[ TITLE * MERGEFORMAT ]

[ SUBJECT * MERGEFORMAT ]

**Document Title:** Low Level Design For the DCS Authroisation Agent

**Document Type:** Low Level Design

**Release:** Release Independent

**Abstract:** [ COMMENTS * MERGEFORMAT ]

**Document Status:** DRAFT

**Author & Dept:** Anne Mohan – POA

Contributors: John Rayner

**Internal Distribution:** Andy Williams

**External Distribution:** (Specify those individuals outside of the Post Office Account who require approved version only. For Document Management to distribute following approval)

## Approval Authorities:

| Name | Role | Signature | Date |
|---|---|---|---|
| Andy Williams | HLD Author | | |

*Note: See Post Office Account HNG-X Reviewers/Approvers Role Matrix (PGM/DCM/ION/0001) for guidance.*

# 0 Document Control

## 0.1 Table of Contents

**[ TOC \O "1-3" \H \Z \T "POA APPENDIX HEADING 1,1,POA APPENDIX HEADING 2,2" ]**

## 0.2 Figures

[ TOC \h \z \c "Figure" ]

## 0.3 Document History

| Version No. | Date | Summary of Changes and Reason for Issue | Associated Change - CP/PEAK/PPRR Reference |
|---|---|---|---|
| 0.1 | 15/2/2007 | First Review | CP4143 |
| 0.2 | 13/03/2007 | Comments from review (some outstanding) | |
| 0.3 | 22/03/2007 | Changes for PCI (changes marked are from v0.1) | CP4305 |
| 0.4 | 07/04/2008 | Reflect the implemented Registry defaults. This affects sections [ REF _Ref195322453 \r \h ]. Note Heartbeat thread information is in common with Banking Agents and is in [ REF NBSLLD \h ]. That document has also been updated against this PEAK. This docment revision was circulated for review. | PEAK 151378 |
| 0.5 | 03/10/2008 | Apply comments from review of version 0.3 | |
| 0.6 | | Apply comments received.<br>Syntax of FI_RESPCD_MAP documented (section [ REF _Ref238031389 \r \h ])<br>Command thread documentation added (section [ REF _Ref238031331 \r \h ])<br>Clarified A1_Info contents (section [ REF _Ref72737898 \r \h ])<br>Changes as a result of post-implementation review by author | PEAK 169062 |

## 0.4 Review Details

| | |
|---|---|
| Review Comments by : | 04/09/2009 |
| Review Comments to : | Anne Mohan & [ HYPERLINK "mailto:PostOfficeAccountDocumentManagemen GRO |
| Mandatory Review | |
| Role | Name |
| Solutions Designer | Andy Williams* |
| System Test | John Rogers* |
| SSC | Mik Peach* |
| | |
| Optional Review | |
| Role | Name |
| Security and Risk Team | [ HYPERLINK GRO GRO |
| System Qualities Architect | Dave Chapman |
| Architect | Jason Clark |

| | |
|---|---|
| Business Continuity | Adam Parker |
| Service Support | Kirsty Gallacher |
| HNG-X Service Transition | Graham Welsh |
| Service Network | Ian Mills |
| Data Centre Migration | Geoff Butts, Peter Okely |
| SV&I Manager | Sheila Bamber |
| Tester | Hamish Munro |
| RV Manager | James Brett (POL) |
| VI & TE Manager | Mark Ascott |
| Integrity Testing | Alan Child, Michael Welch |
| Development | Garham Allen |
| Network Architect (Data Centre) | Mark Jarosz |
| Issued for Information – Please restrict this distribution list to a minimum | |
| Position/Role | Name |
| | |
| | |

( * ) = Reviewers that returned comments

## 0.5 Associated Documents (Internal & External)

| Reference | Version | Date | Title | Source |
|---|---|---|---|---|
| PGM/DCM/TEM/0001 (DO NOT REMOVE) | 1.0 | 13/6/06 | Fujitsu Services Post Office Account HNG-X Document Template | Dimensions |
| DES/APP/HLD/0006 [GENHLD] | | | Generic Authorisation Services High Level Design | Dimensions |
| DES/APP/HLD/0007 [DCSHLD] | | | DCS Authorisation Agents High Level Design | Dimensions |
| DEV/APP/LLD/0017 [NBSLLD] | | | NBS Authorisation Agents Low Level Design | Dimensions |
| DES/SYM/HLD/0045 [TIDHLD] | | | MID/TID Allocation Service High Level Design | Dimensions |
| DES/APP/IFS/0006 [MSGIFS] | | | RAC Message Flows in HNG-X | Dimensions |
| EF/IFS/002 [SSEAIS] | | | Horizon-Streamline Application Interface Specification | PVCS |
| DES/APP/SPG/002 [OPOVER] | | | HNG-X Agents Operational Overview | Dimensions |
| DES/APP/IFS/0009 [NBXJNL] | | | Specification of NBX Journal Records for Online Authorisation Services | Dimensions |
| NB/DES/006 [TESELEM] | | | Transaction Enquiry Service (TES) Elements Specification | PVCS |

| Reference | Version | Date | Title | Source |
|---|---|---|---|---|
| AD/DES/039 [AGTGEN] | | | Generic Agent Components for CSR+ High Level Design | PVCS |
| NB/IFS/035 [BUSPARAMS] | | | NBX Business Parameters | PVCS |
| DES/APP/HLD/0017 [NPSHLD] | | | Network Persistent Store (NPS) High Level Design | Dimensions |
| DES/SEC/IFS/0001 [PCICRY] | | | HNG-X Cryptographic Applications Programming Interface Specification | Dimensions |
| DEV/APP/LLD/0023 [XMLLLD] | | | XML Message Handling and Agent Migration Guide | Dimensions |

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

## 0.6 Abbreviations

| Abbreviation | Definition |
|---|---|
| DCS | Debit (and Credit) Card System |
| DRS | Data Reconciliation Service. The Horizon's Network banking database |
| FI_EE | FI Enquiry Engine. In a DCS context this means the SSE. |
| H(PAN) | Hashed PAN. |
| MA | Merchant Acquirer. The MA communicates with the appropriate Card Issuers (Banks and other card providers) |
| MID | Merchant ID (externally allocated value identifying the Outlet) |
| NBE | Network Banking Engine |
| NBS | Network Banking System |
| NBX | NBE Replacement |
| OSR | Online Service Router |
| PAN (or clear PAN) | Primary Account Number |
| (PAN)PK | An Encrypted PAN Block encrypted under some key using the triple DES algorithm. The (PAN)PK value includes identification of the key used for encryption. |

| Abbreviation | Definition |
|---|---|
| PCI | Payment Card Industry |
| PI | Process Interface |
| [R1] etc. | This is an abbreviation for a DCS request message as issued by the counter. Other [R*n*] abbreviations are used for the request at different stages on its way to the SSE. In particular an [R3] is the message sent to the SSE. |
| [A1] etc. | This is an abbreviation for a DCS authorisation message as issued by the SSE. Other [A*n*] abbreviations are used for each stage as the authorisation detail is passed back to the counter. |
| SAN | Structured Attribute Notation |
| SSE | This is used within the agent for Streamline (Streamline Service Engine) |
| Streamline | The Merchant Acquirer (MA) accessed by this agent. |
| TES | Transaction Enquiry Service (does not support DCS transactions) |
| TESQA | TES Query Application |
| TID | Terminal ID (externally allocated value identifying the Counter)<br>The Terminal Identity included in messages sent to Streamline |
| VA | Virtual address, an IP address and port combination. For DCS there is only one combination |
| X.25 | Standard network layer protocol for packet switched wide area network communication. |
| YDDD | Part of the primary key for NPS Status table, also part of the key for R1ToA3 hash table. Derived from <LclDte> value as year mod 10 + day of year i.e. 4002 from 02-Jan-2004 and 0028 from 28-Jan-2010. |

## 0.7 Glossary

| Term | Definition |
|---|---|
| Encrypted PAN Block | A PAN Block encrypted under some key using the triple DES algorithm. The (PAN)PK value includes identification of the key used for encryption. For DCS it contains the PAN and optionally the Expiry Date, the Start Date and the Issue Number of a card. |
| Hashed PAN | A substitute for a (clear) PAN used for display purposes. It is derived from a PAN by applying a one-way hashing algoritm H to the PAN. A hashed PAN always contains an alphabetic character. |
| SSE Handlers | The term used to describe connections to STREAMLINE. Each EE_IO thread represents an SSE Handler. Each SSE Handler may make multiple socket connections. This term is synonymous with PIs (used by NBS and ETS). |
| SYSMAN | The infrastructure components used for Systems Management |
| Trigger | A value associated with the bead and the message it contains which summarises the state of the bead/message at any point, and is used to control subsequent processing. |

## 0.8 Changes Expected

| Changes |
|---|
| None |

## 0.9 Accuracy

Fujitsu Services endeavours to ensure that the information contained in this document is correct but, whilst every effort is made to ensure the accuracy of such information, it accepts no liability for any loss (however caused) sustained as a result of any error or omission in the same.

## 0.10 Copyright

# 1 General

## 1.1 Introduction

This document provides the low-level design of the **DCS Authorisation Agent for HNG-X**. This agent provides the interface from the Branch Access layer (BAL) specifically the Online Service Router (OSR) to Streamline using the standard RAC (request, authorisation, and confirmation) protocol.

The following diagram is reproduced here from [ REF DCSHLD \h ] to assist in understanding the overall message flows.

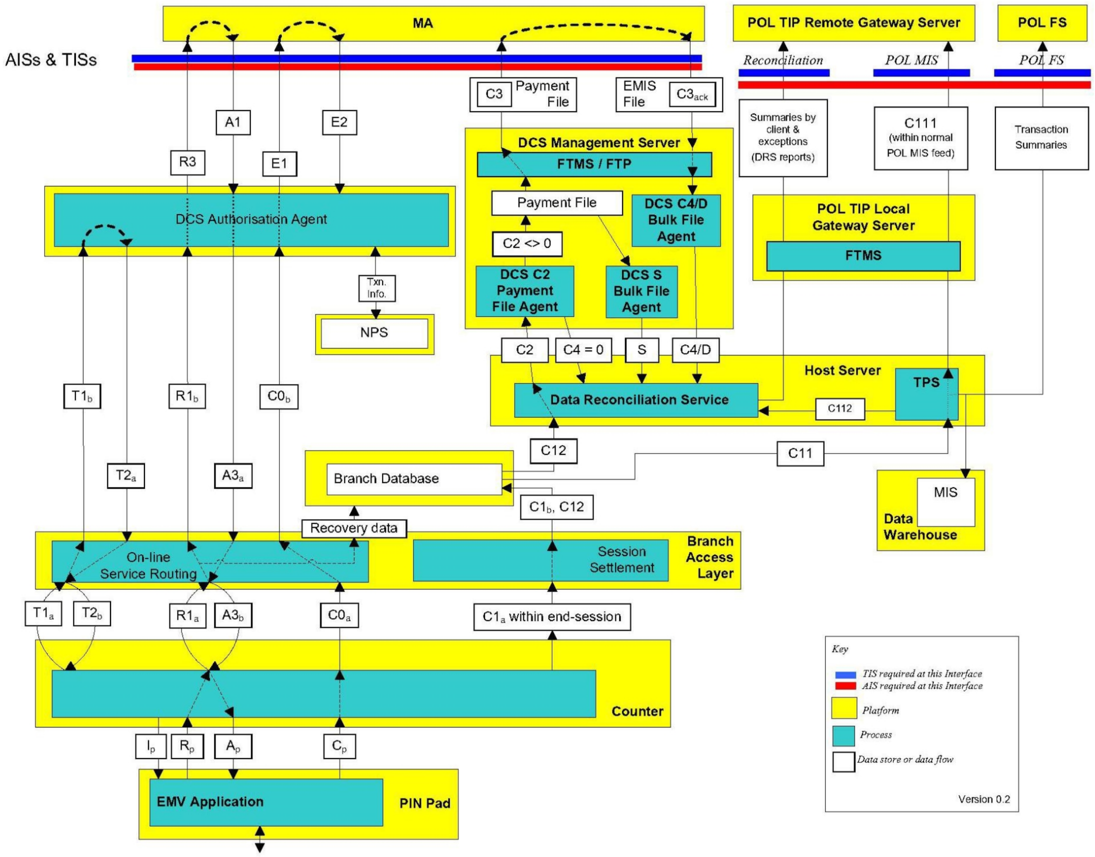

**Figure [ SEQ Figure * ARABIC ] - DCS Message Flows**

This agent is required to provide a very resilient system. It is based on the NBS Authorisation Agent described in [ REF NBSLLD \h ], which includes a description of the generic parts of the Authorisation Agent's low-level design for HNG-X.

The DCS Authorisation Agent supports Reversals as well as the [R] and [A] messages. However the reversals are not 'must deliver', unlike those associated with the NBS Authorisation Agent, and so can be handled in a less resilient manner.

### 1.1.1 Differences between Horizon and HNG-X Agents

The business functionality in the Horizon agent has been implemented in the HNG-X agent.

Two requirements have been implemented in a different way in the HNG-X agent:

i) The agent must not forward to Streamline the next message for a particular counter (TID) until the SSE has responded to the previous message or the response has been timed out. This is implemented by use of the HoldQ in the GetR1X thread, which retains [R1] until the response to the previous message has been received or until the response been has timed out. Section [ REF _Ref209432087 \w \h ] discusses the HoldQ. In Horizon, such an [R1] was put on a 'blocked' queue in the EE_IO thread if the response to the previous message was still outstanding.

ii) Each message sent to Streamline for a particular TID must be assigned a different Message Number from the previous message. This is implemented by storing a Message Number for each TID in a table in the NPS. The agent reads these numbers when it starts up and stores them in an extension to the internal table of MID/TID numbers that is also created when the agent starts up. Section [ REF _Ref210788512 \w \h ] discusses the MID/TID table and section [ REF _Ref161989025 \w \h ] discusses the Message Number. In Horizon, the Message Number was generated by the counter and passed to the agent in the <Data.RACMess.Id.TransNum:> attribute of the message.

The HNG-X requirement of PCI compliance has been added to the agent.

As already stated, the structure of the agent is based on the NBS Authorisation agent. The significant differences from Horizon are that it does not use Riposte and that it does use the NPS Oracle database. This document contains many references to [ REF NBSLLD \h ] as the functionality that is common to NBS and DCS is fully described in that document.

The risk of producing a new agent for HNG-X has been minimised by using the well-proven model of the NBS Authorisation agent.

## 1.2 Scope

This document provides the low-level design of the DCS Authorisation Agent for HNG-X which only processes HNG-X messages.

The starting point for the implementation of this agent is the NBS Authorisation agents and some functionality that is common to NBS and DCS is described fully in [ REF NBSLLD \h ]. This document makes a cross reference in all these cases.

## 1.3 Assumptions

# 2 THE DCS AUTHORISATION AGENT

## 2.1 Overview of the Design

### 2.1.1 Functional Overview

The DCS Authorisation Agent is tailored to interface with Streamline in order to authorise requests for debit and credit card transactions. The system provided by Streamline is referred to as the Streamline Service Engine (SSE). It is also referred to as FI_EE which is a generic term which can be used by other Authorisation agents.

There is a single logical DCS Authorisation Agent which communicates with Streamline, which in NBS terms is viewed as a single logical Financial Institution or FI.

However there will be active and standby run-time instances of the agent.

Although all access to Streamline is through a single TCP address and port number it is convenient to treat it as if it consisted of a number (4) of SSE Handlers, and have a separate agent thread (EE_IO described in section [ REF _Ref70417032 \w \h ]) controlling access to each. See section [ REF _Ref209330174 \w \h ] for the definition of SSE Handler.

The agent is capable of accessing SSE within each each SSE Handler via a number of TCP/IP virtual addresses (VAs) however this is not done and the VA count is set to 1.

The agent can be configured to open a number of sockets for each SSE Handler, and is required to load balance the [R3] across all its sockets and across all SSE Handlers.

The networks from the DCS Agent Servers are duplicated for resilience. This is invisible to the agent but there may be a delay of up to 20 seconds (normally it is much less) for the network to reconfigure itself after a failure. The agent must allow for this delay before deciding that it has lost contact with the FI_EE.

The supported business transactions are authorisations and reversals, and the agent also supports transaction status enquiries for recovery purposes.

Like the NBS Authorisation agents the DCS Authorisation agent uses an Oracle database, referred to as the **Network Persistent Store** (**NPS**), to provide a highly resilient persistent storage for a number of different purposes e.g.

a) A Transaction Journal (or audit log) of all interactions with the FI_EE and, in some cases, non-interactions (e.g. a badly formatted R1 or C0). These are used for long term audit, but are not currently fed on to the TES database.

b) A record of the current Status of each transaction. This is used for the normal operation of the agent.

c) Configuration information of business parameters. The DCS Authorisation agent has many less such parameters than the NBS agents.

d) Communication between the active and standby agents (Heartbeat table).

e) A history of the agent's status changes (Heartbeat History table), harvested by SYSMAN.

f) Statistics, harvested by SYSMAN.

g) Management Journal of all management interactions, harvested by SYSMAN.

h) External commands (Operarator Commands table).

