id	
MsgVersion.Major		Hard-coded value: '2+'
MsgVersion.Minor		<C_LD_MEMO:vvv> where vvv is value of PRODUCT_VERSION (from VersionInfo.h)>
Data.Text	authority text	authority & '||' & text
Data.Caption	caption	
Data.UserGroup	user_group	
Data.Brand	authority	
Data.Priority	priority	

2.8.8 Implementation Notes

The agent reads entries from the desktop_memo table 1 at a time and for each entry reads the corresponding desktop_memo_distribution entries in blocks of 1000. It commits the database at the end of each such block and or when it completes the processing of a desktop_memo entry. The aim is to have a simple scheme that prevents very large commit units when a message (or series of messages) are sent to a large number (up to 20000) of POs. Performance is unlikely to be critical when a message is sent to a small number of PO groups.

The agent avoids introducing a new commit point into the standard loop by the use of a new response code of RC_7_RETRY_IACT (related to RC_5_RETRY_HERE used by bulk loaders). This causes the database scan to be repeated when a message is to be sent to 1000 or more PO groups.

As normal it is only when the agent commits that changes to the actioned_ind and time_actioned_ind_updated fields will become visible. The actioned_ind and time_actioned_ind_updated fields in the desktop_memo table will only be updated when the processing of the entry is completely finished i.e. after all the associated desktop_memo_distribution entries have been processed. This requires reintroduction of the "mark_OK" Oracle context hook for interactive loaders.

FUJ00171949

2.9 End of Day Harvester (C_HV_EOD)

The End of Day interactive harvester provides a common view of which outlets' EOD Markers have been received by the 'harvest cut-off time', so that the various bulk harvesters (TPS, APS & OBCS) harvest the same days' transaction for the same outlets. It was introduced by CP1633. The 'harvest cut-off time' is normally determined under Maestro control, but the EOD harvesters have a latest cut-off time that is configured through reference data. The EOD harvester runs continuously and on detecting that it has reached the 'harvest cut-off time' for the current day carries out some processing referred to as the 'Circadian Transition'. A Maestro schedule is checking how many EODs have been received by the EOD harvesters (using agent_count_objects) and declares 'harvest cut-off time' (call of Declare_EOD_Harvest_Cutoff_Now) on achieving certain thresholds. The call of Declare_EOD_Harvest_Cutoff_Now writes an EODCutoffNow message to each cluster and this is harvested by the EOD harvesters.

A chain is maintained for every Post Office with details of the EODs received each day. These details include an EOD date, a received date, an EOD marker and a pointer to the harvest trailer. The received date runs from one day's 'harvest cut-off time' to the next. This received date is very important as the bulk harvesters only include EODs where the received date is less than or equal to their current business day. An EOD date may be earlier than their business day, but will be ignored if its received date is later.

During the day the harvester delays the processing of the received EODs to allow time for messages to be replicated from the counters. Circadian Transition causes a general flushing out for the current day and then checkpoints. In very rare circumstances an EOD entry for a particular received date will not be available for that business day and will be left to the next business day. This happens when all the EOD details are available, but some of the required messages have not been replicated in time for the 'Circadian Transition'. These messages are rewritten with the correct received date once it is deemed that the required messages are replicated. All the bulk harvesters will still harvest such EODs on the same processing date. The EOD harvesters confirm the receipt of the EOD messages with an EODConfirmation message.[3]

When all the EOD harvesters (there is one per cluster) have completed their set of EOD entries for a received date a new EODStatus object is created. This object lets the bulk harvesters know that there will be no more EOD entries set up for the current business day and that they can therefore finish once all the existing entries are processed.

[3] Lack of such confirmation may require the Post Master to provide printed details on the APS transactions.

FUJ00171949

2.9.1 Functional description

The EOD harvester listens for EOD Harvest Trailer messages and EODCutoffNow messages on a checkpointed message port. The harvester stores the group/node/num for each EOD Harvest Trailer and the EOD Marker pointed to in the Trailer onto a queue. The queue is used to delay the processing of an EOD to allow time for all the messages in the group to replicate to the Correspondence Server. The queue is processed (i.e. the EOD Markers are processed):

a) each time a new Trailer is detected and the delay period (Registry value SYNCHRONIZATION_DELAY, default 10000 milliseconds=10 seconds) has elapsed. Only half the queue is processed on each such timeout.

b) after a timeout on reading the message port (Registry value CLOSECHECKINTERVAL, default 15000 milliseconds=15 seconds). The whole queue is processed.

c) at Circadian Transition. The whole queue is processed.

d) after processing SUCCESS_UNIT_COUNT messages, before taking a checkpoint. The whole queue is processed.

e) when the harvester is closed down (seldom used as harvester is designed to run continuously). The whole queue is processed.

Before processing an EOD Marker, reconciliation is checked to determine whether all the messages 'below' the marker have been replicated to the Correspondence Server. If the group is not reconciled the entry remains on the queue to be processed after further delay(s). The queue size allows 5000 entries (groups) per cluster. If the queue should become full the entries are forced out i.e. the EOD is processed even though it is not reconciled. Entries are also forced out when taking a checkpoint.

When processing an EOD Marker the harvester creates/updates the PO EODStatus object for that group and adds a new link to the chain of PO EOD messages. The harvester does nothing if a PO EODStatus object already exists for the EOD date being processed or for a later date. Once the PO EOD message is written for a reconciled EOD marker an EODConfirmation message is written to confirm its arrival.

If the group is not reconciled but is being forced out the <NotReady:> attribute is added to the PO EOD message, which is chained to the <EODData.LastEODNotReady:> attribute in the PO EODStatus object, and a Not Ready object is created for that group. The Not Ready objects are processed at start up and at Circadian Transition (EOD time is reached). The reconciliation for each non-ready group is rechecked and, if then OK, the PO EODStatus object and and EOD message are updated. The <NotReady:> attribute is removed from the PO EOD message and the new message's chain transferred from the <EODData.LastEODNotReady:> attribute to the <LastEOD:> attribute.

The agent declares Circadian Transition when it receives an EODCutoffNow message for the current day (EODCutoffNow messages for other days are generally ignored) or it reaches its configured EOD time. The agent always checkpoints after the Circadian Transition. Its checkpoint name is "AGT_C_HV_EOD_CHKPT". After this checkpoint the agent writes an EOD Cluster Status object for its cluster. It then loops waiting for a complete set of EOD Cluster Status objects and then ensures that an EODStatus object is written. After Circadian Transition the new EOD cutoff time will be set for the next EOD day using the (reference data) configured time. Some checks are included during Circadian Transition to pick up unexpected problems (such as the resetting of machines dates or duplicate agents running):

a) The PO Status object is checked before each Circadian Transition so it is never done twice for the same EOD day.

b) The time difference between the Agent server and the Correspondence server is checked when a Circadian Transition is done. If the difference is greater than a configurable time an error is reported, the Circadian Transition is not done and the harvester will terminate.

FUJ00171949

2.9.2 Recovery requirements

The EOD harvester checks the PO Status objects at initialisation to decide whether we need to do a late Circadian Transition and what is the current harvesting day. The existence of a PO Status Object is used to decide if Circadian Transition has happened or not. This may cause us to repeat the Circadian Transition but if we do it is going to be harmless, as the bulk harvesters do not know of the previous attempt.

If the agent has started before the current day's configured cut-off time the first check is whether the Circadian Transition has been done for the previous EOD day.

a) If not, Circadian Transition is completed for the previous day and the new cut-off time is set for today's end of EOD day.
b) If it is, we check whether the Circadian Transition has been done for the current EOD day and then set the new cut-off time to either today's end of EOD day (does not exist) or tomorrow's end of EOD day (does exist).

If the agent has started on or after the current day's configured cut-off time, the check is whether the Circadian Transition has already been done for the current EOD day and if not the Circadian Transition is completed for the current day. In either case the new cut off time is set for tomorrow's end of EOD day

Other aspects of recovery are the same as for any other traditional interactive harvester.

2.9.3 Sizing & Performance

Same as for any other interactive harvester.