The NPS is a resilient database with access via two Oracle instances or nodes. The Agents will retain connections to both nodes with one from each of the two Heartbeat threads.

This agent receives authorisation request messages from the OSR. The normal processing of these messages is as follows (see also [ REF DCSHLD \h ]):

a) Lookup the MID and TID for the counter in the MID/TID table, and use the entry to allocate an SSE Message Number for the [R1]

b) Reformat the data into an [R3] for Streamline

c) Insert the message's transaction status details and audit the [R3], both in NPS

d) Pass this [R3] to the SSE

e) Listen for the [A1] authorisation response from the SSE

f) Validate the FI_EE's [A1] and evaluate the response to be returned on the [A3]

g) Update the message's transaction status details and audit the [A1]

h) Reformat the data as an [A3] (this includes SLA information)

i) Write this [A3] to the connection that the original [R1] was received on.

The agent times out any [R1] that takes too long within the SSE (see section [ REF _Ref209323338 \w \h ]). The timeout is audited, and the standard [A3] is replaced by the timeout response (which still includes SLA information).

This agent also processes the [C0] reversal request messages which are received from an OSR. Reversals are not 'must deliver' so no guarenteed route through the NPS is provided (as there is for ETS and NBS). The normal processing of these messages is as follows (see also [ REF DCSHLD \h ]):

a) Immediately return an "NRSP" control message to the OSR to indicate that other reply will be sent.

b) Lookup the TID for the counter in the MID/TID table

c) Discard the message if it is stale

d) Use the TID to allocated an SSE Message Number for the [C0]

e) Read the transaction's status details to check that the reversal is valid and retrieve essential values

f) Generate the [E1] for Streamline

g) Update the transaction's status details and audit the [E1]

h) Pass this [E1] to the SSE

i) Listen for the [E2] acceptance from the SSE

j) Update the transaction's status details and audit the [E2]

Any [E1] that takes too long within the SSE is timed out and the timeout is audited. There are no retries.

There are no '**network management messages**' for the SSE.

The agent also processes Transaction Status Request [T1] messages and returns the appropriate response [T2].

This agent needs to provide a good throughput of messages and a high availability. It is based on the HNG-X NBS Authorisation Agents, see [ REF NBSLLD \h ], but with a different construction of threads and queues.

The agent has a high availability. Much of the agent's resilience to failure is dependent on the external resilience provided by and to the platform it runs on (e.g. resilient LAN cards). The agent aims to exploit such resilience. The intention is that an agent detects a problem with an Oracle node (or LAN) fairly quickly and switches to the other connection within a matter of seconds. The service should start up and continue running when only one Oracle instance is available.

In addition to the active agent there is a standby agent waiting to take over if the active agent fails. The active agent monitors itself for various problems and stands down when it is not in a healthy state (e.g. loss of thread). It also stands down and terminates when it cannot provide any service at all due to a lack of critical external resources – connections to the FI_EE do <u>not</u> count as a critical resource. However, unlike the Horizon version of the DCS Authorisation Agent, it does not stand down when it is unable to re-establish a lost connection with the SSE, nor does it negotiate with the standby agent to see if the latter has better connectivity.

The agent supports a limited amount of remote management. In particular it allows the dynamic setting of TRACELEVEL and TRACECALLS whilst running (see [ REF OPOVER \h ]).

### 2.1.2 Control Interfaces

The DCS Authorisation Agent runs as a Windows service. A pair of matching services, with different service names is run on different servers for resilience. Permissions can be set up as desired. The service names are:

**TMSNX_DCS_<*s*>**

where <***s***> is either B or W to distinguish the services and so identify the server.

> *The service name is intentionally succinct as it is a prefix on every event message and in every Transaction Status record.*

The underlying executable is:

**NX_NQ_DCS.exe**

### 2.1.3 Management of the Agent

This agent is launched and re-launched by Tivoli.

The agent provides Systems management information externally via two routes:

a) message to the event log;

b) information to SYSMAN via NPS.

The relevant messages in the event logs consist of two types.

a) direct reporting on the status of the service or on its access to a resource (see [ REF _Ref78290324 \r \p \h * MERGEFORMAT ]);

b) a message reporting a specific problem (either as an error or a warning).

External management of the agent is limited to:

a) tracetune to dynamically amend the tracelevel;

b) a limited set of operator commands via the NPS.

#### 2.1.3.1 Resource Management

These messages will have an event number in the range of 8000 to 10000 and include a MONID for the resource and a MONSEV for the severity (one of G for good, B for bad, W for warning and I for information). The following monitoring of resources is provided.

| MONID | MONSEV | Additional text |
|---|---|---|
| *short_service_name* | G | Service is available. |
| *short_service_name* | B | Service is closing. *reason* |
| *short_service_name* | I | Service is active. |
| *short_service_name* | I | Service is standby |
| | | |
| *short_service_name.SSEn* | G | *SSEn* is available. |
| *short_service_name.SSEn* | B | *SSEn* is unavailable. *reason.* |
| | | |
| *short_service_name*.CRYPTO | G | Crypto facilities are available. |
| *short_service_name*.CRYPTO | B | Crypto facilities are unavailable. *Reason.* |
| | | |
| *short_service_name*.DB*n* | G | *NPS_instance_n* is available. |
| *short_service_name*.DB*n* | B | *NPS_instance_n* is unavailable. *reason* |
| | | |
| *short_service_name*.CLOCK | G | Clock drift from NPS's clock is OK. |
| *short_service_name*.CLOCK | B | Clock drift is above threshold. |

**Table [ SEQ Table * ARABIC ]: MONID events**

*short_service_name* is the NT service name without its TMS prefix.

*reason* is a reason for the failure.

*SSEn* (e.g. SSE1) is the name (in upper case) of a target SSE Handler configured for the agent.

*NPS_instance_n* (*n* = 1 or 2) is the name of the TNSNAMES entry. Note that DB1 refers to the preferred Oracle instance, DB2 to the non-preferred instance. Only one of DB1 or DB2 needs to be available for the service to be able to run successfully.

## 2.1.4 Configuration by Registry

The Authorisation Agent uses the Windows Registry for configuration data, inparticular data that needs to be accessed before a connection is made to the NPS.

One specific principle is that any configuration item that may take different values for different Agent instances serving the same FI must be in registry. Examples are the Agent's Priority, which differs between the Agent services running on the primary and secondary servers, or the list of Oracle Instance names that are normally in a different order for different Authorisation agent instances.

### 2.1.4.1 Registry Entries Used

The DCS Authorisation Agent uses a key with the name

HKEY_LOCAL_MACHINE\SOFTWARE\ICL\PathwayAgents\NX_NQ_DCS

Different instances of the same agent are configured with unique values by providing values at a lower level using the hierarchy as above plus service name i.e. NX_NQ_DCS***service_name***.

Worked examples of the registry entries for the DCS Authorisation Agent are given in [ REF OPOVER \h ].

In the following %***x*** indicates a parameter that is substituted by the Agent at run-time. Note that it is not to be substituted by PIT at build-time.

Some of the entries below use the SAN syntax introduced for HNG-X and described in [ REF GENHLD \h ].

| Value Name | Default Value[1] | Comment |
|---|---|---|
| CLOSECHECKINTERVAL | 15000 (15 seconds) | Timeout parameter used as the frequency at which the agent will inspect control events e.g. the shutdown event. |
| CONNECTPOLLTIME | 60000 (1 minute) | Delay in milliseconds before retrying an attempt to connect to a resource (database, crypto service) |
| DBLOCATION(*) | None<br>**Explicitly configured in registry** (but see (*) at bottom of table) | Comma-separated list of two names in TNSNAMES file for connecting to NPS, in the preferred order, e.g. NPS2,NPS1. |
| ECHOTEST_DOWN | 30000 (30 seconds) | Timeout interval, in milliseconds, for connection to FI and OSR. Also interval in which agent must receive an RSTS control message after initial connection to OSR, after which connection to the port is deemed lost. |

[1] The default values shown are all set in the code and are not expected to be overridden in the registry for the initial HNG-X solution. Exceptions to this, or where there is no default value, are stated explicitly.

| Value Name | Default Value[1] | Comment |
|---|---|---|
| FI_RESPCD_MAP | {Data={1="ArC=00";1_1="Set:=01";2="ArC=05";2_1=AM~RETAIN CARD;2_1_1="Set:=02";3=AM~BAD FORMAT;3_1="Set:=69";4="Set:=08"}} | Structured information to map Response Code and Message fields in the [A1]/[E2] to a value for <RespCd> in the [A3]. Note that the syntax of the value is described in [ REF _Ref213228805 \w \h ]. |
| HEARTBEAT | These are as for NBS Authorisation<br>See [ REF NBSLLD \h ]<br>**Explicitly configured in registry** {Priority=1} for primary service and {Priority=2} for secondary | These are as for the NBS Authorisation<br>See [ REF NBSLLD \h ] |
| IDENTITY | DCS | Identity of the Agent in terms of the FI_EE it services. This value is used for the substitution variable %I in some other registry entries. |
| INFILE | C:\AgentData\DCS\Fad2Mid_Cnt2_Tid.dat | Name of the file providing the counter to Mid/Tid mappings, both for General Sales and Bureau de Change transactions. If the file is of type '.dat' and the file is read successfully then a file of type '.ack' (with the same name) is written to indicate that success. |
| LISTENS | GetR1X={Port=;MaxCn=50}<br>**Explicitly configured in registry** GetR1X={Port=TMSNX_%I_CL5} | **GetR1X** is for listening for the Online Service Router. **Port**: the set of Service names resolved in the Services file. The name string includes %***x*** that will be substituted with:<br>%I: Identity of the Agent<br>[There is no **Host** parameter as the computer_name is always used.]<br>**MaxCn**: maximum number of concurrent connections accepted, also used for the "backlog" value on the listen. |
| PASSWORD(*) | None<br>**Explicitly configured in registry** (but see (*) at bottom of table) | The password for the Oracle username |
| PERFMONFILE | None<br>**Explicitly configured in registry** C:\AgentStats\NX_NQ_DCS_Perf.mon | Name of memory mapped file. |
| QCONTROL | {TxnQ={HashKeyCt=1500;CSecCt=32};<br>FreeQ={Malloc=50;Cache=5;AgtLimit=4000}} | **HashKeyCt**: size of Hash Table, for fast access to entries.<br>**CSecCt**: number of critical sections for Hash Table to reduce possible contention.<br>**Malloc**: number of R1ToA3 entries to create at a time. |

| Value Name | Default Value[1] | Comment |
|---|---|---|
| | | **Cache**: free R1ToA3 entries are received and returned in batches of this size.<br>**AgtLimit**: agent closes if any of the underlying free queues exceeds its portion of this concurrency of R1ToA3 entries. |
| RECORD_PRESERVATION | 1 | This is to turn on/off the X.25 Record Boundary Preservation for Data Communication. The default is 1, i.e. Boundary Preservation is on. |
| REFERRAL_PHONE | 08457600510 | Phone number to be added into [A3] message when the Response Code indicates a Referral. |
| ReportFailDelay | 10000 (10 second) | Delay in milliseconds after the first attempt to connect to a resource (database, crypto service) before the first time a failure is reported. |
| ReportFailPeriod | 600000 (10 minutes) | Minimum period in milliseconds between reporting failures, unless the reason for failure has changed. |
| SOCKET_CLOSE_ALLOWANCE | Default 2000 (2 seconds) | Time in ms to allow a socket to close before attempting to open a replacement. |
| SOCKET_HOST | None<br>**Explicitly configured in registry**<br>STREAMLINE | For Agent-initiated connections, the Host name resolved by Active Directory to map to the TCP/IP address of the SSE Handler. |
| SOCKET_SERVICE | None<br>**Explicitly configured in registry**<br>dcs_port | For Agent-initiated connections, the Service name resolved in the Services file (c:\winnt\system32\drivers\etc\services) to map to the SSE Handler's port number. |
| TESTENVIRONMENT | None<br>**Explicitly configured in registry**<br>Live rigs: 0 or should not exist<br>Test rigs: 0 or +ve value | Value is always forced to 0 when the PCACryptoAPI DLL initialisation routine indicates a live environment.<br>On test rigs non-zero values can be used for special purposes. Currently:<br>2=Allows development to test against NPS with static partition. |
| THREADCONTROL | {GetR1={Count=0};<br>GetR1x={Count=1;IdleWait=100};<br>PreEE={Count=7;IdleWait=1000;C0Timeout=15000};<br>EE_IO={Count=4;SocketConcurrency=3;Target=SSE1,SSE2,SSE3,SSE4;LowThreshold=5;HighThreshold=10;MaxDiff=100;IdleWait=20 | **Count**: number of threads required of type. For GetR1X this **must** be 1, and for EE_IO this **must** be the number of target SSE Handlers supported, 4.<br>**IdleWait**: time to wait in ms when no work to do.<br>**C0Timeout**: timeout in milliseconds for [E1] when it is sent to SSE. |

| Value Name | Default Value[1] | Comment |
|---|---|---|
| | 0;QuickWait=20;MaxTimeout=30000;SSEPerfPeriod=120000;SSEPerfThreshold=8;SSEPerfTOpercent=40};<br>PostEE={Count=7;IdleWait=1000}}<br>{ReplyA3={Count=3;IdleWait=1000};<br>StatusMsg={Count=1;IdleWait=1000};<br>Commands={Count=1;IdleWait=30000;CMDTimeOut=60000};<br>All={InactiveWait=500;MinConnectDelay=400}<br>Heartbeat={NBEAdminPoll=15000}} | **SocketConcurrency**: maximum number of concurrent sockets per thread.<br>**Target**: comma-separated list of internal identifiers for the target system accessed from each thread. For EE_IO thread, this is a list of target SSE Handlers at the SSE, and must match the Count.<br>**LowThreshold**: bias towards sends when below this threshold.<br>**HighThreshold**: bias towards recvs when above this threshold.<br>**MaxDiff**: a socket is considered to be lagging too far behind once it is this far behind the other sockets. Not used in DCS<br>**QuickWait**: wait in ms– (for EE_IO) when we expect to receive an [A2] soon;<br>**MaxTimeout**: maximum FI_EE timeout in milliseconds.<br>**SSEPerfPeriod**: the period in milliseconds to continually check for unresponsive sockets.<br>**SSEPerfThreshold**: minimum number of messages within period on order for check takes place.<br>**SSEPerfTOpercent**: percentage of messages sent in period that must be timed out for socket connection to be deemed unresponsive.<br>**CMDTimeOut**: time during which an operator command should have completed (in ms).<br>**InactiveWait**: wait between state change checks when not active.<br>**MinConnectDelay**: delay in milliseconds between each socket connection attempt. Implemented to prevent overload of X.25.<br>**NBEAdminPoll**: time the NPS_SYSTEM_PARAMETERS table is polled by Exceptions thread (in ms) |
| TOTALCONNECTIONTIMEOUT | 300000 (5 minutes) | Time in milliseconds allowed for this agent to connect to one of its NPS Oracle instances on startup. |
| TRACECALLS | 0 i.e. no tracing | Performance/super trace |
| TRACELEVEL(†) | 0 i.e. no tracing | Controls which trace messages are to be output. Not used by agent because this would direct messages to the standard event log rather than to a private set of trace files (see (†) at bottom of table) |

| Value Name | Default Value[1] | Comment |
|---|---|---|
| TRACE_PARAMS | "maxsize=1024" | Each trace log has a size of 1024Kb. |
| TRACE_TYPE | "ManyTraceLog"[2] | Trace is output to a private set of trace logs, one per thread. |
| USERNAME(*) | None<br>**Explicitly configured in registry** (but see (*) at bottom of table) | User name for agent to connect to Oracle database |
| VA_COUNT | 1 for DCS | Count of Virtual IP Addresses per EE_IO thread. Must not exceed SocketConcurrency.[3] |
| WALLET_CREDENTIALS(*) | None<br>**Explicitly configured in registry** DCSNPS1,DCSNPS2 | Information to connect to Oracle database |

Table [ SEQ Table * ARABIC ]: Registry

(*) Not used (or configured in registry) when a value exists for WALLET_CREDENTIALS. Implementation allows WALLET_CREDENTIALS to be omitted whence DBLOCATION, PASSWORD and USERNAME must be present.