2.9.4 Control Interfaces

This agent runs as an NT service with one service instance per cluster. The service name is defined as:

TMSEndOfDay<n>

where '<n>' is the cluster id, omitted for cluster 1.

Thus the service names for two instances of the agent processing clusters 1 and 2 respectively will be **TMSEndOfDay** and **TMSEndOfDay2**.

The underlying executable is:

C_HV_EOD.exe

2.9.5 Maestro schedule

N/A. This agent runs all the time under Tivoli control. However an independent Maestro schedule decides when to call the 'harvest cut-off time'.

2.9.6 Registry Entries Used

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

The registry entry values utilised are listed in [GEN]. The following registry entry values of particular interest are reproduced here for the convenience of the reader:

FUJ00171949

Value Name	Type	Max Size	Opt?	Usage
VIRTUALOFFICE	DWORD		N	The group id of the virtual office.
PMEXPIRYTIME	DWORD		Y	When, in days, a Riposte message will expire and become subject to archiving. Default value is 0. When zero, the message expiry time will be set according to the Riposte configuration parameter DefaultMessageExpiry.[4]
RIPOSTE	SZ		N	locale
CLUSTERID	DWORD		N	1 to 4
SYNCHRONIZATION_DELAY	DWORD		Y	Delay (in milliseconds) to wait for synchronisation from counter. Default 10 seconds
SUCCESS_UNIT_COUNT	DWORD		Y	The count of messages processed before the agent writes a checkpoint.[5]
CLOSECHECKINTERVAL	DWORD		Y	Timeout parameter passed to RiposteReadMessagePort(). Used as the frequency at which the agent will inspect other control events (e.g. the shutdown event) and to check on late replication.[6]
IGNOREOLDERTHAN	DWORD		Y	EODs more than this number of days old are ignored. Default 30 days. This is supported for migration when the complete history of EODs could be harvested.
MAX_TIME_DIFFERENCE	DWORD		N	The maximum time difference, in milliseconds, allowed between an Agent's and Correspondence Server's clocks. Default is 7200000.

Note EODFINISHTIME is no longer supported, the reference data object supersedes it.

2.9.7 Oracle Table Format

Not Applicable. This agent has no access to a database.

2.9.8 Riposte Message Format

2.9.8.1 EOD Cluster Status object

An ***EOD Cluster Status object*** is created for each cluster, by the EOD harvester running in that cluster, when the EOD cut off time for that cluster has been reached (i.e. that cluster has completed Circadian Transition). It resides in the Virtual Post Office (in practice, 999998) and records the EOD date that is now ready for harvest.

[4] PMEXPIRYTIME is set to 35 to match counter expiry
[5] SUCCESS_UNIT_COUNT is set to 1000000 for this agent.
[6] CLOSECHECKINTERVAL is set to 15000 (15 seconds) for this agent.

FUJ00171949


```
<Collection:EODAgent>
<ObjectName:StatusCluster%d>   // %d is the Cluster number
<EODData:
   <EODDate:dd-Mon-yyyy>       // EOD date
>
<Version:version_info>
```

2.9.8.2 EOD Status object

There is just one instance of the ***EOD Status object***. It resides in the Virtual Post Office (in practice, 999998). It is created and maintained by the EOD Harvester.

The EOD Status object is the means by which the EOD Harvesters collectively inform the bulk harvesters that the EOD ***harvest cut-off time*** has been reached, i.e. that all the clusters have recorded that the day is Complete.

```
<Collection:EODAgent>
<ObjectName:EODStatus>
<EODData:
   <EODDate:dd-Mon-yyyy>       // The EOD date
   <Date:dd-Mon-yyyy>          // The EOD date, but its use is deprecated
   <AllComplete:1>             // The value 1 is the only value defined
>
<Version:version_info>
```

The existence of the EOD Status object for a particular EODData.Date implies that all previous days are necessarily All Complete.

2.9.8.3 PO EOD Status object

There is one ***PO EOD Status object*** for each real Post Office. It resides in the Virtual Post Office (in practice, 999998) and is maintained by the EOD Harvester when it reads an EOD Harvest Trailer for that office through its message port. It points at the last PO EOD Message for that Post Office in a chain of such messages and may also point to a 'not ready' EOD message if the group is not reconciled for the date held in EODData.Date.

FUJ00171949


```
<Collection:EODAgent>
<ObjectName:group_id>          // Padded to 6 digits with leading zeroes
<EODData:
   <Date:dd-Mon-yyyy>          // EODDate from EOD Marker message
   <LastEOD:                   // Pointer to last PO EOD Message for the PO
      <GroupID:group_id>
      <Id:node_id>
      <Num:message_num>
   >
   <LastEODNotReady:           // Optional, pointer to PO EOD Message not yet
                                  processed
      <GroupID:group_id>
      <Id:node_id>
      <Num:message_num>
   >
>
<Version:version_info>
```

The code has EODData.Date set to the value of the 'not ready' EOD Marker when the LastEODNotReady attribute is set. This means it is no longer consistent with the LastEOD pointer. I don't think this is correct because the bulk harvesters (In GetEODMarker routine) use the Date and LastEOD attributes and assume these to be consistent.

2.9.8.4 PO EOD message

The virtual office contains a chain of ***PO EOD messages*** for each real Post Office, one message for each EOD detected by the EOD Harvester[7]. It points to both the Post Office's EOD Marker message and the corresponding Harvest Trailer message.

[7] Occasionally there will be a second i.e. when the first includes the <NotReady:> attribute.

FUJ00171949


```
<Application:EODAgent>
<Data:
   <TranType:POEODMessage>
   <EODDate:dd-Mon-yyyy>          // EODDate from EOD Marker message
   <EODTime:hh:mm:ss>             // EODTime from EOD Marker message
   <ReceivedDate:dd-Mon-yyyy>     // Date harvester receives EOD message
   <EODMessage:                   // Pointer to EOD Marker message
      <GroupID:group_id>
      <Id:node_id>
      <Num:message_num>
   >
   <EODTrailer:                   // Pointer to Harvest Trailer message
      <GroupID:group_id>
      <Id:node_id>
      <Num:message_num>
      <Date:dd-Mon-yyyy>          // Date attribute from harvest trailer message
      <Time:hh:mm:ss>             // Time attribute from harvest trailer message
   >
   <NotReady:>                    // Included with a value Y only if replication
                                  incomplete at time of writing
   <Previous:                     // Pointer to previous PO EOD message or 'not
                                  ready' PO EOD message; empty if first such
                                  message
      <GroupID:group_id>
      <Id:node_id>
      <Num:message_num>
   >
>
```

2.9.8.5 EOD Not Ready object

A Not Ready object is created when the EOD Marker for a group is processed but the group is not reconciled. These objects are used to identify the groups that must be re-checked (when EOD time has been reached) for reconciliation. When the group is found to be reconciled the object is deleted.

```
<Collection:EODAgentNotReady>
<ObjectName:group_id>
<Reconciled:0>
<Version:version_info>
```

2.9.8.6 EODAgent Object

Reference Data that specifies the latest finish time for that day's EODs. At this time it can consider the set of today's EODs to be complete and start harvesting tomorrow's.

```
<Collection:EODAgent>
<ObjectName:Parameters>
<EODFinishTime:23:30>
```

2.9.8.7 EODCutoffNow Message

The existence of this message is a request to start Circadian Transition for the specified CutOffDate.

FUJ00171949


```
<Application:EODAgent>
<Data:
   <TranType:EODCutOffNow>
   <CutOffDate:dd-Mon-yyyy>     // Date to apply the cut off time
>
<Version:version_info>
```

2.9.8.8 EODConfirmation Message

Written whenever an EOD harvester queues a PO EOD message for processing by the bulk harvesters. It is used to inform the Post Master that the APS details have arrived (since the APS transactions contain details on customers payments to clients).

```
<Application:EODAgent>
<Data:
   <TranType:EODConfirmation>
   <EODDate:dd-Mon-yyyy>              // EOD date from harvest trailer
   <HarvestedDate:dd-Mon-yyyy>// Date EOD marker was harvested
>
<Version:version_info>
```

2.9.9 Riposte Message Mapping

Not applicable.

FUJ00171949

2.10 Counter Call Scheduler (C_HV_POSCH)

2.10.1 Functional Description

The Counter Call Scheduler is a service that runs on the gateway counter to provide intelligent management of the Riposte connections between the gateway and the Correspondence Servers. This management is co-ordinated with the WAN network management (to the data centre) provided by the CNIM service. The overall management is particularly geared to the Fujitsu Services P.O.A. requirements and has 4 major advantages over the default Riposte UBI (Unconnected Broadcast Interval) basis.

1) It can take advantage of any fixed WAN network connection (provided by CNIM) by nailing up the Riposte connection for the whole of that network connection
2) It reduces the likelihood of peaks in the number of WAN and Riposte connections over short intervals and hence improves the chance of a successful first time connection. This reduces the chance of failing an SLA e.g. on responding to OBCS foreigns.
3) It reduces the number of connections required, whilst it still maintains a steady flow of messages
4) It gives priority to any SLA critical data (e.g. end of day data).

In addition this agent

a) detects outlets where counters are being switched off at night and generally monitors and reports on the connections between the gateway and other counters (LAN network);
b) provides support to the counters for an on-line status indicator e.g. forwards status information on the status of the WAN network.

The agent divides the functionality between 2 processing threads. One thread manages this agent's scheduling of the Riposte connections according to any threshold and time requirements (in such cases a Riposte connection results in a WAN connection). The other thread monitors the CNIM for changes in the WAN network connections and responds to such changes. Interaction between the two threads is intended to be minimal. The CNIM thread indicates when the WAN network is bad and hence there is no point in the Scheduler thread attempting a connection. The Scheduler thread maintains information on the times that the Riposte connections are last started and last completed (allows CNIM thread to detect a lost 'fixed' connection). It also raises a flag if it decides not to attempt a connection because the WAN network is bad. This flag causes a connection by the CNIM thread once the network recovers.

2.10.1.1 Scheduling Thread

When required, this thread requests a Riposte connection by writing a Flush Message as a priority message. It reports on the status of the LAN network (its ability to communicate with the other counters) with either a Counter Connection Status Object or Counter Connection Status Message.

The scheduling thread is a listening thread (traditional harvester thread) and is configured from reference data (see General Configuration Details). This configuration includes the setting up of the listening thread's selection criteria for any Riposte messages that are considered to have some influence on the scheduling (see Message Selection Configuration Details).

Processing is caused either by the arrival of a message or by a timeout on a polling period. This processing evaluates the current situation and what the impact of the current message (if any) is to be; it then decides whether a connection is required. A connection is requested by writing a priority message (flush message). It can be required for any of the following reasons

1) time instigated i.e. connections are required at particular times during the day
2) a maximum time has expired since there has been data to replicate
3) the data to be replicated has exceeded a count threshold and it has been more than the minimum allowed time since the last connection

FUJ00171949

4) an urgent message needs to be sent

Once a connection is attempted it may be periodically retried until the connection succeeds. There are 2 ways in which a connection can fail and cause a retry

a) the priority message does not result in a connection, or
b) a connection is started (not necessarily one requested by this agent) but does not complete (some data is not replicated).

However no connection is attempted if CNIM has already indicated that it cannot connect to the WAN and so there should be little use of these retry flush messages. In such cases the flush message is still written (documentary) but not as a priority message (indicated by an 'X' appended to the reason). Some messages may delay a connection so that any associated messages are handled by a single connection.

The service will not use a checkpoint but will always start up using 'current' for this thread.

Connections to the other counters are monitored. Details on any changes in the counter connection states are reported to the data centre so that the support staff can be informed of problems. The agent can also be configured to raise a memo to inform the Post Office staff of any recently failed connection between the gateway and another counter. The option for raising a memo is switched on and off via the reference data (see Memo Control Details and Memo Configuration Details) and is completely independent of the rest of the connection scheduling. Any particular problem should only be reported once, however another counter failing may result in a second memo.

2.10.1.1.1 Time-based Connections

The facilities provided for the time-based connections allow a sequence of requests for connections at particular times ('AfterTime') optionally followed by further periodic connections ('Then.Every') until an end time ('UpToTime'). There must be no overlap of times in the sequence (any ' UpToTime ' should not be later than the next 'AfterTime'), which should be ordered in time order throughout the day.

In practice time-based connections are not required at an exact time but within some time band. Any connection within this time band satisfies the time-based connection. The configuration allows the start and end (again randomised) of the time band to be specified. If a connection happens between the start and end of the time band there is no need to make an explicit time-based connection. However if there is no connection within the time band then an explicit connection is requested at the end.

The end of the time band is randomised over an interval to prevent all Post Offices trying to connect at exactly the same time. Even if a connection has not occurred by the expected connection time, it will be suppressed if the office is closed and an 'only if open' connection is required.

Where a periodic connection is required it is normally timed from the completion of a connection (requested or not). However it starts from the expected connection time (for an 'after time' or previous periodic) when the connection is suppressed because it is closed. Should there be a connection for any reason then the time for the next periodic is reset. Resetting stops once the new time exceeds the 'UpToTime'.

Time-based connections are not affected by any lack of data to replicate.

2.10.1.1.2 Threshold-based Connections

Threshold connections are caused either by passing a maximum time interval since there has been data to replicate or by too many messages waiting to be replicated. However, to prevent a period of high input causing too many connections, a minimum time interval is also imposed. The threshold set

FUJ00171949

for the maximum message count must be sufficient to ensure full use of the time given with the minimum charge for an ISDN connection.

All 3 of these thresholds are supplied as a lower and upper range so that the value used can be randomised each use. This is to prevent peaks through similar profile Post Offices all making their connections at the same time.

Different sets of threshold are supported for use whilst an office is open from when it is closed. A threshold that is set whilst an office is closed can be immediately reset if it stretches into the open hours. In these circumstances, the threshold value is reduced if it exceeds the open threshold time that would be set if evaluated from the changeover time.

When there is a long wait before there is any data to replicate then the connection for the Riposte UnconnectedBroadcastInterval may beat this threshold based connection.

2.10.1.1.3 Message Type Controls

Filters can be set up for selecting messages of particular types to allow them to have special control over the subsequent connection.

Some messages are subject to SLAs and need timely replication to the Correspondence Server. For these messages, the threshold for the current maximum time of the next connection can be checked and where necessary brought forward. In this case the message type details provide a lower and upper range for the maximum interval permitted to the next connection. The range is used to determine a randomised maximum time for the next connection and if this is less than the current threshold it replaces the current threshold. It is randomised to prevent any peaking of the connections.

Some messages expect to be followed by more messages and so there is no point making too early a connection. Both the minimum and maximum time thresholds are checked for being outside of the minimum interval required by the message. Where they are not they are incremented by this minimum interval to ensure they are. The values are incremented to retain some randomisation.

The check for incrementing by the minimum interval precedes the check for reducing the maximum time and both can be set within the same message type. This seems completely irrational but can be useful when reducing the maximum time. Setting a minimum interval greater than the maximum interval can ensure that the maximum time is always reset for a message of that type, hence giving all a random value across the supplied maximum interval range. Without specifying the high minimum interval, some would not be reset and there would be more connections requested at the lower end of the range than at the upper end. Some would even occur earlier.

FUJ00171949

2.10.1.1.4 Counter Connection Monitoring

Any counter connection problems and their subsequent clearance must be reported to the centre in a timely manner. The relevant states are good (all counters are on line) or bad (at least one counter is inaccessible) and hence the **critical** changes are when

a) the first counter fails
b) all counters are back on line again.

Changes in the number of failed counters, i.e. between these 2 states, are still reported on, but at a lower priority since the outlet is considered bad irrespective of how many or which counters have failed.

No special action is taken on a failure until it has existed for at least 5 minutes (configurable). This avoids reporting on any temporary comms glitches or counters that are immediately rebooted. However, whenever any failure is being reported the current status of all counters is included in that report (hence it may include some that have only just failed).