(†) TRACELEVEL allows control over the level of trace on a per thread type basis. There is a default TRACELEVEL specified via the TuneableTrace facilities[4] (rather than the agent's TRACELEVEL) with a level of 4 for each thread type (More information on this is provided in [ REF OPOVER \h ]).

For those attributes specified in a Structured attribute format (SAN) the registry value to be used follow the normal agent precedent rules e.g. those specified under NX_NQ_DCS/*service name* take precedence over those under NX_NQ_DCS. However overriding the default values is then carried out on an attribute by attribute basis so only those attributes with values different from the defaults need to be supplied. This data is case insensitive.

#### 2.1.4.1.1 Index for a Virtual IP Address

There is a %V substitution value provided for agents where there is a number of alternative Virtual Addresses to choose from. This is not used by DCS.

### 2.1.5 Configuration via NPS

The general principal is that the **NBX Configuration Parameters table** (the **TMS_TX_NBX_CONFIGURATION** table) in NPS is for configuration data that would have been reference data on traditional agents.

[2] Any other trace destination could have a disastrous affect on the performance of the agent.
[3] SocketConcurrency controls the number of socket connections made, whereas VA_COUNT controls the number of distinct Virtual IP addresses connected to. In SOCKET_HOST and SOCKET_SERVICE, %V (in the range 1..VA_COUNT) is used to select the different addresses. SocketConcurrency greater that VA_COUNT is used to provide alternative connections for resilience – for the DCS Agent VA_COUNT is 1.
[4] Allows dynamic control over the trace level.

One specific principle is that configuration via the NPS is at the level of a specific FI Type which for this agent is DCS. It does not provide for any lower level of granularity, for example by Windows service name.

Each row in the table is for a specific configuration item, identified by its Parameter_Name, and Applies_To a specific FI Type. Rows with any other Applies_To value act as ‘comments’ and provide a description of a specific configuration item or set of configuration items; by convention, an Applies_To value of “Description” is used for such rows. The entries that apply to DCS will have the APPLIES_TO to column set to ‘DCS’, and are defined in [ REF DCSHLD \h ]. For documentary purposes only, an Applies_To value of “Generic” is used for configuration items that apply equally to all FI Types. However, such items will have been expanded into one row for each FI Type in the **TMS_TX_NBX_CONFIGURATION** table itself.

The [ REF GENHLD \h ] defines those parameters that Fujitsu Services Ltd. can change without prior consultation with Post Office Ltd., subject to there being no change in the characteristics of the service contrary to those contractually agreed and accepted.

## 2.2 Implementation Overview

### 2.2.1 General

Resilience is achieved in various ways:

a) Reducing the likelihood that a single agent cannot complete the processing e.g. by having access to two NPS Oracle instances.

b) Providing two agents with one active and the other waiting as a standby. The decision on which agent is currently active is decided from the heartbeat records written by the two agents. The detailed design of this resilience is covered in section [ REF _Ref74103009 \r \h * MERGEFORMAT ].

c) The active agent detecting when it is not as healthy as it should be.

d) Providing an independent resilient network for each FI. The Agent will report inability to (re)establish connections, but will rely on the network to re-establish itself (with manual intervention as necessary) and will **not** attempt to second guess network availability.

The Heartbeat threads manage all aspects of the resilience.

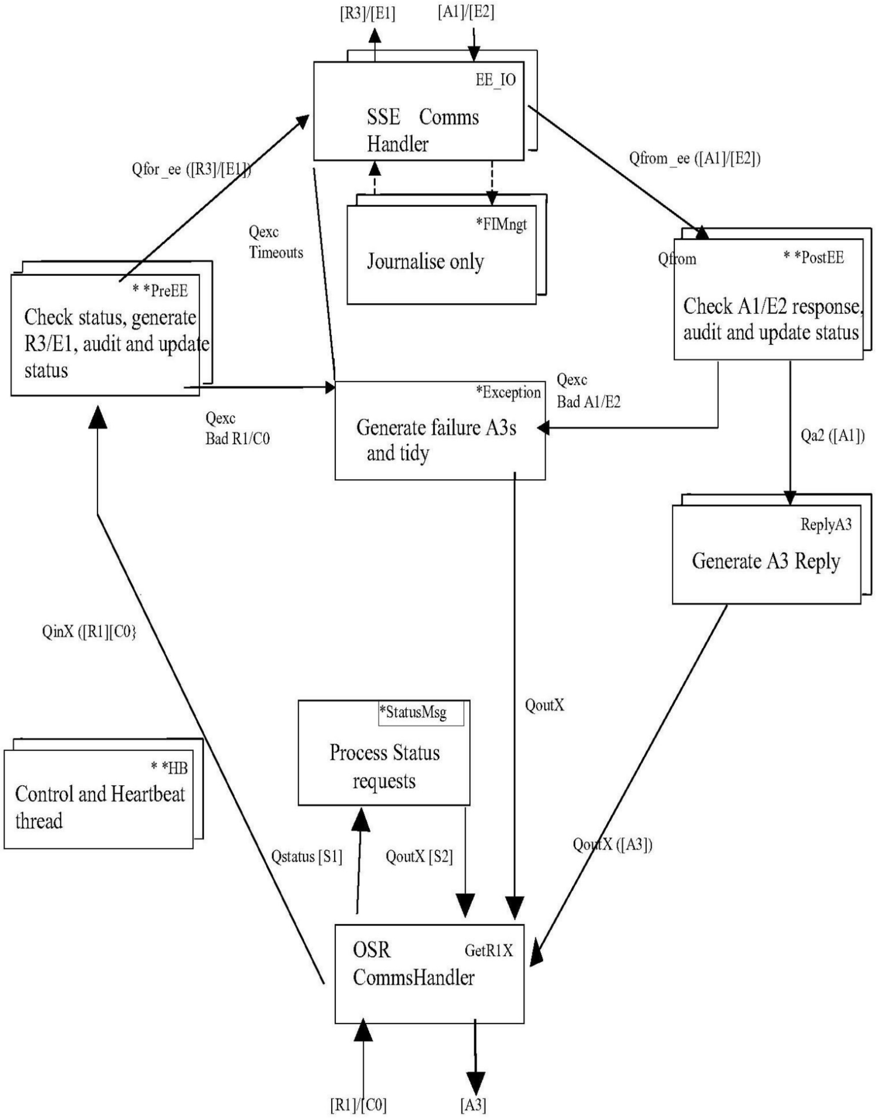

**Figure [ SEQ Figure * ARABIC ] - Overview of the Agent's structure and message flow**

> *Note: [ REF _Ref120678465 \h ] omits the Commands thread that supports processing external commands read from the Operator Commands table in the NPS.*

Under HNG-X messages are no longer signed. [R1] messages will be sent directly to the PreEE thread. The ReplyA3 thread will format outgoing messages but will not need to do any signing.

The **PreEE** and **PostEE** threads are low on mill usage but have long elapsed times as they have external interactions, particularly with NPS – indeed NPS has proved to be the bottleneck.

The threads that connect to the NPS do so during the initialisation phase of the agent. All failed connections are periodically retried. Where appropriate the system chooses a single pool of threads that have a successful connection to the same NPS host. The Oracle connections all use the agent's multithreaded Oracle library.

In situations where an agent cannot recover from a failure then it will close down and be restarted by Tivoli. Wherever possible, it will make the reason for this closedown available in a heartbeat record.

There is a **Heartbeat thread**, which acts as a control thread for the whole Agent, and a second Heartbeat thread that connects to a different NPS node; one of these will monitor the other. As both of these Heartbeat threads could hang when calling NPS, there is a third thread, the **Heartbeat Monitor thread**, whose sole task is to monitor the two Heartbeat threads and terminate the Agent if they both appear to have hung.

The agent connects to the FI only if the heartbeat thread sets this agent instance to be the Active agent; the Standby agent does not connect to the FI.

#### 2.2.1.1 NPS connection overview

The NPS is a dual system accessed via two distinct Oracle instances, the two providing a consistent view of the database. The Oracle configuration (in the TNSNAMES.ORA file) on the platform will offer two names (NPS1 and NPS2) to be used by the DCS Authorisation Agents, one for each of the two Oracle instances.

The NPS as a whole is duplicated for disaster recovery (DR) purposes. Fallback to the DR system is necessarily slow (hours rather than minutes) and it is understood that the Agent configuration need not take account of this, with everything being transparent behind the same names in TNSNAMES.ORA.

Each Heartbeat thread connects to a different NPS Oracle instance. The main working threads that access NPS, namely the PreEE and PostEE threads, are also duplicated in order to have separate connections; one of each pair is 'active', the reserve acting as a hot standby. These threads remain connected to their NPS Oracle instances whenever possible. The Heartbeat thread controls which sets of threads are running. A separate control is allocated to each thread so that processing can be switched over independently. *(These threads are indicated by ** in the above diagram.)*

The FIMngt, and Exceptions threads *(indicated by * in the above diagram)* do not follow this policy as their performance is less critical and in the case of the Exceptions thread it is desirable not to fail a message. These threads switch their single connection whenever it becomes necessary, trying the alternative connection first. This might sometimes result in an unnecessary switch after a transient failure, but offers the best chance of a speedier reconnection. It is also for this reason that these threads connect to specific Oracle instances rather than simply to the Oracle service.

Each Agent is configured with a preference as to which Oracle instance to use. This preference is not guaranteed, as it is achieved by imposing a delay before the second Heartbeat thread attempts to connect to the non-preferred instance.

Each Authorisation Agent instance, both active and standby, is required to home their active connection to the preferred instance every night. *(Note that this action is controlled by the Agent itself, not by Maestro as once proposed.)*

### 2.2.1.2 Beads, queues and the Hash Table

Each new [R1]/[C0] is allocated to an in-store data entry for managing that message as it passes through the agent. These in-store data entries are referred to here as **R1ToA3 entries** (or **beads**). An R1ToA3 entry not only contains the data required during processing but also includes state information and some fields required for queuing. These entries are re-usable and have a state of 'free' whilst unallocated. Once allocated, the state is updated by each thread as it passes through each stage of processing. In addition to the normal states there are some associated with timed out messages and discarded messages.

The management of queues within these agents is critical. In addition to the message flow queues shown in [ REF _Ref120678465 \h * MERGEFORMAT ] the agent has some additional queues and tables for:

a) the free chain of unallocated R1ToA3 entries (FreeQ)

b) a **Hash Table** keyed on TID / Indicator for fast access to R1ToA3 entries (TxnQ). The Indicator distinguishes between an [R1] entry and a Reversal entry so that both entries can co-exist. *(Note: The Reversal indicator is sometimes known as the [C0] indicator.)*

It is important that the management of the critical sections in the code (critical sections handle shared resources that must not be concurrently accessed by more than one thread) must not allow deadlock. The safest approach is to only ever hold one critical section at a time. If it is necessary to hold 2 concurrently there must be a defined order for acquiring them, but this is not expected to be necessary.

The Hash Table (TxnQ) should include at least 1000 hash values so that searching is very fast. It should have several Critical sections in order to break the hash values down into a number of small subsets. Note that the Hash Table is also used to find an entry after a reply from the SSE.

[T1] messages are allocated a bead, but are not included on the hash table.

### 2.2.1.3 Transaction States

The processing of a transaction is controlled by transition through a series of states. These are recorded in the transaction's CURRENT_STATUS in the Transaction Status table in NPS.

The following table summarises the various transaction states. For documentary purposes only, they are categorised into a small number of categories. The last column gives the numeric value of the state as held in the Transaction Status table.

| Category | Transaction State | Description | IsR1 | Set by Thread | Value |
|---|---|---|---|---|---|
| In progress | R3_Auth | Generated [R3] (but not necessarily sent to Streamline) | True | PreEE | 2 |
| In progress | Reversing | Generated [E1] (but not necessarily sent to Streamline) | False | PreEE | 2 |
| May be complete | Authorised | Received [A1] indicating Accepted | True | PostEE | 4 |
| May be complete | A1_Timed_Out | No [A1] was received before it timed out | True | PostEE | 5 |
| Completed | Declined | Received [A1] Decline | True | PostEE | 6 |
| Completed | R3_Not_Sent | Unable to send [R3] as no usable session | True | PreEE | 7 |
| Completed | Reversed_OK | Received [E2] indicating successfully reversed | False | PostEE | 8 |

| Category | Transaction State | Description | IsR1 | Set by Thread | Value |
|---|---|---|---|---|---|
| Completed | Reversal_Failed | Unable to reverse, e.g. unable to send [E1] as no usable session, reversal declined | False | PreEE | 9 |
| Completed | Orphan_C0 | Received [C0] when there is no previous Status entry for this transaction | False | PreEE | 10 |
| Completed | Discrepancy | Bad [A1] or [E2] received. There may be a discrepancy between the FI_EE's view and the Counter's view as to the success of the transaction.<br>(For an Orphan_A1_E2 there will not be enough information to locate or create a Status entry.) | False | PostEE | 11 |
| Completed | Orphan_A1_E2 | Unmatched A1 or E2 received, can be after a timeout | False | PostEE | N/A |
| | | *The following Status values are never used in the Transaction Status table, but are used internally by the Agent* | | | |
| Not started | Unknown | Status entry has not yet been fetched | | | 0 |
| Not started | No_Status_Entry | There is no Status entry | | | 1 |

**Table [ SEQ Table * ARABIC ]: Transaction States (Status values)**

#### 2.2.1.4 Timeouts

The only timing out of messages is on those that are late from the SSE (or can't be sent to the SSE for a while).

#### 2.2.1.5 Cryptography overview

PCI requires the DCS agent to use new Cryptographic API [ REF PCICRY \h * MERGEFORMAT ] to encrypt the PAN block to create the (PAN)PK so it can be stored on the NPS. The Cryptographic API functions used are:

PcaInitialiseCrypto
PcaTerminateCrypto
PcaEncryptPAN

The contents of the PAN block, as defined in [ REF DCSHLD \h * MERGEFORMAT ], are a clear PAN, an Issue Number, a StartDate and an Expiry Date. A clear PAN is always present: the others are zero filled if not present, and there is a flag to indicated whether the Issue Number is present as all zeroes is a legal value. The PAN block is supplied to the `PcaEncryptPAN` function and the (PAN)PK returned.

The agent calls `PcaInitialiseCrypto` during initialisation. The agent continually repeats the call if it was not successful, up to the registry configurable time of `TotalConnectionTimeout`. There are two successful return code values from the function, one indicating that test keys are in use, the other indicating live keys. An informational event log message is written when the test key value is returned. The agent uses the test key return code to set a flag that is used to produce more detailed trace information in a test environment. Also the sensitive card data is not sanitised when traced in a test environment. The registry value `TESTENVIRONMENT` is forced to zero when the live key value is returned.

The agent calls the function `PcaTerminateCrypto` when closing down.

### 2.2.1.6 PCI CheckList

Section [ REF _Ref162443911 \r \h * MERGEFORMAT ] outlines the Cryptographic aspects of PCI. This section outlines the other associated changes for DCS.

[ REF GENHLD \h ] contains a table identifying the sensitive data fields involved and showing which messages contain them. The two new values are the Hashed PAN (H(PAN)), which is in fact not sensitive, and the Encrypted PAN Block ((PAN)PK). The implementation of PCI does not change the messages to/from the SSE at all. It does affect messages to and from the counter as follows.

1. [R1] and [C0] messages contain the new Hashed PAN value in the field that previously held the real PAN value (called PAN), and the real PAN value is held in a new Clear PAN field (called CPAN).

2. [A3] messages contain the Encrypted PAN Block in a field called EPAN.

3. [T2] messages contain both the Hashed PAN and the Encrypted PAN block in fields called PAN and EPAN respectively.

Hashed PAN and the Encrypted PAN Block are written to the NPS in the R3_Info field in the Status table, and to the PAN and ENCRYPTED_PAN columns in the Journal table. The Clear PAN is never written to the NPS, nor are any of the other sensitive values, except as part of the Encrypted PAN block.

When journalising a whole message in the MESSAGE column in the Journal table they are sanitised as follows:

1. The Card Details field in [R3] and [E1] messages are overwritten with '*'s.

2. A Clear PAN has all but its first 6 and last 4 characters overwritten with '*'s. This applies to [R1] and [C0] messages

3. Any other of the sensitive fields is also overwritten.

The sanitising is also done when writing any message or part message to a diagnostic trace log.

## 2.3 The Threads and Associated Queues

### 2.3.1 GetR1 (Listen) threads

The DCS Authorisation agent does not have any GetR1 threads.

### 2.3.2 GetR1X (Listen) threads

In the DCS Authorisation agent the GetR1X thread has all the functionality of the GetR1X thread in the NBS Authorisation agents (see [ REF NBSLLD \h ]) apart from the section headed PCI which does not apply. There is only one GetR1X and it handles all messages received from and sent to the Online Service Router (OSR) and returns replies to the same socket that a message was received on.