The new CountersStatus message content is introduced for reporting on changes in the connection status. This is not a message but a particular set of attributes that can be included in any message though, in practice, they are only included in 3 message types. The content is identified by a "<Data.Subtype:CountersStatus>" attribute value. A change in the connection state is always reported via a CountersStatus object (as this makes it available on a restart). A change in the number of failed counters is reported via a CountersStatus message.

After a restart the current status is reported at the first opportunity so as to confirm the current status to the centre. Normally this is by including the CountersStatus information in the first flush message However if the service has not had a reason to write a CountersStatus or flush message by 11.00 a.m. (configurable) it writes a CountersStatus message anyway.

CountersStatus messages (that are written for reporting a change in the set of offline counters) are non-critical and will never be written within 15 minutes (configurable) of the previous CountersStatus message/object.

Writing a CounterStatus object/message reports on a change of status but does not necessarily mean that the information is available at the centre. Where this information needs to get to the centre in a timely manner then a flush may be required. Such a flush is always delayed until as late as possible in case the replication can piggyback on another ISDN connection or in case the status reverts and the urgency is removed. A CountersStatus message is never considered important. When the counter's connection state changes to bad, the objective is to get the resulting CountersStatus object to the centre within 2 hours[8] (configurable). However this urgency may be cancelled if the state changes back to good before the centre knows of the bad state. Once the centre knows of a bad state there is some urgency with getting the centre to know of a change to a good state. This should either be within 2 hours of the bad state or, if too late for this, in 15 minutes time (configurable).

There is an option to report counter failures to the Post Office staff so the problem can be cleared. A MemoView message is raised to report the failure and this is based on the same details as the CounterStatus. A MemoView is not reported on a restart if the status had been bad when the service stopped as it is assumed that it would have been reported earlier.

[8] Currently unnecessary as the UBI is 2 hours. However this design should not place constraints on any future configuration changes.

FUJ00171949

2.10.1.2 The CNIM monitoring thread

When required, this thread requests a Riposte connection by writing a NailUp Message as a priority message. It reports on the current WAN network status in a Network Status object. It may also logically delete the objects indicating that a clerk has been informed that the WAN or LAN is bad once it detects the problem has been cleared (see On-line Status Support section).

The CNIM thread is polled regularly to check for changes in the WAN connection status. It can be tuned via QOS Configuration reference data.

The CNIM monitoring thread interfaces with the CNIM service, via CNIM_Get_Status (see [CNIM]), in order to be informed whenever there is any change to the network connection. There is a short timeout on these calls (5 seconds, configurable) to allow other work to take place periodically. This thread is responsible for nailing-up Riposte connections during 'fixed' connection periods. These 'fixed' connection periods are either periods that Fujitsu Services is contracted to provide a fixed connection or for outlets where the workload is such that a fixed connection is deemed desirable. If the CNIM_Get_Status interface is not available, or if it returns a status of unknown, this thread makes its own interpretation of the WAN network status. It does this via Riposte, writing a priority NailUp message to attempt a connection and then using the Scheduler thread's information to check when Riposte makes contact with the data centre.

This thread will cause Riposte connections for any of the reasons in the following table. Where more than one entry applies the connection will assume the longest nail-up period.

Trigger	Reason
Fixed	Establish the Riposte connection whenever the CNIM connection has been nailed up and a fixed connection is required. This may be a reconnection after a failure. It is nailed up for 3 seconds (configurable) less than the network connection has been nailed up for. These may be either a 'long-duration nail-up' or a 'short-duration nail-up'.
LostFixed	Re-establish the Riposte connection whenever the Riposte connection is lost during a successful CNIM 'fixed' connection. The Riposte connection is nailed-up again for the outstanding fixed period. These may be either a 'long-duration nail-up' or a 'short-duration nail-up'.
LateFlush	Establish the Riposte connection whenever the connect-required flag is set. This flag is set when the Riposte monitoring thread wanted to write a priority Flush message but couldn't because the network connection was unavailable. The Riposte connection is established as soon as the network connection succeeds. These are always treated as a 'short-duration nail-up'.
PostFixed	When the network connection fails during a fixed period and does not come back up until after the end of that fixed period then the Riposte connection should be re-established once the network connection succeeds. There may be some important messages to be replicated from the fixed period. These are always treated as a 'short-duration nail-up'.
Emergency	Establish the Riposte connection if CNIM makes an emergency connection over the voice line. These are always treated as a 'Short-duration nail-up'. Such connections should be very rare and there is no point in trying to be too sophisticated in such cases.
WakeUp	If a network connection is unavailable for too long then there is an attempt to establish the Riposte connection just in case it may wake up the CNIM. These are always treated as a 'short-duration nail-up'.
CNIMUnknown	The call to CNIM_Get_Status has either failed or it has returned a status of CNIM_GS_UNKNOWN. If this persists, a 'short-duration nail-up' is attempted to determine the actual network status. This is not a genuine nail-up but an attempt to check the network status via Riposte.

FUJ00171949

2.10.1.2.1 On-line Status Support

The CCS is to be used in an ancillary role of making network connection information available to the counters (via the on-line status library). This allows the desktop to be better informed when there are problems and hence keeping the clerks informed.

1) It maintains the WAN Network Status object with the latest change in availability of the WAN network. This update is triggered when CNIM_Get_Status indicates that there is a change to the status of the WAN network connection. There are 3 primary states of 'OK', 'Cut' and 'Bad'. 'OK' covers the states of CNIM_GS_CONNECTED and CNIM_GS_DISCONNECTED. 'Cut' is used to indicate that CNIM has unexpectedly lost a connection. 'Bad' indicates that a connection attempt has failed (this implies that CNIM has failed on all the current line numbers) and this is further subdivided to allow some tailoring of the message to the clerk. An emergency connection by the CNIM is ignored as far as this object is concerned. If the CNIM service is unavailable, the CCS will make its own assessment of the WAN network based on information from the Riposte service.

2) It clears the global flags on the reported network problems when the problem no longer exists. There is one flag for any local counter LAN problem (being able to see this flag means the problem is cleared) and another for the WAN connection. This processing is triggered on every return from CNIM_Get_Status (including timeouts). The two flags are maintained in separate objects (the WAN Reported Bad and the LAN Reported Bad objects). They are logically rather than physically deleted when the flags are cleared.

2.10.2 Control Interfaces

This runs as an NT service on the counters. Although it exists and is started on all counters the service will only continue running on the gateway counter. The executable is identified as follows:

C_HV_POSCH.exe

The service name is TMSPOCallSched.

2.10.2.1 Maestro Schedule

Not applicable. This service is to be stopped and started in co-ordination with the Riposte service (e.g. by the cleardesk program).

2.10.3 Registry Entries Used

The Registry key for this agent will be C_HV_POSCH but no registry configuration is required to run this agent. In general all configuration of this agent will be from reference data.

Details of all registry keys are contained in [GEN]. The following tables give further usage information for registry values that can be supplied in the agent configuration to override the default values.

Value Name	Value	Comment
TRACELEVEL	DWORD	Support only
RIPOSTE	SZ	For testing only this may be supplied as \RiposteService where RiposteService is the name of a Riposte service. This facility is implemented but not unit tested.
PERFMONFILE	EXPAND_SZ	If a filename is supplied then performance information is written

FUJ00171949

		to the file

The reference data always overrides any registry setting for CLOSECHECKINTERVAL.

The reference data consists of a set of objects in the POCallScheduler Collection. There will be

1) one object with general configuration details
2) a separate object for the details on each message type to be processed
3) one object with control over which memos are to be reported (only offline counters supported)
4) a separate object for any memo that may be reported

2.10.4 Riposte Message Format

All of the following reference data definitions are specified as for non-temporal reference data. However all the reference data may be supplied as temporal if required.