In addition it performs the following tasks:

1. Extract the MID and TID for the counter
2. Generate the SSE Message Number
3. Hold messages on a queue (Qhold) until any previous message for the counter has been processed.

#### 2.3.2.1 Extract the MID and TID

The agent reads the MID/TID file (location provided by registry value of INFILE) during early initialisation and generates an internal table of the MID and TID values for each counter in each post office. The MID/TID table has an entry for each counter in each post office; each entry has two sets of MID and TID values. One set is used for the counters Bureau transactions and the other for its General Sales transactions.

The HLD describes the checks the Agent performs on the file (over and above basic format checks). If the file is invalid, the Agent will terminate. After successfully processing the MID/TID file the agent creates a '**flag**' file with an '**.ack**' extension.

The MID and TID for the current counter are extracted for [R1] and [C0], but not [T1], messages from the internal table by LookupMIDTID() and LookupBureauMIDTID() functions. As stated in the HLD Bureau transactions are recognised by the XML element BdC being set to 1, rather than 0.

The TID is used to generate the hash key, with 'R1' or 'C0' added according to the message type. The TID is also used as the terminal identity included in all messages sent to Streamline (see [ REF DCSHLD \h ]).

**[DN:** None of this handling is changed from Horizon, and has been done by bringing the basic code across from the Horizon agent. It has also been extended as described in [ REF _Ref161989025 \r \h ]. **]**

#### 2.3.2.2 Generate the SSE Message Number

The internal MID/TID table is extended to hold a Message Number for each MID/TID entry and this represents the SSE Message Number of the last message sent to Streamline. These numbers are also held in the NPS in table TMS_RX_TID_MESSAGE_NUM.

The Message Numbers in the internal table are initialised from the TMS_RX_TID_MESSAGE_NUM table. This is done by the heartbeat thread (see section [ REF _Ref74103009 \r \h * MERGEFORMAT ]) just before the agent becomes active, so that it can pick up the values used by the agent that was previously active. Entries for which there is no value will be left set to a recognisable value (-1).

Streamline requires that each message from a TID is assigned a different Message Number and that it is one greater than the previous number used (in practise, consectutive numbers are not enforced by Streamline). In Horizon SSE Message Numbers were set on the counter and contained in an [R1]/[C0] in the TransNum attribute; in HNG-X this attribute will not be present in [R1]/[C0] messages. To ensure an increasing sequence during migration the agent looks for the existence of the TransNum attribute and will use that number in preference to a value from the internal MID/TID table. The [R1]/[C0] messages will not contains the TransNum attribute once a counter has been migrated.

If the [R1]/[C0] message does not contain the TransNum attribute, the GetR1X thread looks for a Message Number in the internal table and, if found, increments it (modulo 10000) and so generates a new Message Number for the [R1]/[C0]. As already stated, the internal table will return -1 if no previous value exists, and the Message Number generated will be 0 (i.e. 0000).

The Message Number is retained in the R1ToA3 entry and is also written as R3_Info to the transaction's Status entry (see [ REF _Ref73851177 \r \h * MERGEFORMAT ]). It is written to the TMS_RX_TID_MESSAGE_NUM table in the NPS (by the PreEE thread) as part of the same interaction that records the [R3] message to the NPS.

**[DN:** Although it should be possible to predict whether we need an UPDATE or an INSERT to record the Message Number in the NPS it is more robust to try an UPDATE, the normal case, and if this fails switch to do an INSERT. This can all be done within the same SQL block. **]**

The agent must only process one message from a counter at a time and uses the Qhold queue to do this (see section [ REF _Ref209432087 \w \h ]). The SSE Message Number is not generated until the message has negotiated the Qhold queue.

#### 2.3.2.3 The Qhold Queue

This queue is internal to the GetR1X thread and is used to satisfy the requirement that the agent must not send more than one message to Streamline at a time for the same TID. Having sent a message for a TID it must wait for the reply from Streamline, or for the message to be timed out, before sending another.

The internal MID/TID table is extended to have a flag (MsgInSystem) for each TID entry and this is set before a message is passed to the QinX queue. An index to the TID entry is stored in the bead to allow the flag to be cleared. The flag is cleared when the bead is freed (for an [R1] this will be after the [A3] has been sent back to the OSR). This flag will apply to the counter as a whole, rather than having a separate flag for each of its two TIDs (one Bureau and one General Sales).

**[DN:** The HLD suggests that MsgInSystem could be cleared when a reply is received from Streamline or when a message is timed out. This is true - however it seems sensible to wait until after the reply or timeout is recorded in the status table as this cuts down on the number of transaction states that need to be handled,**]**

Before setting the flag, and before generating the SSE Message Number, the agent must test whether the flag is already set, if it is the message is added to the end of Qhold.

The thread periodically (`ThreadControl.GetR1X.IdleWait`) scans Qhold to see if any entries have had their MsgInSystem cleared and if so remove the entry from the queue and complete the GetR1X processing of the message. Qhold must be strictly first in/first out for messages for the same TID so that their order is maintained.

Since there is only one GetR1X thread, and since this is the only thread that tests MsgInSystem, there is no need to use a Critical Section when manipulating the flag, though it should be declared as **volatile**.

**[DN:** There is scope for optimisation in this area, and it is proposed to keep a count of the number of held messages. This could prove to be an indication as to whether any optimisation would be worthwhile.**]**

### 2.3.3 Qin queue

There are no Qin queues in the DCS Authorisation agent.

### 2.3.4 Verify threads

There are no Verify threads in the DCS Authorisation agent.

### 2.3.5 QinX queue

The QinX was previously known as the Qverified queue.

The QinX queue is the input queue to the PreEE threads. It is in the same style as the Qin queue in that it is not a single queue but a number of queues, each with its own critical section. There is one queue per PreEE thread so that, in principal, each PreEE thread has its own queue. However, in practice, any PreEE thread can access any of these queues and will do so whenever its own queue is empty or it cannot get immediate access to its own queue (i.e. the critical section is currently in use elsewhere).

The Message Sequence Number associated with the R1ToA3 entry (or bead) by the GetR1X thread is used to balance the load when appending entries to these queues. If the intended queue cannot be accessed (critical section in use elsewhere) the entry can be queued on any available queue.

Each R1ToA3 entry represents one of the following. The entry contains the Trigger and a value for Current_Status (which may be Unknown).

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| R1 | GetR1X | [R1] | R1 | The Status entry has not yet been read, so the Current_Status is Unknown |
| C0 | GetR1X | [C0] | Reversal | The Status entry has not yet been read, so the Current_Status is Unknown |
| *Marked for 'reprocessing'* | Exceptions | Any of the above | | See [ REF _Ref73333853 \w \h * MERGEFORMAT ] [ REF _Ref73333853 \h * MERGEFORMAT ] |

**Table [ SEQ Table * ARABIC ]: R1ToA3 entries on the QinX queues**

### 2.3.6 PreEE threads

The PreEE threads are responsible for processing R1ToA3 entries on the QinX queue, generating [R3] or [E1] entries as appropriate, and passing them to the Qfor_EE queue for a particular SSE Handler.

If one of these threads runs out of work it has a short wait (200ms configurable via PreEE.IdleWait in the THREADCONTROL registry entry) on the agent's stop event before trying again.

> As PreEE threads access the NPS database, for each thread there is a reserve thread for accessing the database via the alternative Oracle instance.

#### 2.3.6.1 Reprocessing an R1ToA3 entry

An R1ToA3 entry may need reprocessing following a failure in the original processing. This can occur both internally within the thread and following requeuing by the Exceptions thread. A full discussion is given in [ REF _Ref73333853 \r \h * MERGEFORMAT ].

The 'for reprocessing' flag can indicate one of the following:

- unexpected state
- NPS failure, not committed
- NPS failure, commit status unknown

If the R1ToA3 entry is marked 'for reprocessing' on entry to the thread, the original entry conditions have to be reinstated. This is confined to the original trigger value, which has to be retained in the R1ToA3 entry for this purpose.

To establish the actual state of the transaction, its Status entry has to be read from the Status table. If there is none, then the Current Status becomes No_Status_Entry.

If the 'for reprocessing' flag is other than 'NPS failure, commit status unknown', the entry can simply be reprocessed but this time using the freshly established Current Status.

##### 2.3.6.1.1 Reprocessing when commit status is unknown

For 'NPS failure, commit status unknown' for an R1 (ReqType=IsR1Msg):

a) If there is no Status entry or there is a Status entry in state Orphan_C0, the commit must have failed. The [R1] can simply be reprocessed.

b) If the Status entry is in any other state, the [R3] will have been generated but not sent. It is now audited as {R1|No_R3} with an Agent_Diagnostic describing the NPS failure. *(It is unnecessarily complex for the PreEE thread to reprocess the [R1], so not processing just the one [R1] is considered adequate.)*

For 'NPS failure, commit status unknown' for a Reversal (ReqType=IsC0Msg):

c) The [C0] can simply be reprocessed.

#### 2.3.6.2 R1: Checking the transaction type

There is no NPS configuration data relating to transaction types in DCS.

The handling of the different transaction types (attribute TranType) is just a part of the Message Mappings described below.

#### 2.3.6.3 Reversal: Fetching the Status entry

If the Current_Status for a reversal is Unknown or any other value than No_Status_Entry, the Status entry is now fetched from the Transaction Status table. If it is found, the Current_Status in the R1ToA3 entry is set accordingly; otherwise it is set to No_Status_Entry.

This check mops up the case where the transaction to be reversed is so old that there is no longer a Status entry for it in the Transaction Status table, as well as the (more normal) case where the [R1] was not processed by a DCS Authorisation Agent.

#### 2.3.6.4 Checking for an 'Old Reversal'

Every reversal is first checked for being 'old', i.e. the transaction to be reversed is sufficiently old that the reversal should not be forwarded on to the FI_EE. The check is that the elapsed time since Receipt_Date and Receipt_Time is greater than the `Reversal.Old.Threshold` configuration parameter.

If the check shows it to be an 'Old Reversal', the message is processed for the trigger Old_Reversal according to the state tables see [ REF _Ref73353387 \r \p \h * MERGEFORMAT ] and [ REF DCSHLD \h ].

If the message is so ancient that any Status entry for the transaction would have been removed from the Transaction Status table by routine housekeeping, the insertion of a new Status entry for it must be bypassed. This check is that the elapsed time since Receipt_Date (only the date and not the time) is greater than the `Reversal.Old.MaxAge` configuration parameter.

#### 2.3.6.5 Selecting a SSE Handler

The PreEE thread is responsible for selecting the SSE Handler to which to route an [R3].

The thread is also responsible for load balancing the [R3]s across the SSE Handlers and the SSE socket connections available within them.

For DCS reversals there are no restrictions as to which SSE Handler or VA the [E1] may be sent and the socket to be used is selected in the same way as for the [R3]. This makes it unnecessary to retain memory of the socket to which the [R3] was sent.

The DCS Agent follows the NBS approach to load balancing which is to load balance per socket.

To enable the PreEE thread to do the above, the EE_IO threads advertise (via global variables) the state of each SSE Handler and of each of the SSE Handler's sockets. The following information is required:

a) Count of SSE Handlers (from EE_IO.Count in the THREADCONTROL registry entry)

b) Count of sockets per SSE Handler (from EE_IO.SocketConcurrency in the THREADCONTROL registry entry)

c) For each SSE Handler:
    SSE Handler Index (the EE_IO thread number: p = 0, 1, …)
    SSE Handler Identifier (a name for reporting purposes)
    Flag as to whether there is a fully operational logged-on session ('SSE Handler available')

d) For each connected and fully operational socket:
    Socket Index (q = 0, 1, …)
    Flag as to whether the socket is connected and fully operational ('socket available' and 'socket not Unresponsive')

Messages are not sent to a socket marked as being Unresponsive. The EE_IO thread will mark a socket as Unresponsive when it detects it is performing badly (see [ REF _Ref121128343 \r \h * MERGEFORMAT ]).

##### 2.3.6.5.1 Load balancing [R3]s across the SSE Handlers and VAs

As just stated the DCS implements load balancing across the available sockets.

Each [R1] has been assigned a Message Sequence Number by a GetR1X thread for the purpose of load balancing. (Those for [R1]s have been allocated independently of any allocated to Reversals.) For this purpose the sockets are considered to be ordered as per the following example.

The SSE Handlers could be {$SH_p$, p = 0..3} with each $SH_p$ having three sockets {$Q_{pq}$, q = 0..2}. Then the order is {$Q_{00}$, $Q_{10}$, $Q_{20}$, $Q_{30}$, $Q_{01}$, $Q_{11}$, $Q_{21}$, $Q_{31}$, $Q_{02}$, $Q_{12}$, $Q_{22}$, $Q_{32}$}. Note that consecutive entries in this ordering are for different SSE Handlers. This ordering is achieved simply as now described.

Let $T_Q$ be the total count of possible sockets across all SSE Handlers. By first choice the message with Message Sequence Number $S_A$ is allocated to SSE Handler$_X$ given by:
    X = $S_A$ modulo (count of SSE Handlers).
Within that SSE Handler it is allocated to the socket $Q_{XY}$ whose index Y is given by:
    Y = ($S_A$ modulo $T_Q$) / (count of SSE Handlers)

If this preferred socket is not operational, and if overall there are sufficient sockets that are operational (see below), then by second choice the message is allocated to the M$^{th}$ operational socket (M=0..$C_Q$ using the above ordering of sockets $Q_{m=pq}$ across all SSE Handlers) where
    M = $S_B$ modulo $C_Q$

where $S_B$ is a locally maintained sequence number of messages that have been subjected to the second choice (SecondChoiceCount), and $C_Q$ is the total count of connected and fully operational sockets across all SSE Handlers.

It may be that the EE_IO threads have only just started establishing connections and need to be given time to get going. To prevent overloading the few connections that have been established, no second-choice allocations are made unless the number of connections has at least exceeded a threshold. This configurable threshold, UseAltSocket.PercentMustBeOper, is expressed as a percentage. Currently it is set to 33, meaning that 4 sockets out of the 12 must be operational for second-choice allocations to be used.

If the threshold is not reached, then one of two behaviours ensues. If the FI_EE was previously not known to be Down, this thread will enter a loop of 1-second sleeps to see if any sufficient sockets becomes operational, subject to a configurable overall time limit (Timeout.WaitToSend, currently 18 seconds). If after this period nothing has changed, this thread deems the FI_EE to be **Down**, and the R1ToA3 entry is placed on the Qexc queue with a trigger of R3_No_PI.[5] If the FI_EE was already known to be Down, the R1ToA3 entry is similarly passed to Qexc, but this occurs only after a short sleep (100 ms) to prevent flooding the Exceptions thread.

The change from not Down to Down is accompanied by a warning in the Event log:
    "FI deemed to be down (fewer than %d operational sockets)."
The converse change is accompanied by an information message to the Event log:
    "Cleared 'FI down' flag (there are at least %d operational sockets)."

This Down flag is local to a PreEE thread, so the above messages will be output once per thread. *(Note that the Heartbeat thread is maintaining a similar but distinct flag for the purposes of reporting the FI_EE's operational status to OMDB via the Heartbeat table.)*

The allocated SSE Handler and socketVA are recorded in PI_ROUTING in the Status record though this is not used in the routing of any Reversal (see below).

[5] The trigger R3_EE_Down had been intended for a similar circumstance but it is no longer set. However, some relics of it exist within the code.

##### 2.3.6.5.2 Routing Reversals

For DCS there are no restrictions as to which SSE Handler or socket an [E1] may be sent, and the decision is made independently of and in the same way as for an [R3].

#### 2.3.6.6 Checking for a Stale [R1] / [C0]

With the delays that may arise because of bottlenecks in the NPS's processing of previous messages, or because of detecting when the EE has just gone Down, it is necessary to test whether the [R1] is **Stale**, i.e. whether it has been in the Agent layer so long that it is pointless forwarding it to the EE for authorisation. The check is controlled by the configuration parameter StaleR1.Threshold. If more than this interval has elapsed since the Reversal was first initiated (using the OSR timestamp control field ), the message is processed for the trigger Stale_R1 according to the state tables described in [ REF _Ref73353387 \r \p \h * MERGEFORMAT ] and given in [ REF DCSHLD \h * MERGEFORMAT ].

A [C0] is also checked for being Stale, using the configuration parameter StaleC0.Threshold, and if so it is simply discarded. There is no 'guaranteed' [C0] to processed in due course, but it is necessary to discard [C0]s to help relieve any build-up of queues particularly after an agent failure.

The Stale Thresholds are now set at 15 seconds. Originally they were higher, closer to the Counter timeout value, but this could result in an [A1]-Accept being received from the FI by a stressed Agent too late to return to the Counter in time.