2.10.4.1 General Configuration Details

The message format for the reference data supplied for the general configuration is defined as:

FUJ00171949


```
<Collection:POCallScheduler>
<ObjectName:General>
<Data:
   <DataType:General>
   <PollInterval:<Lower:><Upper:>>
   <ConnectIsLateAfter:>
   <RetryConnectionInterval:>
   <MaxConnectionRetries:>
   <ThresholdConnects:
      <MaxMsgCount:<Lower:><Upper:>>
      <MinInterval:<Lower:><Upper:>>
      <MaxInterval:<Lower:><Upper:>>
      <NotOpenOverrides:
         <MaxMsgCount:<Lower:><Upper:>>
         <MinInterval:<Lower:><Upper:>>
         <MaxInterval:<Lower:><Upper:>>
      >
   >
   <TimedConnects:
      <1:
         <AfterTime:>
         <ConnectDelay:<Lower:><Upper:>>
         <OnlyIfOpen:>
         <Then:
            <Every:<Lower:><Upper:>>
            <UptoTime:>
         >
      >
      ...
   >
   <CounterStatus:
      <FailDelay:>
      <FailMaxFlushInterval:>
      <WhileBadFailDelay:>
      <FailToGoodMaxInterval:>
      <LateToGoodMaxInterval:>
      <ConfirmStatusTime:>
   >
>
```

Field	Range Requ'd	Description
PollInterval	Yes	Range for randomising the polling interval (in milliseconds). This is used in place of the CLOSECHECKINTERVAL registry value and defaults to 10 seconds.
ConnectIsLateAfter		An instigated connect is late if it does not take place within this interval (in milliseconds). Default 5 minutes. Late connects are recorded in the next flush message.
RetryConnectionInterval		A failed connection request should be retried after this interval (in milliseconds). Default 20 minutes.
MaxConnectionRetries		The maximum number of retries for a failed connection request. There is no point in RetryConnectionInterval * MaxConnectionRetries exceeding the Riposte UnconnectedBroadcastInterval.
ThresholdConnects		contains values for controlling threshold connections
MaxMsgCount	Yes	Range for randomising the threshold at which the number of outstanding messages can cause an early connection (subject to being after MinInterval). The number should be sufficient to always utilise the minimum ISDN connection

FUJ00171949

		time.
MinInterval	Yes	Range for randomising the earliest time the next connection will be instigated by the agent (in milliseconds). MinInterval should be sufficient to prevent any overloading of the routers at peak periods through having too many connections.
MaxInterval	Yes	Range for randomising the latest time by which the next connection is required (in milliseconds).
NotOpenOverrides		if supplied it includes overrides to be used for the MaxMsgCount, MaxInterval and MinInterval values when outside of opening hours
TimedConnects		if supplied it includes details of connections required daily (sequence is supplied in time order thoughout the day from 00:00 to 23:59)
AfterTime		Start of connection time band. Check after this time to see if a connection is required (local time in hh:mm)
ConnectDelay	Yes	Range for randomising the end of the connection time band (delay in milliseconds). (First) Connection is to be forced at AfterTime + ConnectDelay unless a connection happens after AfterTime. The range should be sufficient to allow all the Outlets to have at least a minimum length connection, with spare capacity, when one site is out together with one router at the other site.
OnlyIfOpen		Y or N (default N). If Y, the AfterTime or periodic connection is to be suppressed if the office is closed. Next periodic is timed from the time the connection is suppressed.
Then		Further connections are required periodically. It always starts timing from the last connect
Every	Yes	Time between each connection (in milliseconds)
UptoTime		Time to finish these periodic connections (local time in hh:mm)
FailDelay		Time allowed before a connection is deemed to have failed (in milliseconds) when all counters were considered to be good. Default 5 minute.
FailMaxFlushInterval		if failed connections are outstanding for this interval they must have been reported up to the CS (in milliseconds). Default 2 hours.
WhileBadFailDelay		Time allowed before a connection is deemed to have failed (in milliseconds) when there is already a bad counter. Default 15 minute.
FailToGoodMaxInterval		Once a failure is reported to the CS any change to good must be reported to the CS within this interval from the original failure unless it is already too late (in milliseconds). Default 2 hours.
LateToGoodMaxInterval		When the change to good is too late for the FailToGoodMaxInterval it must be reported to the CS within this interval of it changing to good (in milliseconds). Default 15 minutes.
ConfirmStatusTime		Latest time to write first Connection Status details (local time in hh:mm). Default 11.00.

Note: where an entry allows 'Range Required' the attribute value should be supplied as <Lower:*lowint*><Upper:*highint*> to allow random values to be generated in the range lowint to highint. A single integer is also supported, in which case the value is a constant.

2.10.4.2 Message Selection Configuration Details

The message format for the reference data supplied for both configuring the selection and action required for a type of message. It is defined as:

FUJ00171949


```
<Collection:POCallScheduler>
<ObjectName:>
<Data:
    <DataType:Msg>
    <Selection:
         <1:
              <AttrName:>
              <AttrValue:>
         >
         …
         <IgnoreIf:
              <Op:>
               <1:
                   <AttrName:>
                   <AttrValue:>
              >
              …
         >
    >
    <MsgMinInterval:>
    <MsgMaxInterval:<Lower:><Upper:>>
>
```

Where an attribute is of the form <1:…>… it implies an ordered list, starting with 1, that is terminated immediately the next digit is missing.

The Selection is based on the reference data used by the acknowledgement agents

Field	Range Required	Description
ObjectName		Uniquely identifies this configuration item. For this type of message (identified by <DataType:Msg>) the name should start with "Msg"
Selection		Identifies the selection criteria used to harvest messages from the message port. Values from the sequence 1,2... are concatenated in the form "*<1.AttrName:1.AttrValue><2.AttrName:2.AttrValue>...*" , where italics indicate the attributes value. It is specified this way to avoid invalid attribute grammar in the object.
AttrName		An attribute name as used for selection criteria e.g "Data.TranType"
AttrValue		The value as used for selection criteria e.g. "CashStatement"
IgnoreIf		Allows some harvested messages to be ignored base on its content.
Op		AND (message contains all named attributes with the given values) or OR (message contains at least one named attribute with its given value), default AND
MsgMinInterval		On receiving messages of this type this specifies a minimum time before a connection should be instigated (unless overridden by a later message). If specified in the same object as MsgMaxInterval, MsgMinInterval is applied and then superseded by the MsgMaxInterval change (may be used to always reset MaxInterval with MsgMaxInterval)
MsgMaxInterval	Yes	On receiving messages of this type these attributes specify the maximum time (randomised between range) before a connection is instigated (unless

FUJ00171949

		overridden by a later message).

2.10.4.3 Memo Control Details

The message format for the reference data supplied to control the reporting of memos is defined as:

```
   <Collection:POCallScheduler>
   <ObjectName:MemoControl>
   <Data:
<DataType:MemoControl>
      <NodeOfflineMemo:
         <MemoObjectName:>
         <DelayReport:>
         <OnceDaily:>
>
   >
```

Field	Range Required	Description
NodeOfflineMemo		Details for controlling the counters offline memo
MemoObjectName		The ObjectName of the object containing details for writing a memo when the gateway has a disconnected counter. No memo is reported if this attribute is omitted.
DelayReport		Minimum time a counter has to be disconnected before raising the memo (in milliseconds). Default is report after 5 minutes.
OnceDaily		Y or N. Y if a failed connection is to be reported at most once a day (unless service restarted). N if to be reported for each disconnection. Default N

2.10.4.4 Memo Configuration Details

The message format for the reference data supplied for configuring a MemoView message is defined as:

```
<Collection:POCallScheduler>
<ObjectName:>
<Data:
   <DataType:Memo>
   <Text:>
   <InsertIf:<1:><Many:>>
   <Caption:>
   <UserGroup:>
   <Brand:>
   <Priority:>
   <MessageID:>
   <Metric:>
   <ViewDelay:>
>
```