#### 2.3.6.7 Generating the [R3] / [E1]

If an [R3] or [E1] cannot be constructed because there is a problem with the input data (e.g. a mandatory field cannot be generated because of a fault in the input data), the message is processed for the trigger Bad_R1 or Bad_C0 according to the state tables (see [ REF _Ref73353387 \r \p \h * MERGEFORMAT ] and [ REF DCSHLD \h ]).

#### 2.3.6.8 Late Reversals

A Reversal is deemed to be 'late' if the attempt to send the [E1] occurred longer than a threshold after the [R1] was sent.

The times compared are the Transmission Times of the [R1] and the [C0]. The configuration parameter is `Reversal.Late.Threshold`. The [R1]'s Transmission Time has been remembered in the R3_TSMP field in the Status entry. This time difference (in seconds) is to be recorded in the Journal record for the [E1], in REVERSAL_DELAY.

#### 2.3.6.9 Message Mappings

The message mappings are defined in [ REF DCSHLD \h ]. Apart from the SSE Message Number (described above) they are the same as for the Horizon DCS Authorisation agent.

The mappings are not driven by configuration parameters as they are in the NBS Authorisation agent.

Many of the ICC-specific fields are in binary or binary-coded decimal on the interface between the Counter and the PIN Pad and chip. The Counter will already have converted them into hexadecimal or decimal characters in the [R1] messages, the format required in the messages to the SSE.

It is not necessary to retain data from an [R1]/[R3] or an [A1] in order to process a [C0]. However the R3_INFO, A1_INFO and REVERSAL_INFO columns in the transaction's Status Entry (see [ REF _Ref73851177 \w \h ]) are used to hold data required for a [T2] status message.

#### 2.3.6.10 The Status entry and state changes

*(The terms 'status' and 'state' have become a bit confused here and are essentially synonymous. Note, for example, that a transaction's current state is held in a field called Current_Status.)*

In the case of an [R1] the PreEE thread assumes that it will be processed normally and generates an [R3]. The status entry is created with an insert that is expected to succeed. If it doesn't the Oracle transaction is aborted, the Status entry must be read (without locking) to find its current state and the message re-processed accordingly. On a successful insert the [R3] is audited and the SSE Message Number saved to the TMS_RX_TID_MESSAGE_NUM table (see [ REF _Ref161989025 \w \h ]) in the same commit as the insert and then the R1ToA3 entry is queued on the appropriate Qfor_EE queue for its now allocated SSE Handler.

Access to the Transaction Status table uses YDDD, Terminal_Id and Trans_Num as the keys to select the required Status entry – this information will already be in the R1ToA3 entry. The processing is controlled according to the state transition tables.

In the case of a [C0] (or retry) the normal process is to generate an [E1], audit it and save the (SSE) Message Number in the same commit as the status entry is updated, and then queue the entry on the Qfor_EE queue for its SSE Handler. Qfor_EE is a queue in the same style as Qin except that messages must be queued on the queue for the SSE Handler.

Any update of the Status entry must only succeed if it is the expected change of state (i.e. no-one has updated the entry since the fetch). This is achieved by using a WHERE clause on the update checking that CURRENT_STATUS is as expected. If it is not in the expected state, the Status entry must be read (without locking) to find its current state and the message re-processed accordingly.

The behaviour of the PreEE thread is described at a set of state changes in the State Transition tables given in the [ REF DCSHLD \h ]. They include an indication as to which rules need to be performed in this thread and which in the exceptions thread. A documentary overview of how these tables are used in the PreEE, EE_IO and PostEE threads is given in section [ REF _Ref159737666 \r \h ] (Appendix V1).

#### 2.3.6.11 Auditing / Journalising

Every entry in the Transaction Journal table contains a Message Part Sequence Number, which is 1 for the first entry for a transaction, incremented by one for each subsequent entry for that transaction. The sequence number used must be recorded in the Status entry. Therefore, every insert into the Journal must be preceded by an insert / update of the Status entry in the same commit unit. For a Status insert the sequence number is 1. Each update must contain a WHERE clause checking that the previous value of the sequence number is as expected.

Sensitive data must not be written in clear to the transaction journal (the data that is considered sensitive is identified in [ REF GENHLD \h ]). This means that the journal record for an [R3] or an [E1] must not contain any sensitive data item in clear in a freestanding field, and sensitive data in the journal copy of the message must be overwritten with asterisks. (Neither must this sensitive data be visible in any diagnostic tracing.)

The following table gives information on the Journal records. This should be read in conjunction with the spreadsheet in the [ REF NBXJNL \h ].

| Journal Type & Subtype | Audited here or in Exceptions thread[6] | Condition | Data to be audited |
|---|---|---|---|
| | | | |
| {R1\|Dup} | Exceptions | Duplicate [R1], should never occur<br>No Failure-[A3] is generated | [R1] fields |
| {R1\|After_C0} | Exceptions | [R1] not forwarded after the [C0] has already reversed it<br>No Failure-[A3] is generated | ditto |
| {R3\|DCS} | *PreEE | DCS Authorisation Request | [R3] & [R1] fields |
| {A3\|No_R3} | *Exceptions | Failure-[A3] required following inability to send an [R3]. One of:<br>▪ R3_No_PI: Response_Code 30<br>▪ R3_EE_Closed: Response_Code 31<br>▪ Bad_R1: Response_Code 37<br>▪ | [R1] & [A3] fields |
| {C0\|Orphan} | Exceptions | [C0] received without a prior [R1]. Normally because the [R1] was never received, but could be because the [C0] has somehow overtaken the [R1] | [C0] fields |
| {C0\|Old} | Exceptions | Old Reversal: see [ REF _Ref72479892 \r \h ] | ditto |
| {C0\|No_E1} | Exceptions | Format error in [C0][7] | ditto |
| {E1\|E1_C0} | *PreEE | DCS Reversal Request (only 1 per transaction) | [E1] & [C0] fields. |

**Table [ SEQ Table * ARABIC ]: Journal records for PreEE**

#### 2.3.6.12 NPS exception handling

When writing to the Status and Journal entries etc in a single commit unit, the following exceptions may occur:

[6] Ones marked with an asterisk must be audited in the stated thread: those in the PreEE thread for performance reasons, in the Exceptions thread when a Failure-[A3] is to be generated. The choice between the two threads in the other cases is advisory but not necessary for reasons of logic.

[7] If the format is so bad that the keys necessary to construct the bead are missing, then it cannot be audited. However, an NT event will have been raised{*}.

a) Unexpected Status value or Message Part Sequence Number

b) Oracle error

Please see [ REF _Ref73333853 \r \h * MERGEFORMAT ] for a full discussion. As part of the recovery processing, the R1ToA3 entry may need to be reprocessed by this thread – see [ REF _Ref73355194 \r \h * MERGEFORMAT ].

### 2.3.6.13 Crypto

No Digital Signature checking for DCS.

#### 2.3.6.13.1 Decrypting the sensitive Encrypted Data

No decryption done for DCS.

#### 2.3.6.13.2 PIN Block translation

No PIN Block translation for DCS.

#### 2.3.6.13.3 Encrypting the Sensitive Data

The PreEE thread encrypts the sensitive data that needs to be written to the Status and Journal tables in the NPS. Details of the encryption are given in section [ REF _Ref209932849 \w \h ].

The encryption is done for [R1]s and Orphan [C0]s only. For normal [C0]s the Encrypted PAN Block will be the same as for the [R1] it reverses and can be extracted from the R3_Info field which will have been read from the Status table as part of the normal processing of such a [C0].

When the NPS is updated the Encrypted PAN Block and the Hashed PAN will be written to the Status table, in the R3_Info, and to the Journal table, in columns ENCRYPTED_PAN and PAN respectively. The clear PAN is no longer journalised. This will apply on most if not all NPS Status/Journal updates not just the ones in the PreEE thread though it is only this thread that will actually change the values.

(Note that the Hashed PAN for a non-Orphan [C0] is the same as that for the [R1] it reverses. We could check this but it should be unnecessary.)

### 2.3.6.14 Statistics

The statistics in the table below are collected for the SSE; they are not specific to any SSE Handler. They are collected per thread and accumulated across the thread class by the Heartbeat thread.

| Statistic | External name | Internal name | Description |
|---|---|---|---|
| Count of Late Reversals | LTR | LateRevCount | See [ REF _Ref72660003 \r \h ] |
| Count of Stale [R1]s and [C0]s | STAL | R1C0StaleCount | Counted in this thread for both [R1]s and [C0]s. Also counted by other threads |

**Table [ SEQ Table * ARABIC ]: Statistics maintained by the PreEE threads**

## 2.3.7 Qfor_EE queue

The Qfor_EE queue is the input queue to the EE_IO threads. It is in the same style as the QinX queue in that it is not a single queue but a number of queues, each with its own critical section. However in this case each EE_IO processing thread only has access to its own queue. The PreEE thread chooses the queue to use according to the R1ToA3's SSE Handler value (see [ REF _Ref210812244 \r \h ]). Should a critical section already be in use elsewhere there is a null sleep (gives other threads a go) before obtaining the critical section (which may then involve a wait).

Each R1ToA3 entry represents one of the following. The entry contains the Trigger and a value for Current_Status, though these are not of relevance to the EE_IO threads.

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| R1 | PreEE | [R1] | R1 | ReqType=IsR1Msg |
| C0 | PreEE | [E1] | C0 | ReqType=IsC0Msg |

Table [ SEQ Table * ARABIC ]: R1ToA3 entries on the Qfor_EE queues

## 2.3.8 Qt queue

The Qt queue is mainly included to assist with understanding the model. The EE_IO thread does not necessarily transfer entries directly from Qfor_EE to Qt. They are only moved onto Qt when they are sent to the FI_EE. This queue is private to the thread and does not need to worry about critical sections (if used there will never be a clash). In practice, there is one queue for each supported timeout value. The trigger is changed to Trigger_At_FI after the entry has been put on a Qt queue.

## 2.3.9 EE_IO and FIMngt threads

### 2.3.9.1 Overview

The DCS agent treats the SSE as if it had a number of SSE Handlers, although this is not a real division. Each EE_IO thread represents a SSE Handler. There may be several socket connections to the SSE from each SSE Handler. Each EE_IO thread has a logical identifier for identifying the SSE Handler that it supports, configured in ThreadControl as a list of **Targets** with one entry per thread. Each such identifier is regarded as the **SS Handler Name** when recording events in, for example, the Transaction and Management Journals.

The DCS agent always initiates the connections to the SSE. In general a SS Handler may present itself as one or more TCP/IP Virtual Addresses (VA = an IP address and port combination); though there is only one such combination for DCS.

More information on configuration is given in [ REF _Ref74125544 \h * MERGEFORMAT ].

Each EE_IO thread has data that is globally accessible within the agent and this data has a hierarchy that is:

a) SSE Handler-based (e.g. SSE Handler name),

b) target Virtual Address-based, and

c) connection-based (availability and threshold details).

Access to and from the FI is via sockets. On activation a separate thread is spawned for establishing the socket connections and disconnections (**AsyncIO thread**). This thread continually repairs any lost connections all the time that the EE_IO thread is active (unless it is administratively closed). It closes the connections when the thread is no longer required to be active. These tasks are carried out in a separate thread so they do not delay the normal EE_IO thread's processing. A configurable delay (`MinConnectDelay`) is imposed between successive connection attempts, to the same or different SSE Handlers, to avoid overloading X.25. The delay applies to the first connection for an SSE Handler; it is doubled for the second and subsequent connections.

Apart from establishing the socket connections, the EE_IO thread is used for all socket communication with its SSE Handler. The one-to-one relationship between an EE_IO thread and a SSE Handler makes it easier for handling timeouts and for managing the calls to the several sockets (without the need for critical sections). Each transaction message is targeted at a SSE Handler and a socket within that SSE Handler by the PreEE thread (see [ REF _Ref210812244 \r \h ]).

There are no management messages for the SSE for the DCS agent to support, and so no FIMngt threads are required for this purpose. However FIMngt threads are required for the journalising of the state of the connections to the SSE, for the SSE Handlers and the individual sockets. This is necessary since only the FIMngt threads and not the EE_IO threads have a connection to the NPS. There is one FIMngt thread for each EE_IO thread.

The establishment of a SSE Handler session is covered in section [ REF _Ref72583576 \w \h ].

### 2.3.9.2 Message Processing Control

Whilst active the EE_IO thread has a loop that performs the following:

a) Send an [R3]/[E1] and include its R1ToA3 on Qt (up to one send per socket).

b) Receive replies from the SSE and match them against the messages that have been sent (up to one recv per socket).

c) Checks Qt for timed-out [R3]/[E1]s.

d) Checks the agent's stop event or for a change from active.

If there is no work to do, it does a short wait (20ms, configurable via `EE_IO.IdleWait`).

### 2.3.9.3 Receiving Messages

The receiving section of code always attempts to receive at least one [A1]/[E2]. After completing a scan of the sockets the EE_IO thread checks its Qt queue for any timed out [A1]/[E2]s.

The EE_IO thread needs to validate the input message at this point to match an [A1]/[E2] message to its request message.

The thread matches an [A1]/[E2] against the messages sent by searching for an entry in the hash table having the same TID as the [A1]/[E2]. It is not possible to distinguish an [A2] from a [E2] by their content, but only by using the R1ToA3 entry on the hash table. If an entry is found checks are made that its (SSE) Message Number matches that in the incoming message, and that its state is Trigger_At_FI, and only if these checks are successful is the message processed further.

SSE timeouts are controlled using clock ticks, which potentially cycle round. Periodically the thread checks the heads of the Qt and internal waiting queues for [R3]/[E1]s exceeding their FI_EE time out. The timeout for an [A1] is controlled by the value of the mandatory <AgtTmOut> attribute in the [R1]. The timeout of an [E2] is controlled by `Reversal.Retry.Wait`. Any that have timed out are moved to Qfrom_EE with a reason of FI_EE timeout.

### 2.3.9.4 Sending Messages

Sending of [R1]/[E1]s has priority whilst we are below the low threshold and there are still messages to send (i.e. on the appropriate Qfor_EE queue). This section of code always attempts to send at least one [R1]/[E1] even when above the high threshold.

When the [R3]/[E1] is sent to the FI_EE, the R1ToA3 is included on Qt. This Qt queue is local to the thread and is used to check for timeouts. Each local Qt is potentially a number of separate queues with one for each potential timeout value (allowed range provided via the registry). This allows the [R3]/[E1] to be appended to the end of the timeout queue and for the oldest (for [R3]/[E1] timeout purposes) to always be at the head of that queue.

An [R3] may be failed without sending it to the SSE if it has been in this agent for too long (`Timeout.WaitToSend`, currently 18 seconds). Such messages are placed on the Qfrom_ee queue with a trigger of Failed_To_Send_R3, so that there will be no expectation of trying to reverse it. An [E1] may also be failed without sending it to the SSE if it has been waiting for a socket connection for too long (same timeout). This time the trigger is Failed_To_Send_E1, so that the count of reversal attempts is corrected.

A check is made on the performance of each socket, and if it falls below the acceptable level it is marked as Unresponsive, and when all current messages have been processed – had replies or been timed out - so that there are no outstanding responses the connection is aborted. It will be reconnected again in the normal way.

The socket performance is monitored over periods of SSEPerfPeriod msecs (default 2 mins), and is considered unacceptable if for two consecutive periods:

1. Number_of_timeouts >= (SSEPerfTOpercent * Number_of_Messages_Sent) / 100

2. Number_of_Messages_Sent >= SSEPerfThreshold

Where by default SSEPerfTOpercent = 40 and SSEPerfThreshold = 8.

Note that marking a socket as Unresponsive does prevent the PreEE thread from allocating further messages to the socket (see [ REF _Ref210812244 \r \h ]).

### 2.3.9.5 Socket and SSE Handler Availability Controls

It is only the Active agent that makes connections to a PI. The Standby agent will not attempt to connect to the PI until it has become Active, which is only after the currently Active agent has stood down.

Within an SSE Handler, the SSE is only considered available when there is at least one connection to the SSE. SSE availability is dependent on socket availability.

Each socket can be in one of the following states: Uninitialised, Available, Failed or For_Disconnection. It also has a connection/disconnection time associated with it.

Making sockets Available/ Uninitialised is devolved to an AsyncIO thread. Whilst the EE_IO thread is active or down, the AsyncIO thread tries to maintain the socket connections by frequently checking that they are in a Connected state. Those that are not are handled as follows.

| Current state | Target state | Comment |
|---|---|---|
| Failed | Uninitialised | Disconnects and sets the timestamp. |
| For_Disconnection | Uninitialised | Issues a shutdown request and disconnects after a short delay. Sets the timestamp. |
| Uninitialised | Connected | Only attempted a configurable 2 seconds (by `SOCKET_CLOSE_ALLOWANCE`) after the disconnect (to allow some time to detect other sockets failing). In addition there is a configurable delay (`All.MinConnectDelay`) imposed between any two connections to avoid overloading the X.25. On successful connection it sets the timestamp. |

**Table [ SEQ Table * ARABIC ]: Socket state changes from the AsyncIO thread**

During the normal calls of send and recv on the sockets the connection may be closed from the remote end or fail. In both cases the socket state is changed to Failed. In situations where the thread wants to close a socket (agent unavailable or FI closed) it changes the state to 'For_Disconnection'.