FUJ00171949

Field	Description
ObjectName	Uniquely identifies this configuration item. For this type of message (identified by <DataType:Memo>) the name should start with "Memo"
Text	Should start with "From# *Brand*||" where || is interpreted as new line and contain 2 %s entries for substitution. Will substitute the 1st %s with details on the time the memo's raised (e.g. "Monday 6th November at 12.01") and the second %s with details on the failing nodes (e.g. "NodeId of 3"). The maximum size of the (expanded) text in a MemoView message is documented as 1600 in [MSGSUB]. However in this case the size is reduced by the inclusion of the acknowledgement details in the message. The actual constraint is imposed by the maximum size of 2048 for a Riposte Message, which not only applies to the MemoView message but also to the delivery of this Memo Configuration Details message. The maximum size of this expanded message should therefore be restricted to be under 1300.
InsertIf	Optional. Includes text to substitute in the 2nd %s of Text in place of the list of NodeIds.
InsertIf.1	text to insert if there is 1 failed counter.
InsertIf.Many	text to insert if there are many failed counters. This text may include a %d for the number of failed counters.
Caption	18 characters maximum (alphanumerics and space allowed)
UserGroup	Allowed UserGroup values are: MANAGERS Management Users SUPERVISORS Supervisor Users CLERK Counter Clerk Users AUDITOR Auditor Users AUDITOR E Emergency Manager MIGRATE Migration Users Any All Users will be able to view the Memo Default Any (Any being all of the above)
Brand	Default "CustServ"
Priority	"High" or "Low". Default "High"
MessageID	Should include %s, which will be substituted by node, group, date and time. This is the MessageID for the MemoView message. It is not used as the AckDat.MessageId value that is required by the SLA harvester when Metric is specified. Default derived as "%s" appended to Caption.
Metric	Used for SLA harvesting. See section 2.11. The AckData attribute is omitted from the MemoView message when not supplied.
ViewDelay	Used to set up the required time for viewing (MDate and MTime) in the MemoView message's AckData attribute. Supplied in milliseconds. Default 6 days, maximum 20 days.

2.10.4.5 Counter Connection Status Message

This message is written for reporting non-critical counter connection status changes (i.e. changes in the number of unconnected counters).

FUJ00171949


```
<Application:POCallSched>
<Data:
   <TranType:CounterStatus>
   <Subtype:CounterStatus>
   <POStatus:OK|Bad>
   <StatusChangedAt:<Date:><Time:>>
   <UnconnectedNodes:
      <NodeId:<Node:NodeId>>
      ...
   >
   <MissingNodes:
      <NodeId:<Node:NodeId>>
      ...
   >
   <NeighbourCount:>
>
```

where the CounterStatus attributes are as follows:

Field	Description
Subtype	Provided to allow a single selection criteria for CounterStatus and relevant flush messages
POStatus	OK or Bad. Only set to Bad when UnconnectedNodes is included
StatusChangedAt	If POStatus is Bad it is the date and Time the POStatus last changed to this state. When OK, the change may have been long before this returned date and time but is never later. In particular it is the agent's restart time if the outlet has never returned a CounterStatus with a Bad POStatus.
UnconnectedNodes	only supplied if the gateway has a connection problem with one or more of the other counters (also including the mirror). It is a list of Ids for the supported nodes that are unconnected. The grammar allows additional attributes to be added in the future. If there is a problem with the mirror node it will be reported here, but it is never included in the NeighbourCount.
MissingNodes	only supplied if the counters do not form a contiguous set of nodes. It is a list of Ids for the unsupported nodes that are skipped when arriving at NeighbourCount. Any with node ids above the last real counter are not included. The grammar is consistent with UnconnectedNodes. These nodes are not included in the NeighbourCount.
NeighbourCount	Count of the gateway's neighbour counters, but excluding any mirror node (node 31)

2.10.4.6 Counter Connection Status Object

This message is written for reporting critical counter connection status changes (i.e. changes to and from all counters connected).

```
<Collection:C_HV_POSCH>
<ObjectName:CounterStatus>
<Application:POCallSched>
<Data:
   <TranType:CounterStatus>
   <Subtype:CounterStatus>
   <POStatus:OK|Bad>
   <StatusChangedAt:
      <Date:>
      <Time:>
   >
```

FUJ00171949


```
    <UnconnectedNodes:
        <NodeId:<Node:NodeId>>
        ...
    >
    <MissingNodes:
        <NodeId:<Node:NodeId>>
        ...
    >
    <NeighbourCount:>
>
```

Attribute details are as for the Counter Connection Status Message

2.10.4.7 Flush Message

This is written as a priority message to initiate a connection. It has an expiry of 1 day (which is less than the minimum counter expiry and so will be overridden) and has the following content:

```
<Application:CallFlush>
<Data:
    <Subtype:CounterStatus>      // only included if status is to be harvested
    <POStatus:OK|Bad>
    <StatusChangedAt:            // only included if status is to be harvested
        <Date:>
        <Time:>
    >
    <UnconnectedNodes:           // only included if unconnected counters
        <NodeId:<Node:NodeId>>
        ...
    >
    <MissingNodes:               // only included if counter nodes are not a dense
                                 set
        <NodeId:<Node:NodeId>>
        ...
    >
    <MissedFlush:                // only include when a flush fails
        <Requested:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
        <Connected:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
    >
    <CutConnection:              // only include when a connection fails to
                                 complete and then only <At:> is mandatory
        <At:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
        <Requested:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
        <Completed:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
    >
    <Statistics:
        <NeighbourCount:>
        <MsgCount:>
        <LastConnect:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
        <LastRequest:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
        <Trigger:>
    >
>
```

FUJ00171949

Field	Description
POStatus	"OK" or "Bad". Only set to Bad when UnconnectedNodes or MissedFlush is included
UnconnectedNodes	only supplied if the gateway has a connection problem with one or more of the other counters (also including the mirror). It is a list of Ids for the supported nodes that are unconnected. The grammar allows additional attributes to be added in the future. If there is a problem with the mirror node it will be reported here, but it is never included in the NeighbourCount.
MissedFlush	only supplied when the connection for the previous CallFlush message was late by over ConnectIsLateAfter milliseconds or this message is a retry of a flush (where there has been no connection). In the case of such a retry message the attribute is supplied as "<MissedFlush:>". Requested and Connected are supplied as UTC times.
CutConnection	only supplied after a connection has failed to complete replication. The resulting retry flush includes the "At" and optionally the "Requested" attributes. When the completed connection is late by over ConnectIsLateAfter milliseconds this attribute is included in the next CallFlush request, in which case "At" and "Completed" attributes are included. At, Requested and Completed are supplied as UTC times.
NeighbourCount	Count of the gateway's neighbour counters, but excluding any mirror node (node 31)
MsgCount	current count of messages to be replicated
LastConnect	UTC date and time of the last connection. Value omitted if no connections made since start of service.
LastRequest	UTC date and time of the last request for a connection (excluding any subsequent retry). Value omitted on first request
Trigger	Reason for writing flush message - One of "MaxInterval", "MaxMsgCount", "Urgent", "Time", "Periodic", "Status" or "Retry"*n* (where *n* is the current retry count).

and the remaining CounterStatus attributes are as described for the CounterStatus message.

2.10.4.8 MemoView Message

A MemoView message is optionally sent to warn Post Office staff that a counter is not connected. A memo is never sent for a problem with the mirror node (node 31). The details are set up from the Memo Configuration Details reference data. The message has an expiry of 5 days (which is overridden by the counter's minimum) and contents of

```
<Application:MemoView>
<Sender:C_HV_POSCH>
<MsgVersion:
   <Major:2+>
   <Minor:<C_HV_POSCH:PRODUCT_VERSION>>
>
<MessageID:>
<Data:
   <Text:>
   <Caption:>
   <UserGroup:>
   <Brand:>
   <Priority:>
>
<AckData:
   <TranType:AckRequest>
   <Metric:>
   <MessageId:>
   <MDate:>
   <MTime:>
```

FUJ00171949

>

Field	Description
MessageID	Taken from reference data. As required for MemoView message
Data	The rest of the standard MemoView message data as required by Escher
Text	Taken from reference data. As required for MemoView message
Caption	Taken from reference data. As required for MemoView message
UserGroup	Taken from reference data. As required for MemoView message
Brand	Taken from reference data. As required for MemoView message
Priority	Taken from reference data. As required for MemoView message
AckData	This is only included if a Metric is supplied in the Memo Configuration Details. It is set up as standard SLA data to be acknowledged by all counters and to be harvested by the SLA harvester (Metric is rejected until post M1). This counter acknowledgement from the failed counter indicates when it comes back on-line. It does not indicate that the memo has been viewed, which is indicated by a special MemoView message written from Escher code. Further development is required in the SLA harvester to be able to harvest the message acknowledging a memo has been viewed. See SLA Harvester (W_HV_SLA) section 2.11.
AckData.Metric	A new metric (suggest "MEMO").
AckData.MessageId	Not the same as MessageID but a concatenation of group, node and num (must not exceed 20 characters)
AckData.MDate & MTime	Derive as being ViewData from now. These are not real SLA targets but may be useful for deciding if a Post Office is slow at reading these memos.