#### 2.3.9.5.1 Journalising Socket and SSE Handler Availability

As in the NBS, Authorisation agent changes in the availability of the SSE Handlers are reported identically to the Transaction journal and the Management journal. For DCS the SSE is regarded as available if the corresponding SSE handler (EE_IO thread) has a connection to the SSE and not otherwise.

The PI (aka SSE Handler) status records written are of journal type NMPI_STS with subtype Avail, Unavail_PI and Unavail_NBX.

The states of the individual socket connections are also reported. The journal type is TCP_STS and the subtypes are Avail and Unavail.

Each EE_IO thread keeps track of the changes in state of its connections, and interfaces with its Mngt thread to write the journal entries. It is only the Mngt thread that has a connection to the NPS.

The PI status may also need to be reported on behalf of a previous failed instance agent. As in NBS this is done when an agent starts up or becomes active by the Heartbeat thread.

### 2.3.9.6 Use of Network Management and Administrative Advice Messages

These are not required for DCS.

#### 2.3.9.6.1 EOD Cutover

This is not required for DCS.

### 2.3.10 Qfrom_EE queue

The Qfrom_EE queue is the input queue to the PostEE threads. It is in the same style as the QinX queue in that it is not a single queue but a number of queues, each with its own critical section. There is one queue per PostEE thread so that, in principle, each PostEE thread has its own queue. However, in practice, any PostEE thread can access any of these queues and will do so whenever its own queue is empty or it cannot get immediate access to its own queue (i.e. the critical section is currently in use elsewhere).

The Message Sequence Number associated with the R1ToA3 entry is used to balance the load when appending entries to these queues. If the intended queue cannot be accessed (critical section in use elsewhere) the entry can be queued on any available queue.

Each R1ToA3 entry represents one of the following. The entry contains the Trigger and a value for Current_Status.

The first table shows the entries which actually go to the PostEE thread. The second shows entries which go directly to the Exceptions thread instead. (The HLD allows some flexibility as to which processing is done in the PostEE and Exceptions thread and it is convenient to describe the treatment of these cases in the PostEE section.)

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| A1 | EE_IO | [A1] received before timeout and matched to the R1ToA3 entry | R1 | |
| E2 | EE_IO | [E2] received before timeout and matched to the R1ToA3 entry | Reversal | |
| *Marked for 'reprocessing'* | Exceptions | Any of the above | | See [ REF _Ref73333853 \w \h * MERGEFORMAT ] [ REF _Ref73333853 \h * MERGEFORMAT ] |

Table [ SEQ Table * ARABIC ]: R1ToA3 entries on the Qfrom_EE queues

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| Orphan_A1_E2 | EE_IO | [A1]/[E2] from SSE not matched to an R1ToA3 entry *(there is no concept of late [A1]/E2]s [for DCS)* | – | |
| Failed_To_Send_R3 | EE_IO | [R3] that has not been forwarded to the FI_EE | R1 | The timeout period has normally expired before this happens, but this may not always be the case |

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| A1_Timeout | EE_IO | [R3], no [A1] received before timeout | R1 | |
| Failed_To_Send_E1 | EE_IO | [E1] that has not been forwarded to the FI_EE | Reversal | The timeout period has normally expired before this happens, but this may not always be the case |
| E2_Timeout | EE_IO | [E1], no [E2] received before timeout | Reversal | |

Table [ SEQ Table * ARABIC ]b: R1ToA3 Sent to Exceptions Thread instead

### 2.3.11 PostEE threads

The PostEE thread processes the next [A1]/[E2] from the Qfrom_EE queue according to the status of the transaction within the NPS.

The PostEE thread is the thread which processes all non orphan responses from the SSE. It determines whether it indicates an acceptance or a decline and adjusts the trigger accordingly. The thread validates the supplied message and handles some of the state changes shown in section [ REF _Ref119816376 \r \h * MERGEFORMAT ]. For the common state changes it writes the appropriate Status and Journal entries to the NPS. The R1ToA3 entry may then be passed on to the ReplyA3 thread (to build an [A3] for return to the counter). An R1ToA3 entry that has failed to be sent to the FI (timeout or bad message) and other less common state changes are put on the Qexc queue to be processed by the Exceptions thread (to which section [ REF _Ref119816376 \r \h * MERGEFORMAT ] equally applies). Any update of status must only succeed if it is the expected change of state (the expected state change may be assumed on the basis of the last known state). Where an update fails, the Oracle transaction is aborted and this thread's processing is immediately retried using the newly found current state. Unexpected status changes are either ignored (R1ToA3 entry is released back to the pool of free beads) or passed onto the Exceptions thread for handling there.

#### 2.3.11.1 Reprocessing an R1ToA3 entry

An R1ToA3 entry may need reprocessing following a failure in the original processing. This can occur both internally within the thread and following requeuing by the Exceptions thread. A full discussion is given in [ REF _Ref73333853 \r \h * MERGEFORMAT ].

The 'for reprocessing' flag can indicate one of the following:

- unexpected state
- NPS failure, not committed
- NPS failure, commit status unknown

If the R1ToA3 entry is marked 'for reprocessing' on entry to the thread, the original entry conditions have to be reinstated. I believe, in practice, this is confined to the original trigger value, which has to be retained in the R1ToA3 entry for this purpose.

To establish the actual state of the transaction, its Status entry has to be read from the Status table. If there is none, then the Current Status becomes No_Status_Entry.

If the 'for reprocessing' flag is other than 'NPS failure, commit status unknown', the entry can simply be reprocessed but this time using the freshly established Current Status.

#### 2.3.11.1.1 Reprocessing when commit status is unknown

For 'NPS failure, commit status unknown' when there is an SSE reply (as distinct from an 'event'):

- The [A1] or [E2] whether orphan or not is reprocessed. However, in those cases in the state table (see [ REF _Ref73353387 \r \p \h * MERGEFORMAT ] and [ REF DCSHLD \h ]) where a duplicate would be audited, as {A1|Dup} or {E1|Rpt}, the auditing is bypassed. These are marked is the state tables with a "†".

Reprocessing of 'events' in the PostEE thread when the commit status is unknown could be problematical. Therefore, all the Status and Journal updates associated with events are farmed off to the Exceptions thread, where the recovery from such a condition is deterministic.

### 2.3.11.2 The 'event' triggers

If the trigger on the R1ToA3 entry is Failed_To_Send_R3, A1_Timeout, Failed_To_Send_E1 or E2_Timeout, the entry is essentially an 'event' rather than a message. These triggers are processed according to the state tables (see [ REF _Ref73353387 \r \p \h * MERGEFORMAT ] and [ REF DCSHLD \h ]).

If the trigger is Failed_To_Send_E1, the E1_Count is decremented so that this count reflects only those [E1]s that have actually been forwarded to the FI_EE (though they may not have been received there).

### 2.3.11.3 Classifying the message

If the trigger is Orphan_A1_E2, then the EE_IO thread was unable to match the SSE message against its in-memory list of [R3]s and [E1]s, and will not even know whether it is an [A1] or [E2]. There is not sufficient information to identify a Status entry so all that can be done is to journalise the message. All SSE messages which could be thought of as late will be treated as Orphan_A1_E2s.

Otherwise the message needs to be classified as acceptance or a decline. The Acquirer Response Code and Message fields in the [A1] determine the classification according to the table of Response Code values in [ REF DCSHLD \h ]. Anything other than simple acceptance (Acquirer Response_Code 00) is treated as a decline.

The interpretation of these fields is not hard coded but is specified by the mapping supplied in the FI_RESPCD_MAP registry entry. The registry entry is in SAN syntax (described in GENHLD) where the keywords and their values have special meanings described below. The Heartbeat thread calls Prepare_SSE_RespCd() during initialisation and this creates an internal linked list of structures from the registry entry. The list is used to map the response code of each [A1] that is returned.

The keyword names are numeric in ascending order and each number may have none or several subsequent keywords with '_*n*' appended to the name i.e. successive keywords could be '1', '1_1', '1_2', '1_2_1' or '3', '3_1'.

The values of keywords that start with the same number are either expressions that are AND'd together or a value that is returned as the mapped response code. These are determined by matching the start of the value with the strings in the table below:

| | |
|---|---|
| "ArC=" | A lower case comparison of strings (stricmp) is made between the value following this text and the Acquirer Response Code in the [A1], thus leading zeroes are important |
| "AM=" | A lower case comparison of strings (stricmp) is made between the |

| | |
|---|---|
| | value following this text and the Message in the [A1], thus leading zeroes are important |
| "AM~" | The C function strstr is used to see if the value following this text occurs in the Message field in the [A1] |
| "Set:=" | The value following this text is returned as an integer value. It is the mapped response code |

Each series of 'same-number' keywords must have their last keyword value starting with "Set:=". The 'same-number' keywords prior to that are AND'd together; if the result is true then the mapped response code from their last keyword value just mentioned is returned. Otherwise the next set of 'same-number' keywords is used in the same way, and so on.

The value for `FI_RESPCD_MAP` (with white space added for clarity) is

```
{Data={
            1 = ArC=00;
            1_1 = Set:=01;
            2 = ArC=05;
            2_1 = AM~RETAIN CARD;
            2_1_1 = Set:=02;
            3 = AM~BAD FORMAT;
            3_1 = Set:=69;
            4 = Set:=08
      }
}
```

(This is directly equivalent to the Reference data used by the Horizon agent for the same purpose.)

A message can also be deemed as Bad if it fails certain validation rules – see [ REF _Ref72485763 \r \p \h * MERGEFORMAT ].

### 2.3.11.4 Validation and Message Mappings

The message mappings are as described in the [ REF DCSHLD \h ]. In DCS it is the PostEE that analyses the [A1] or [E2] message and extracts the individual fields.

If any errors are found the message will be processed according to the trigger Bad_A1 or Bad_E2, as appropriate – refer to the state change tables (see [ REF _Ref73353387 \r \p \h * MERGEFORMAT ] and [ REF DCSHLD \h ]).

There is no data from the [A1] that has to be retained for any [E1] subsequently generated for reversing the transaction.

(In the Horizon DCS agent the analysis of the [A1] was done in the same thread as building the [A3] reply for the counter. For HNG-X these operations will be separated out. PostEE will do the analysis of the [A1] and [E2], but the building of the [A3] counter reply will be done in the ReplyA3 thread described later.)

### 2.3.11.5 The Status entry and state changes

Access to the Transaction Status table uses YDDD, Terminal_Id and Trans_Num as the keys to select the required Status entry – this information will already be in the R1ToA3 entry. The processing is controlled according to the state transition tables given in [ REF DCSHLD \h ]. They include an indication as to which rules need to be performed in this thread and which in the exceptions thread. A documentary overview of how these tables are used in the PreEE, EE_IO and PostEE threads is given in section [ REF _Ref159737901 \r \h ] (Appendix VI).

Any update of the Status entry must only succeed if it is the expected change of state (i.e. no-one has updated the entry since the fetch). This is achieved by using a WHERE clause on the update checking that CURRENT_STATUS is as expected. If it is not in the expected state, the Status entry must be read (without locking) to find its current state and the message reprocessed accordingly – see [ REF _Ref73333853 \r \h * MERGEFORMAT ] for a fuller discussion.

### 2.3.11.6 Auditing / Journalising

Every entry in the Transaction Journal table contains a Message Part Sequence Number, which is 1 for the first entry for a transaction, incremented by one for each subsequent entry for that transaction. The sequence number used must be recorded in the Status entry. Therefore, every insert into the Journal must be preceded by an insert / update of the Status entry in the same commit unit. For a Status insert the sequence number is 1. Each update must contain a WHERE clause checking that the previous value of the sequence number is as expected.

The following table gives information on the Journal records. This should be read in conjunction with the spreadsheet in [ REF NBXJNL \h ].

| Journal Type & Subtype | Audited here or in Exceptions thread[8] | Condition | Data to be audited |
|---|---|---|---|
| {A1\|Acc} | *PostEE | [A1] Accept | [A1] fields |
| {A1\|Dec} | *PostEE | [A1] Decline | Ditto |
| {A1\|Bad} | Exceptions | Validation error in [A1] detected by PostEE thread *(Note that the EE_IO thread will already have detected and rejected most syntactic errors)* | Ditto |
| {A1\|Dup} | Exceptions | Duplicate [A1] | Ditto |
| {A1_E2\|Orphan} | Exceptions | True orphan [A1] or [E2] – no entry in Transaction Status table – as there is insufficient information to identify the a transaction. | Ditto |
| {A3\|No_R3} | *Exceptions | EE_IO thread was unable to forward the [R3]. Failure-[A3] required | [A3] fields<br>Could also include [R1] fields, again |

[8] Ones marked with an asterisk must be audited in the stated thread: those in the PostEE for performance reasons, those in the Exceptions thread for reprocessing reasons (see [ REF _Ref73337951 \r \h ]). The choice between the two threads in the other cases is advisory but not necessary for reasons of logic.

| Journal Type & Subtype | Audited here or in Exceptions thread[8] | Condition | Data to be audited |
|---|---|---|---|
| {A3\|A1_TmOut} | *Exceptions | [A1] has been timed out. Failure-[A3] required: Response_Code 32 | [A3] fields |
| {E2\|} | PostEE | [E2] Accept | [E2] fields |
| {E2\|Dec} | PostEE | [E2] Decline | Ditto |
| {E2\|Bad} | Exceptions | Validation error in [E2] detected by PostEE thread | Ditto |
| {E2\|TmOut} | *Exceptions | [E2] has been timed out | – |
| {C0\|No_E1} | *Exceptions | EE_IO thread was unable to forward the [E1]. | [E2] fields<br>Could also include [E1] fields, again |

**Table [ SEQ Table * ARABIC ]: Journal records for PostEE**

### 2.3.11.7 NPS exception handling

When writing to the Status and Journal files in a single commit unit, the following exceptions may occur:

- Unexpected Status value or Message Part Sequence Number
- Oracle error

Please see [ REF _Ref73333853 \r \h * MERGEFORMAT ] for a full discussion. As part of the recovery processing, the R1ToA3 entry may need to be reprocessed by this thread – see [ REF _Ref73355147 \r \h * MERGEFORMAT ].

### 2.3.11.8 Crypto

This thread does not invoke any cryptographic function.

### 2.3.11.9 Statistics

The statistics in the table below are collected for the FI_EE; they are not specific to any SSE Handler. They are collected per thread and accumulated across the thread class by the Heartbeat thread.

| Statistic | External name | Internal name | Description |
|---|---|---|---|
| Total elapsed time of [R3] / [A1] messages in the SSE | FITM | R3toA1Time | Milliseconds. The time measured is the accumulation of the Agent_SLA_Info, <AgtSLA>, attributes in the [A3]s. For timed-out [A1]s, this is the timeout time |

**Table [ SEQ Table * ARABIC ]: Statistics maintained by the PostEE threads**

Note that no equivalent statistic is maintained for the total elapsed time of [E1] / [E2] messages in the SSE.

### 2.3.12 Qa2 queue

The Qa2 queue is handled similarly to the Qin queue, with one queue for each ReplyA3 thread. Each ReplyA3 thread can access any of these queues and will do so whenever its own queue is empty or it cannot get immediate access to its own queue (the critical section is currently in use). On insertion the sequence number in the R1ToA3 entry is used to balance the load across the queues. If the intended queue cannot be accessed (critical section in use) the entry can be queued on any available queue.

The Qa2 queue is the input queue to the ReplyA3 thread. Each R1ToA3 entry represents one of the following. The entry contains the Trigger and a value for Current_Status.

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| A3 | PostEE or Exc | [A1] for formatting before sending to the OSR | R1 | In fact not all messages on Qa2 will have Trigger_A3 but will be left at their previous trigger. This does not matter because the ReplyA3 thread is not trigger driven. |

**Table [ SEQ Table * ARABIC ]: R1ToA3 entries on the Qa2 queues**

### 2.3.13 ReplyA3 threads

The ReplyA3 thread processes the next entry from the Qa2 queue and produces [A3] replies for the counter and queues them on the appropriate Qout queue. The entry is removed from Qa2.

This thread builds all the [A3]s be they for successful transactions, for failures of the FI_EE being Down or Closed (all sent from the PostEE thread) or for bad R1 messages (the latter sent from the Exceptions thread).

The message mappings are described in [ REF DCSHLD \h ] including the mapping for cases when there is no [A1].