Note : the message written when a MemoView message is viewed does not include any of our SLA type data in it. It does however include a reference to the above MemoView message; so the SLA harvester can be enhanced to access these AckRequest details when deriving the response details

2.10.4.9 QOS Configuration

This object is optional. It allows some of the configuration values for the CNIM thread to be tuned. Its format is defined as:

FUJ00171949


```
<Collection:POCallScheduler>
<ObjectName:QOS>
<Data:
   <DataType:QOS>
   <MinNailUpPeriod:>
   <ReduceNailUpBy:>
   <CNIMTimeoutInterval:>
   <WakeCNIMInterval:>
   <NoCNIM:
      <CheckAllowance:>
      <FirstCheckDelay:>
      <FromTempCheckDelay:>
      <FromPermCheckDelay:>
      <OKCheckDelay:>
      <Report:
         <InfoDelay:>
         <AlertDelay:>
         <RepeatDelay:>
      >
   >
>
```

Field	Description
MinNailUpPeriod	If the nail-up is for longer than this period (in ms) then the supplied nail-up is used to provide an absolute period for the Riposte connection, otherwise MinNailUpPeriod is the maximum time for the Riposte connection. Default 1 minute.
ReduceNailUpBy	the provided absolute period for a Riposte connection is the nail-up time less ReduceNailUpBy. It is supplied in ms and defaults to 3 seconds.
CNIMTimeoutInterval	Timeout supplied to CNIM_Get_Status calls. It is supplied in ms and defaults to 5 seconds.
WakeCNIMInterval	period in msec between attempts to wake up the network when the network is unavailable for a long time. Default 2 hours.
NoCNIM	Details to control this agent making its own checks on network availability (via Riposte) when the CNIM is unavailable.
CheckAllowance	seconds allowed for connection to happen following priority message used to check if we can connect (trigger of "CNIMUnknown"). Default 60 seconds.
FirstCheckDelay	delay in seconds before issuing first priority message. Default 15 minutes.
FromTempCheckDelay	delay in seconds after setting to temporarily unavailable before issuing second priority message. Default 15 minutes.
FromPermCheckDelay	delay in seconds before issuing subsequent priority messages to confirm permanently unavailable. Default 1 hour.
OKCheckDelay	delay in seconds since last connection before issuing a priority message to confirm still available. Default 2 hours.
Report	Various delays (in seconds) before reporting the CNIM as unavailable. The initial information (default to after 15 minutes) is for diagnostics as Tivoli manages the CNIM service. It is eventually (default of 1 hour 45 minutes later) raised as an alert to ensure some attention and this is repeated periodically (default every 4 hours).

2.10.4.10 NailUp Message

This is written as a priority message to cause a Riposte connection at times when CNIM has established a network connection. There are 2 styles of this nail-up depending on the nail-up period requested.

FUJ00171949

a) 'Short-duration nail-up'. Those for up to a 'minimum nail-up time' (1 minute configurable via MinNailUpPeriod) are nailed-up with the 'minimum nail-up time' as the maximum connection time i.e. they will only stay open as long as necessary and will also restrict any outstanding nailing up to the 'minimum nail-up time'. Effectively they are connections that are not assumed to be nailed-up. Note that there are some existing priority messages that nail-up a connection but these are less than 1 minute. We do not want to cut these connections prematurely.
b) 'Long-duration nail-up'. Those for more than the 'minimum nail-up time' are nailed-up for 3 seconds (configurable) short of the network connection's nail-up time. This is an absolute time (i.e. as a replacement connection time on the Riposte call). The small reduction is an attempt to close the Riposte Connection before the network wants to close down.

```
<Application:POCallSched>
<Data:
   <TranType:CCSNailUp>
   <Statistics:
      <NailUpPeriod:>
      <LastDisconnect:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
      <Trigger:>
   >
>
```

Field	Description
NailUpPeriod	Nailed-up time in msecs. It is 0 for short-duration nail-ups.
LastDisconnect	Date and time of disconnect. Only included if a reconnection after a failure
Trigger	Trigger value corresponding to the reason for writing this message (as taken from the table of reasons).

2.10.4.11 Network Status

This message provides information on the WAN network status to the counters for maintaining the on-line status indicator. Its format is defined as:

```
<Collection:AgentNetwork>
<ObjectName:WANStatus>
<Application:OnlineStatus>
<Data:
   <NetworkState:>
   <From:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
   <Silver:Y>
   <Failure:
      <StatusNow:>
      <FailCode:>
      <PermAt:<Date:dd-Mon-YYYY><Time:hh:mm:ss>>
   >
>
where
```

FUJ00171949

Field	Description
NetworkState	OK or Bad or Cut. Only included as "Bad" when CNIM returns a connection status of CNIM_GS_FAILED_ONCE, CNIM_GS_TEMPORARY or CNIM_GS_PERMANENT.
From	Date and time of last change of NetworkState.
Silver	Only included if it is a contracted silver outlet.
Failure	Only included if NetworkStatus is Bad
StatusNow	One of AGT_NW_FAILED_ONCE, AGT_NW_TEMPORARY or AGT_NW_PERMANENT.
FailCode	The failure to be reported to the helpdesk.
PermAt	Date and time that the CNIM expects to change a CNIM_GS_TEMPORARY status to CNIM_GS_PERMANENT

2.10.5 Scheduler Thread Processing Algorithm

The key values used in the algorithm (fields with time in their name are date and time values)

Item	Random	Description
StartDelay	Y (a)	Randomises the start of processing within the average for PollInterval
PollInterval	Y (a)	The time between checks if no relevant message is received in between (10 second randomised by up to + or -1 seconds at start).
ConnectIsLateAfter		Period used to determine if the next flush message should include details of a failed connection to the CS. Default 5 minutes.
ThresholdConnectByTime		Latest time for forcing a connection. 0 when not active
EarliestConnectTime		Earliest time for forcing a connection when MaxMsgCount is exceeded. 0 when threshold is not active
MaxMsgCount	Y (b)	Maximum desirable level of unsent messages. Connection can be forced earlier once this threshold is exceeded. The number should be sufficient to always utilise the minimum ISDN connection time.
PreviousConnectionTime		Time (in seconds) at or just after the last connection
StartThresholdTime		Time at which data became available to replicate since the last connection
StartOpenTime		Time to start using the open hours configuration values
EndOpenTime		Time to start using the NotOpen hours configuration values
(Threshold)MaxInterval	Y (b)	time from StartThresholdTime for deriving ThresholdConnectByTime.
(Threshold)MinInterval	Y (b)	time from l StartThresholdTime for deriving EarliestConnectTime
MsgMinInterval		On receiving particular messages, if either ThresholdConnectByTime or EarliestConnectTime fall within the MsgMinInterval period they should be extended by MsgMinInterval (this is message type specific)
MsgMaxInterval	Y (c)	On receiving particular messages, if ThresholdConnectByTime falls outside of the MsgMaxInterval period it should be reset to MsgMaxInterval (this is message type specific and randomised for that message type)
OutstandingConnectFlag		The priority message has been written but connection is yet to happen.
ForceConnectionFlag		force a connection immediately even though there may be no messages to send
TimedConnectRqdAfterTime		Time at which we need to start checks on an (absolute) time based connection (i.e. start of time band)
TimedConnectForceByTime	Y(d)	Time associated with TimedConnectRqdAfterTime by which connection is to be forced (i.e. end of time band for time-based connection)
CurrentEvery	Y(d)	Interval for setting up the next periodic time-based connection
CurrentUpToTime		Time for stopping periodic time-based connection
RetryConnectionTime		Time for making a retry after a failed connection request

FUJ00171949

a) randomised once at startup
b) randomised every time ThresholdConnectByTime is being set up after a previous connection (but not when being modified for a message)
c) randomised for every message when it is applicable.
d) randomised whenever TimedConnectRqdAfterTime set up

2.10.5.1 Startup

1) Configuration details are read from reference data
2) Replication will be assumed to be to the current marker if it is unavailable from Riposte.
3) wait for a period of StartDelay to randomise polling.
4) Initialise current time-based from sequence 1 (it will be corrected during running if we've passed it)
5) read the CounterStatus object for details on the status at closedown

2.10.5.2 Processing details

The scheduling is triggered either by receiving a message on a message port or by a timeout on the read message port call.

The main processing after a timeout or reading a message is the same and consists of:-

Check on the connection status for the other counters
and possibly raise a MemoView memo for offline counters

Get Riposte's details on its last connection to the
Correspondence Server and potentially update connection information
(check current replication status)

If the gateway is currently connected to the CS we need to
wait until it's finished replicating

Wait until Riposte has successfully completed our last request
for a connection, but retry if it's outstanding too long.

A forced connection may be required now (time based connection or retry)
If not then we
a) return if there is no data to replicate
b) process the harvested message (if any) to adjust thresholds (**process message type)**

Instigate a connection if required (**request connection when required)**

2.10.5.2.1 check current replication status

Call RiposteGetNodeStatusEx to get
a count of messages that are not replicated
the number of milliseconds since the last connection and
whether a connection is in progress at the time of the call

Need to read the time before and after the call of RiposteGetNodeStatusEx as we are returned a number of milliseconds since last connection *so* at least we know it is somewhere between 2 times. However being a few seconds out should not be too critical as long as it always errs the right side in each comparison.

FUJ00171949

Clear the urgency to flush a CounterStatus object if it is replicated

Make a best guess on values to use if Riposte still doesn't know yet

Write a CounterStatus message/object if required

If a connection is in progress return to wait until it has finished

If it's a new connection we need to update the PreviousConnectionTime, force a retry if the connection finished prematurely (and return), clear the flag that says there's an outstanding connection request and if the connection's late set up the MissedFlush attribute

If no messages outstanding set an indicator to say no thresholds in use

Check for change from open to closed or vice versa and switch the threshold set if necessary (add 24 hours to the replaced threshold's start time). This has to be before the time-based checks to support the OnlyIfOpen option

Return if Riposte has not completed our last request for a connection

If the current time-based details are active (we are within its time band) check if some action is required. This may be

1) if we've just had a new connection update the current time-based details to the next time
2) if we've reached the time-based expected connection time we normally set the ForceConnectionFlag (to cause a connection) and return. The exception to this is where the OnlyIfOpen is set and the office is closed when the action is to update the current time-based details to the next time

If a CounterStatus object needs flushing set the ForceConnectionFlag and reason then return

With no messages outstanding we have nothing to do (even if a message was harvested) so return

Thresholds need to be updated (**reset connection thresholds**) if this is the first time there has been data to replicate since the last connection

2.10.5.2.2 reset connection thresholds

We have just received data to be replicated since a new connection.

Set up the next set of threshold values (HighMsgCount , ThresholdConnectByTime and EarliestConnectTime) for controlling a dynamically scheduled connection.

Calculating these values just before the opening time uses the set for a not open office. Need to check if this is too late for the first expected connection after opening. Check (and possibly reset) both the ThresholdConnectByTime and EarliestConnectTime and if either/both change then reset HighMsgCount.

2.10.5.2.3 process message type

Check if the message should be ignored and return if it should

FUJ00171949

Get the MsgMinInterval and MsgMaxInterval details for the message type and set up randomised values. If both are set up the minimum is applied before the maximum.

if MsgMinInterval is set (message expects to be followed by other message) increment (by MsgMinInterval) either of ThresholdConnectByTime or EarliestConnectTime if they are within MsgMinInterval.

if MsgMaxInterval is set (urgent messages need to sent to the CS quickly) set ThresholdConnectByTime to the minimum of ThresholdConnectByTime and MsgMaxInterval from now (EarliestConnectTime is irrelevant)

2.10.5.1.4 request connection when required

A connection is required if either

ForceConnectionFlag is set i.e. a time-based, status or retry connection is required
ThresholdConnectByTime reached
HighMsgCount exceeded and we've reached the EarliestConnectTime

A connection is requested by writing a flush message, which involves:-

Write a priority message to force a connect (includes UnconnectedNodes, MissingNodes and MissedFlush attributes + Statistics)

Set OutstandingConnectFlag to indicate that a connection request is outstanding (retain time of attempt) and set up a retry time

clear buffer for the MissedFlush attribute, unset the ForceConnectionFlag etc. and clear threshold times.

2.10.5.1.5 Assumptions

When there is no data to replicate or a connection request is already outstanding there is no point in processing any harvested message.

a) If it is classed as an urgent message it has either been sent already or will be as soon as Riposte can make a connection.
b) If it is a message to delay a connection then it's too late, but the MsgMinInterval delay is probably not going to be more than the ThresholdMinInterval eventually used and is certainly less than ThresholdMaxInterval.

Occasionally being late for initiating a connection by one or two poll periods is quite acceptable.

2.10.6 Reference Data

This section contains an overview of the current reference data required for this agent on live.

The agent also accesses the existing reference data for the offices opening hours. The object for the opening hours is temporal in the Collection "OpeningPeriods" with the ObjectName of "OfficeHours".

The agent's private configuration data is to be handed over in a set of .nrd and associated .nrg files. In all cases the .nrg files will be for delivery to the dummy office (999995) so the data is forward delivered to all Post Offices. The data can be delivered as temporal data, in which case the <Data:...> attribute is supplied as <RData:<Data:...>>

The following set of Reference data is to be delivered.

FUJ00171949

Filename	ObjectName	Description
POCallSchedGen	General	General configuration data. Open thresholds: maximum time after 1st message 50 to 70 min; exceeds 400 to 600 messages, but not within 15 to 20 min. Closed thresholds: maximum time after 1st message 80 to 100 min; exceeds 900 to 1100 messages, but not within 15 to 20 min. Timed connects: between 06:59 and 07:29
POCallSchedEOD	MsgEOD	Configuration data for EOD messages i.e.. <EODMark.TranType:EODMark><Application: EOD> Potentially delay a connection for an hour (until the Harvest Trailer message is available).
POCallSchedHarvTrlr	MsgHarvestTrailer	Configuration for harvest trailer messages i.e. <EODMark.TranType:HarvestTrailer><Application:EODHT> Connects between 10 and 40 minutes (the delay occasionally picks up LFS messages).
POCallSchedLFSCash	MsgLFSCash	Configuration for LFS Cash Statement messages i.e. <Data.TranType:CashStatement><Application:LFS> Connects between 10 and 40 minutes (the delay picks up further LFS CashStatement and StockStatement messages[9]).
POCallSchedMemoCtrl	MemoControl	Switches on the option to report unconnected counters via a MemoView message.
POCallSchedMemoCO	MemoCounterOffline	Defines the content of the MemoView message to be reported when counters are offline.
Not required as defaults are acceptable	QOS	Overrides the default configuration of the CNIM monitoring.

2.11 SLA Harvester (W_HV_SLA)

The SLA Harvester now harvests to the OMDB. It has been renamed M_HV_SLA and is described in [OMDBAGT].

[9] The StockStatement messages are not all written at the same time. Those written at 00.01 have no SLA requirement and are replicated with the reference data. Those written at 19.10 do have an associated SLA requirement, but will be handled by the delayed connection for the CashStatement.