If the RespCd value in the R1ToA3 entry is set (for Agent detected conditions such as timeout, badR1), the [A3] includes the <AgtDiag> attribute which contains diagnostic text.

If the [A3] is for FI_EE Closed, the AgtErr attribute is included (from details taken from the FI_EE Administrative Status object).

On completion the R1ToA3 entry is moved to the single QoutX queue for the GetR1X thread.

#### 2.3.13.1 Transaction Journal entries, Transaction Status table

This thread does not access the NPS at all.

#### 2.3.13.2 Crypto

None for DCS.

#### 2.3.13.3 Statistics

The statistics in the table below are collected for the FI_EE; they are not specific to any SSE Handler. They are collected per thread and accumulated across the thread class by the Heartbeat thread.

| Statistic | External name | Internal name | Description |
|---|---|---|---|
| None | | | |

**Table [ SEQ Table * ARABIC ]: Statistics maintained by the ReplyA3 threads**

### 2.3.14 Qstatus Queue

This is as for the NBS Authorisation agent, see [ REF NBSLLD \h ].

This queue contains [T1] messages from GetR1X which will be processed by Process Status Message thread.

### 2.3.15 Process Status Message Thread

This is as for the NBS Authorisation agent, see [ REF NBSLLD \h ].

This thread processes the messages from the Qstatus queue. The thread will retrieve the current status for the specified transaction from the Status table (TMS_TX_NBX_STATUS) in the NPS. A [T2] response will be built and message will be placed on the QoutX to be picked up by GetR1X thread. See [ REF GENHLD \h ] for the layout of these messages.

### 2.3.16 Qout queue

This is not used by the DCS Authorisation agent.

### 2.3.17 QoutX queue

This is as for the NBS Authorisation agent, see [ REF NBSLLD \h ].

The QoutX queue is the queue whereby [A3] messages are passed back to the GetR1X for returning the [A3] to the same OSR connection they came in on, and hence back to the counter.

The QoutX queue is in the same style as the QinX queue in that are a number of queues with a separate one for each receiving thread. However in this case each GetR1X processing thread only has access to its own queue. The queue is chosen on the basis of the originating GetR1X thread. Should a critical section already be in use elsewhere there is a null sleep (gives other threads a go) before obtaining the critical section (which may then involve a wait). In fact there is only one GetR1X thread, and so one QoutX queue.

| Trigger | Queued by | R1ToA3 entry | Hash Table (TxnQ) | Comments |
|---|---|---|---|---|
| A3 | ReplyA3 | [A3] for sending to the OSR | R1 | |
| *For_Discard* | PreEE | | any or none | Queued here just to free the R1ToA3 entry |

**Table [ SEQ Table * ARABIC ]: R1ToA3 entries on the QoutX queues**

### 2.3.18 Qgrev queue

Not required for DCS.

### 2.3.19 Grev (Guaranteed Reversals) thread

Not required for DCS.

### 2.3.20 Qrev queue

Not required for DCS.

### 2.3.21 E1Retry (Reversals Management) thread

Not required for DCS.

### 2.3.22 Qexc queue

The Qexc queue is the input queue to the Exceptions thread. It is a normal queue where entries are appended to the end of the queue and any change to the queue is controlled by critical section.

### 2.3.23 Exceptions thread

There is precisely one Exceptions thread in an Agent. It is always active as it is allowed to lag behind the rest of the threads when tidying up.

If the thread runs out of work it has a short wait (1000ms) on the agent's stop event before trying again.

The thread accesses the NPS database. Performance is less critical than with some other threads, so there is no reserve thread acting as a hot standby. Whenever a connection is deemed to be lost, the thread replaces the connection with one to the alternative Oracle instance.

This thread is responsible for handling various scenarios where the message processing has deviated from the standard. It handles most (if not all) of the less common state changes (including unsupported state changes), and for these it writes the appropriate Status and Journal entries to the NPS. For ease of documentation and to aid understanding, these have been included in the descriptions and state change tables of the PreEE and PostEE threads in [ REF DCSHLD \h ].

After these actions the R1ToA3 entry is sent on to another thread, as follows, or released back into the pool of free beads:

a) The R1ToA3 entry may be passed to the ReplyA3 thread so that a Failure-[A3] message is generated. *(This is unlike earlier Agents where this thread built the message and did its own signing. The reason is that the PreEE threads may produce sufficient volumes of [R3]s which it cannot forward to the EE_IO threads such that a single Exceptions thread may not be able to cope – under the conditions this occurs, the ReplyA3 wouldn't have much, if any, normal [A3] traffic to process.)*

b) If the NPS actions failed because of a lost connection the R1ToA3 entry is put back on the Qexc queue so that is can be reprocessed once a connection is re-established.

c) Otherwise, if the update to the Status table failed, the R1ToA3 entry is returned to the thread that put it onto the Qexc queue for reprocessing, as the most likely cause of this is that the information in the Status table has legitimately been changed by another thread. The thread to return it to is saved in the R1ToA3 entry (it may be the PreEE or PostEE thread) but if none is recorded, the entry is released back into the pool of free beads. *(This latter case is an unexpected error after an error – it is the Exceptions thread after all – so to keep it simple we give up on this R1ToA3 entry!)*

Periodically (15000ms configurable via THREADCONTROL.Heartbeat.NBEAdminPoll registry entry) the thread reads the NPS_SYSTEM_PARAMETERS table to see if there is a change to the Administrative Status of the FI (column `PARAMETER_NAME`="NBX_ADMIN_STATUS"). If the sequence number of the entry for NBX_ADMIN_STATUS is greater than the value stored by the agent, the thread updates global variables associated with the Open or Closed status of the FI (e.g. `New_AdminObj_To`) and the saved sequence number. The Heartbeat thread will act on these values. The document [ REF NBSLLD \h ] describes the global variables and the actions of the Heartbeat thread, in particular see the equivalent section to this one and the Heartbeat Thread section in that document.

#### 2.3.23.1 NPS exception handling

When writing to the Status and Journal files in a single commit unit, the following exceptions may occur:

- Unexpected Status value or Message Part Sequence Number
- Oracle error

Please see [ REF _Ref73333853 \r \h * MERGEFORMAT ] for a full discussion.

#### 2.3.23.2 Crypto

This thread does not invoke any cryptographic function.

#### 2.3.23.3 Statistics

The statistics in the table below are collected for the FI_EE; they are not specific to any SSE Handler.

| Statistic | External name | Internal name | Description |
|---|---|---|---|
| Count of Stale [R1]s and [C0]s | STAL | R1C0StaleCount | Counted in this thread for [C0]s only. Also counted by other threads |

**Table [ SEQ Table * ARABIC ]: Statistics maintained by the Exceptions thread**

### 2.3.24 Command thread

As in NBS the DCS Authorisation agent has a Command thread.

This thread reads the NPS_OPERATOR_COMMANDS table to find commands targeted at the active DCS agent. These are entries which have their FI_EE column set to 'DCS'.

The DCS agent only supports commands related to administrative closure of the SSE. These are ADMIN_OPEN, ADMIN_CLOSE and ADMIN_HALF_CLOSE as listed in [ REF GENHLD \h ]. The thread and the commands are implemented as described in [ REF NPSHLD \h ].

### 2.3.25 Heartbeat threads

*See [ REF _Ref74103009 \h * MERGEFORMAT ]*

### 2.3.26 EE_Probe thread

Not required for DCS.

### 2.3.27 NPS exception handling for PreEE, PostEE and Exceptions threads

There is a fundamental difference in the NPS handling between the PreEE and PostEE threads on the one hand and the Exceptions thread on the other. The former both have a reserve thread for accessing the database via an alternative Oracle instance, the latter does not and performs recovery of an NPS connection internally. This difference is exploited in the exception handling when the NPS Status and Journal updates do not go smoothly.

As described earlier, the NPS updates that the PreEE and PostEE threads performs are all of the same pattern: a Status entry insert / update, a Journal insert and possibly an (SSE) Message Number update, followed by a commit. These updates are tied together by a Message Part Sequence Number. The majority (in volume terms) of these are performed by these threads, but some, indeed virtually all, of the more abnormal ones are performed by the Exceptions thread on behalf of the main thread.

Note that formal Oracle transactions are not needed.

#### 2.3.27.1 Unexpected state or Message Part Sequence Number

Again as already described, every insert / update to the Status entry is protected by WHERE clauses on the expected CURRENT_STATUS and expected MESSAGE_PART_SEQ_NO fields. If the insert / update fails by virtue of either of these fields not having the expected value, a dummy commit is done to free resources (though there is nothing to actually commit), and the R1ToA3 entry is marked '**for reprocessing, unexpected state**'.

If this is within the Exceptions thread, the R1ToA3 entry is returned to the main thread's input queue (QinX for PreEE, Qfrom_EE for PostEE) for reprocessing. If this is within the main thread, it is not necessary to formally requeue the R1ToA3 entry; it is sufficient to start reprocessing from the point just after dequeuing the entry.

#### 2.3.27.2 Oracle errors

Oracle may return an unexpected error code on the inserts / update / commit. An error event is written to the NT Event Log – this identifies both the transaction and the Oracle error.

Agents traditionally divide such error codes into those that indicate a data error (such as a constraint violation) and those that indicate a possible operational problem. This division is done by a having a list of error codes that represent the latter. This list has been extended for the new resilient technology being employed for the NPS. If it is a data error (which should never occur), the processing of the R1ToA3 entry is terminated and the bead is freed.

The most problematical is when the Oracle error occurred on the commit – it cannot be known with certainty as to whether the commit has succeeded or failed. As the expected frequency of these occurrences is extremely small, the recovery action should not be over-engineered.

##### 2.3.27.2.1 Recovery in the PreEE and PostEE threads

If the Oracle error occurred within the main PreEE or PostEE threads, the recovery procedure is as follows:

a) The R1ToA3 entry is marked with '**for reprocessing, NPS failure, not committed**' or '**for reprocessing, NPS failure, commit status unknown**', as the case may be, and passed to the Exceptions thread.

b) The database connection is closed – this automatically aborts the database updates – and re-opened, all accompanied by appropriate changes in thread status. Meanwhile the Heartbeat thread may have made the reserve thread active.

c) The Exceptions thread notices the 'for reprocessing, NPS failure' flag, and simply returns the R1ToA3 entry to the main thread's input queue (QinX for PreEE, Qfrom_EE for PostEE) for reprocessing.

### 2.3.27.2.2 Recovery in the Exceptions thread

If the Oracle error occurred within the Exceptions thread, the recovery procedure is different. The Exceptions thread handles its own recovery of the database connection.

a) The R1ToA3 entry is marked with '**for reprocessing, NPS failure, not committed**' or '**for reprocessing, NPS failure, commit status unknown**'.

b) The database connection is closed – this automatically aborts the database updates – and re-connects to the other Oracle instance.

c) If the failure occurred before the commit, the database inserts / updates are simply repeated.

d) If the failure was on the commit, the R1ToA3 entry is returned to the main thread's input queue (QinX for PreEE, Qfrom_EE for PostEE) for reprocessing.

### 2.3.27.3 Reprocessing in the PreEE and PostEE threads

If the R1ToA3 entry is marked 'for reprocessing' on entry to the thread, the original entry conditions have to be reinstated. I believe, in practice, this is confined to the original trigger value, which has to be retained in the R1ToA3 entry for this purpose.

The 'for reprocessing' flag can indicate one of the following:

- unexpected state
- NPS failure, not committed
- NPS failure, commit status unknown

To establish the actual state of the transaction, its Status entry has to be read from the Status table. If there is none, then the Current Status becomes No_Status_Entry.

If the 'for reprocessing' flag is other than 'NPS failure, commit status unknown', the entry can simply be reprocessed but this time using the freshly established Current Status.

Reprocessing when the commit status is unknown is specific to the main thread.

- For the PreEE thread see [ REF _Ref73337894 \r \h * MERGEFORMAT ].
- For the PostEE thread see [ REF _Ref73337951 \r \h * MERGEFORMAT ].

## 2.4 Active/Standby Agent Resilience Overview

*See [ REF _Ref74103009 \h * MERGEFORMAT ]*

## 2.5 Implementation Details

The agent is implemented as a service.

Each thread in [ REF _Ref120678465 \h * MERGEFORMAT ] is to be defined as a separate class in accordance with the structures described in Section [ REF _Ref107200571 \w \h * MERGEFORMAT ].

### 2.5.1.1 R1ToA3 entries

The R1ToA3 entry consists of a Bead_Type part that includes:

| Field | Type | Description |
|---|---|---|
| Current_Status | unsigned long | A value defining status of message in Bead e.g. R3_Financial |
| Requested_Status | unsigned long | New state to be put in to NPS Status table |
| Trigger | unsigned long | A value defining what the R1ToA3 entry currently represents, e.g. R1, A1_Timeout, Failed_To_Send_E1 |
| ReqType | int | Values for IsR1Msg, IsC0Msg, IsT1Msg and IsRevMsg |
| YDDD | unsigned char[5] | Receipt_Date from [R1] / [C0] |
| TerminalId | unsigned char[9] | ggggggnn – from FAD code and counter id |
| TransNum | unsigned char[7] | From HTxnNum in [R1] / [C0] |
| Q_Index | unsigned long | Index of free queue pool used. |
| Cluster_Id(also ClusterId) | unsigned long | Always 5 for DCS |
| Routing_Agent_Tsmp | unsigned char[21] | From OSR |
| Arrival_Time | unsigned long | Time in mSecs that message was read |
| EE_Timeout | unsigned long | FI processing timeout from [R1]. |
| EE_Time | unsigned long | Time in mSecs sent while at FI_EE, Duration in mSecs of FI_EE processing thereafter |
| RespCd | unsigned long | Response code value for agent-detected conditions |
| Mapped_RespCd | unsigned long | Streamline response code mapped to an Agent value |
| NoCrypto | BOOL | Development diagnostic attribute NoCrypto present |
| AgtDiag | unsigned char[41] | Diagnostic text to be put into [A3] when RespCd is set for an agent-detected condition, e.g. timeout, badR1 |
| Message_Part_Seq_No | unsigned long | Sequence number used in 'where' clause when a row in the NPS Status table is updated. Set to 0 when row created then incremented on each update. |
| RRN | unsigned char[13] | From [R1] / [C0] |

| Field | Type | Description |
|---|---|---|
| PAN | unsigned char[20] | Clear PAN from [R1] / [C0] |
| HPAN | unsigned char[20] | Hashed PAN from [R1]/[C0] |
| PANPK | unsigned char[29] | Encrypted PAN, encrypted by agent |
| HTxnNum | unsigned char[64] | From [R1] / [C0] |
| SocketIndexUsed | unsigned int | Socket used when message sent to the FI |
| SeqNo | unsigned long | Approximate sequence number used to distribute messages across queues |
| Process_Queue | Link_Type | On one of queues named in Figure 2 e.g. Qfor_EE queue. |
| Timeout_Queue | Link_Type | Not used |
| Hash_Key_Link | Hash_Key_Link_Type | Key and link within hash chain |
| Message | `Message*(see [ REF XMLLLD \h ])` | Current message. Contains R1, ,A3, or C0 message depending on State |
| SocketBuffer | unsigned char [2048] | Current message. Contains R3, A1, E1 or E2 message depending on State |
| Mid | unsigned char [16] | MID |
| Tid | unsigned char [9] | TID |
| TIDMessageNum | unsigned char [9] | SSE Message Number, from extended MID/TID table |
| ReferralSupported | boolean | From [R1] |
| *etc* | | Other fields include:<br>a) values written to or read from the NPS Status table.<br>b) values required for NPS Journal |

**Table [ SEQ Table * ARABIC ]: Bead_Type structure**

A Link_Type identifies the list it is on and includes Next and Prev pointers for the chain it is on. A list is headed by a List_Type, which consists of a critical section and a dummy Link_Type. This allows a link to be removed from a list without having to know which list it is on. Each Link_Type indicates if it is currently on a queue or not and the State value for the queue it is on (automatically set from the State on the dummy Link_Type). This State is automatically incremented when the entry is removed from the queue.

A Hash_Key_Link_Type contains the key and includes Next and Prev pointers.

Possible Process_State values for a queue are:

| State's enum sequence | Comment |
|---|---|
| Free_Q = 0 | Initial setting and then an automatic setting on return to queue |
| PostFree | Automatic change on removal from Free queue |
| Qin_Q | Automatic change on insertion in Qin queue |
| PostQin | Automatic change on removal from Qin |
| Hold_Q | Automatic change on insertion in Hold_Q |
| PostHold | Automatic change on removal from Hold_Q |
| Qverified_Q | Automatic change on insertion in Qverified |
| PostQverified | Automatic change on removal from Qverified |
| Qfor_ee_Q | Automatic change on insertion in Qfor_EE queue |
| PostQfor_ee | Automatic change on removal from Qfor_EE |
| EETimeout_Q | Automatic change on insertion in Qt queue |
| Post EETimeout | Automatic change on removal from Qt queue |
| Qfrom_ee _Q | Automatic change on insertion in Qfrom_EE queue |
| PostQfrom_ee | Automatic change on removal from Qfrom_EE queue |
| Qtobesigned _Q | Automatic change on insertion in Qa2 queue |
| PostQtobesigned | Automatic change on removal from Qa2 queue |
| Qout _Q | Automatic change on insertion in Qout queue |
| PostQout | Automatic change on removal from Qout queue |
| Exceptions_Q | Automatic change on insertion in Qexc queue |
| PostExceptions | Automatic change on removal from Qexc queue |
| NF_Q | Automatic change on insertion in Nearly Free queue |
| PostNF | Automatic change on removal from Nearly Free queue |
| QinX_Q | Automatic change on insertion to QinX |
| PostQinX | Automatic change on removal from QinX |
| Qstatus_Q | Automatic change on insertion to Qstatus |
| PostQstatus | Automatic change on removal from Qstatus |

**Table [ SEQ Table * ARABIC ]: Process States for queues**

#### 2.5.1.2 Queue management

This is as for the NBS Authorisation agent (see [ REF NBSLLD \h ]).

### 2.5.1.3 Socket usage overview

*(A brief introduction is given in [ REF _Ref74372478 \r \h * MERGEFORMAT ].)*

This is mainly as for the NBS Authorisation agent (see [ REF NBSLLD \h ]). Differences are identified below.

For DCS the FI_EE is the SSE which is treated as if it had 4 SSE Handlers (`EE_IO.Count=4`). Each SSE Handler has only one virtual address, and in fact each SSE Handler will use the same virtual address.

For DCS there is no requirement that an [E1] must be sent to the same SSE Handler as the [R3] it is reversing, so these will be load balanced across the individual sockets in the same way as the [R3]s are.

The DCS Authorisation agent normally uses the Record Boundary Preservation protocol on the messages it sends and receives. It can also be configured to run without this protocol by the `RECORD_PRESERVATION` registry parameter. This protocol and its use are described below.

#### 2.5.1.3.1 Running with the Record Boundary Preservation (RBP) Protocol

If this protocol is in use then all messages, input and output have the 6 byte Record Boundary Preservation header.

| | |
|---|---|
| byte 0 | 0xD7 |
| byte 1 | 0x4A |
| byte 2 | most significant byte of length (exluding this header) |
| byte 3 | least significant byte of length (exluding this header) |
| byte 4 | 0x00 (if final fragment of message) or 0x01 (otherwise) |
| byte 5 | 0x00 |

**Sending**

Messages are sent as a single fragment with a header that has byte 4 set to 0x00.
The message after the header will begin with an STX and end with an ETX.

**Receiving**

Data is read 128 bytes at a time and assembled into messages.
An RBP header is expected and messages are synchronised on the first byte of the header i.e. 0xD7.
Fragments are assembled, with any data between fragments being discarded, until a final fragment has been found.
A check is made for the STX which should start the message, and the ETX which should end it, any excess data being discarded.
The data from the STX to the ETX inclusive constitutes the message and is processed.

#### 2.5.1.3.2 Running without the Record Boundary Preservation (RBP) Protocol

**Sending**

Messages have no header and are sent as a single send.
They start with an STX and end with an ETX.

**Receiving**

Data is read 128 bytes at a time and assembled into messages.
There is no header to give the length of the message they are recognised as starting with an STX and ending with an ETX.
Data between an ETX and an STX is discarded, as is data between 2 STXs without an intervening ETX.

### 2.5.1.4 QThread_Type Thread Structure

Each thread class has a thread initialisation function associated with it (e.g. Initialise_R1X_QThreads). When called this function creates and initialises an array with a QThread_Type entry for every thread in that class. Tailoring of this configuration is via the THREADCONTROL registry setting. Each QThread_Type entry includes:

- Thread identification information
- Details on the requested and actual states plus control information for changing states
- Functions to be used as hooks for the class
- Various queues for use by the thread
- Private thread information including statistic counts
- NPS Journal control information.

### 2.5.1.5 Inter-thread management

This is as for the NBS Authorisation agent (see [ REF NBSLLD \h ]).

## 2.6 Test Facilities

## 2.6.1 Testing options for volume and performance testing

This is not required for DCS.

## 2.7 Protocols

### 2.7.1 Protocol between the OSR and DCS Authorisation Agent

This is described in [ REF GENHLD \h ].

### 2.7.2 Probe Interface to the NBX Authorisation Agents for LINK

Not required for DCS.

## 2.8 Message Formats

The formats of the following messages are defined in [ REF MSGIFS \h ]

- [R1]
- [A3]
- [C0]
- [T1]/[T2]

## 2.9 NPS Tables

The formal definitions of the tables in the NPS are in the NBX Persistent Store HLD [ REF NPSHLD \h ]. Some are reproduced here for the convenience of the reader.

The tables used by the Authorisation Agents are accessed via common synonyms which maps on to a table specific to the agents identity. The table used is determined by the username the agent logs on as i.e. for DCS where there is a separate table per logical Authorisation Agent the actual table name ends with _DCS. This document always refers to the common synonym name. The tables of interest are:

| Table | Table or synonym | Comments |
|---|---|---|
| Transaction Status | TMS_RX_TXN_STATUS | Separate table per logical Authorisation Agent |
| C0 Reversals | TMS_RX_C0_REVERSALS | NOT USED BY DCS |
| Transaction Journal | TMS_RX_TXN_JOURNAL | Separate table per logical Authorisation Agent |
| Management Journal | TMS_RX_MGT_JOURNAL | One table written to by all Authorisation Agents |
| NBX Configuration | TMS_TX_NBX_CONFIGURATION | One read-only table accessed by all Authorisation Agents |
| NBX Heartbeats | TMS_RX_NBX_HB | Separate table per logical Authorisation Agent |

| Table | Table or synonym | Comments |
|---|---|---|
| NBX Heartbeats History | TMS_RX_NBX_HB_HISTORY | One table written to by all Authorisation Agents |
| NBX Statistics | TMS_RX_NBX_STATS | One table written to by all Authorisation Agents |
| NBX PI Statistics<br>(PI aka SSE Handler) | TMS_RX_NBX_PI_STATS | One table written to by all Authorisation Agents |
| NPS System Parameters | NPS_SYSTEM_PARAMETERS | One table accessed by all Authorisation Agents |
| NBX Operator Commands | NPS_OPERATOR_COMMANDS | One table accessed by all Authorisation Agents |
| Message Numbers<br>(included in R3s and E1s) | TMS_RX_TID_MESSAGE_NUM | One table for each Authorisation agent that use TIDs (i.e. DCS, ETS) |

**Table [ SEQ Table * ARABIC ]: NPS tables accessed by a DCS Authorisation Agent**

### 2.9.1 TMS_RX_TXN_STATUS

The **Transaction Status table** is used by Authorisation Agents to create and maintain the status of all the transactions. There is a separate instance of this table for each logical Authorisation Agent and the DCS agent has its own table. The table name ends in '_DCS' but NPS defines a synonym without the suffix; the agent never uses the suffixed name. The table is solely for use by the Authorisation Agent, though housekeeping is performed by the NPS.

The columns YDDD + TERMINAL_ID + TRANS_NUM will form the primary key of the table. Almost all Agent access is directly through the primary key. All [R1], [C0] and [T1] messages contain the required key information.

NB. [T1]s require read access to the TMS_RX_TXN_STATUS table but do not cause the table to be changed.

| Primary Key | Derivation for [R1], [C0] | Derivation for [A1], [E2] |
|---|---|---|
| YDDD | From Transaction_Receipt_Date (<LclDte>). For example, 4002 for 02-JAN-2004 and 0004 for 04-JAN-2010 | By matching R1ToA3 entry |
| TERMINAL_ID | Concatenation of 6-digit FAD code and 2-digit counter id (both zero-padded at left) | By matching R1ToA3 entry |
| TRANS_NUM | Last but one component of Horizon_Txn_Num, modulo 1 million, zero-padded at left to 6 digits | By matching R1ToA3 entry |

**Table [ SEQ Table * ARABIC ]: Key for identifying a Status entry for a Transaction**

The table will be partitioned on YDDD (Day of the Year). Transactions must be retained for at least five days, so at any one time there will be six partitions containing transactions. NPS housekeeping will remove older partitions and will also create new partitions for one or two days ahead. Details of partitioning are in [ REF NPSHLD \h ].

For efficiency of access, each partition in the table will be sub-partitioned on the hash value of TERMINAL_ID. [ REF NPSHLD \h ] states that there will be 64 sub-partitions (each containing approximately 22000 transactions).

For DCS there needs to be sufficient information in the table to recover reversals and to support [T1]/[T2] status enquires.

| Column Name | Null? | Type | Remarks |
|---|---|---|---|
| YDDD | NOT NULL | VARCHAR2(4) | Local (not UTC) |
| TERMINAL_ID | NOT NULL | NUMBER(8) | Group_Id and Node_Id can be extracted if required |
| TRANS_NUM | NOT NULL | NUMBER(6) | |
| HORIZON_TRANSACTION_ID | NOT NULL | VARCHAR2(32) | <HTxnNum> |
| RECEIPT_DATE | NOT NULL | DATE | Concatenation of <LclDte> & <LclTme> |
| CURRENT_STATUS | NOT NULL | NUMBER(4) | See [ REF _Ref71691153 \h * MERGEFORMAT ] |
| INSERT_TSMP | NOT NULL | DATE default SYSDATE | |
| MESSAGE_PART_SEQ_NO | NOT NULL | NUMBER(5) | See [ REF _Ref210643120 \w \h ] |
| SETTLEMENT_DATE | | DATE | Not used for DCS. |
| CLUSTER_ID | | NUMBER(2) | Set to 5 |
| STAN | | NUMBER(6) | Not used for DCS |
| PI_ROUTING | | VARCHAR2(24) | Internal identification of the SSE Handler and socket |
| R3_INFO | | VARCHAR2(1024) | See [ REF _Ref72737853 \r \p \h * MERGEFORMAT ]. |
| A1_INFO | | VARCHAR2(1024) | See [ REF _Ref72737898 \r \p \h * MERGEFORMAT ]. |
| REVERSAL_INFO | | VARCHAR2(1024) | Not used for DCS |
| ROUTING_AGENT_TSMP | | DATE | See [ REF _Ref213226908 \r \h ] below |
| E1_COUNT | | NUMBER(6) | Not used for DCS |
| AGENT_DATA | | VARCHAR2(128) | Reserved for future use |
| R3_TSMP | | DATE | Transmission timestamp (UTC) of [R3], used for checking lateness of Reversals |

**Table [ SEQ Table * ARABIC ]: Transaction Status record**

### 2.9.1.1 Fields in the Transaction Status record

#### 2.9.1.1.1 R3_INFO

Some Reversal data needed for an [E1] and/or a [T2] is retained in the R3_Info column in the Status table. The format is a list of SAN attributes e.g. {xx=nn;yy=mm}.

| Type | Value | Source | Size of field |
|---|---|---|---|
| AgtSLA | Duration of msg at FI in mSecs | Agent | unsigned int |
| AuthAmt | Amount field | [R1] or [C0] | 13 |
| BdC | Bureau de Change flag | [R1] or [C0] | 1 |
| CurrCd | Currency code | [R1] or [C0] | 4 |
| HPAN | Hashed PAN | [R1] or [C0] | 20 |
| MID | MID | MID/TID file | 9 |
| PANPK | PAN(PK) | Agent | 29 |
| S | Internal agent flag denoting that msg sent to e-pay | Agent | 2 |
| TID | TID | MID/TID file | 9 |
| TransNum | e-pay Message Number of [R1] | Agent/NPS | 7 |

**Table [ SEQ Table * ARABIC ]: R3_Info in Status records retained for Reversals or [T2]s**

#### 2.9.1.1.2 A1_INFO

Some data needed for a [T2] is retained in the A1_Info column in the Status table. The format is as a list of SAN attributes e.g. {xx=nn;yy=mm}.

| Type | Value | Source | Size of field |
|---|---|---|---|
| AuthAmt[9] | Amount | [A1] | 12 |
| AuthCd[ NOTEREF _Ref214784916 \h * MERGEFORMAT ] | Authorisation Code | [A1] | 10 |
| IS[ NOTEREF _Ref214784916 \h * MERGEFORMAT ] | ICC Response Data, subfield 2 | [A1] | 257 |
| IssResp | Acquirer Response Code | [A1] | 81 |
| Phone[10] | `REFERRAL_PHONE` registry value | [A1] | variable |

[9] Omitted if not present in the [A1]

| Type | Value | Source | Size of field |
|---|---|---|---|
| RespCd | SSE Response Code mapped to an Agent value | [A1] | unsigned int |
| X91[ NOTEREF _Ref214784916 \h * MERGEFORMAT ] | ICC Response Data, subfield 1 | [A1] | 33 |

**Table [ SEQ Table * ARABIC ]: A1_Info in Status records retained for [T2]s**

#### 2.9.1.1.3 REVERSAL_INFO

Not used for DCS.

#### 2.9.1.1.4 ROUTING_AGENT_TSMP

This timestamp records when the Online Services Router (OSR) first detected the message. If the current status reflects a reversal stage, the timestamp is for the reversal, otherwise it is for the [R1].

For an [R1] or [C0] received from an OSR, the OSR captures the time it read the message and passes it to the Authorisation Agent in a control field in the message itself.

### 2.9.2 TMS_RX_TXN_JOURNAL

The **Transaction Journal table** is used by the DCS Authorisation Agents to audit (log) all messages passed across the external interface to the SSE, and to log various significant events. The spreadsheet embedded in [ REF NBXJNL \h ] identifies the different classes of journal record.

There is a separate instance of this table for each logical Authorisation Agent, and so one for DCS. The Authorisation Agent inserts records into the table for the following purposes:

a) For auditing all messages passing across the external interface to the SSE. An NPS process will direct a copy of the data to the audit system. *(Note: For this reason, this document generally refers to 'auditing' records when referring to inserting a journal record.)*

b) As a feed into the Transaction Enquiry Service (TES). The spreadsheet embedded in [ REF TESELEM \h ] includes the fields populated for each class of journal record. A TES extraction process will extract data into the TES in near real-time. No such feed is currently planned for DCS. However when journalising an [R3] or [E1] the agent will journalise individual fields from [R1] or [C0] to avoid having to journalise the whole [R1] or [C0] message.

c) For support purposes.

In general, only the active Authorisation Agent inserts messages in to the Transaction Journal. The only messages inserted by the standby Agent are possible PI Unavailable messages during Agent start-up. Conceivably, there could also be transaction-oriented messages if transactions are still being flushed through the Agent after it has just stood down from being active; however, it is expected that the threads that could do this will have been deactivated.

[10] Included only when the [R1] contains a *Referral_Flag* element with a value of 1 and the [A1] Acquirer Response Code indicates a Referral (see [ REF DCSHLD \h ]).

The Authorisation Agent never reads or updates a record once written. No other process writes to the table, though housekeeping will be performed by NPS.

The TERMINAL_ID and the RETRIEVAL_REFERENCE_NUMBER fields identify transactions in NBX; the latter field contains both the YDDD and TRANS_NUM as subfields.

| Column Name | Null? | Type | Remarks |
|---|---|---|---|
| PARTITION_LOGICAL_DT | NOT NULL | DATE | See Journal Management |
| SUBPARTITION_ID | NOT NULL | NUMBER(3) | See Journal Management |
| LOGICAL_SUBPARTITION_ID | NOT NULL | NUMBER(6) | See Journal Management |
| FI_TYPE | NOT NULL | VARCHAR2(6) | 'DCS' |
| SERVICE_NAME | NOT NULL | VARCHAR2(32) | Leading "TMS" omitted |
| AGENT_HOST | NOT NULL | VARCHAR2(32) | |
| INSERT_TSMP | NOT NULL | DATE default SYSDATE | |
| JOURNAL_TYPE | NOT NULL | VARCHAR2(8) | See [ REF _Ref70736989 \r \p \h * MERGEFORMAT ] |
| JOURNAL_SUBTYPE | | VARCHAR2(8) | See [ REF _Ref70736989 \r \p \h * MERGEFORMAT ] |
| PI | | VARCHAR2(6) | PI (aka SSE Handler) name |
| MESSAGE_PART_SEQ_NO | | NUMBER(5) | |
| CURRENT_STATUS | NOT NULL | NUMBER(4) | For diagnostic purposes only. See [ REF _Ref71691153 \h * MERGEFORMAT ] |
| STANDBY_FLAG | | VARCHAR2(1) | "Y" if generated by standby agent, otherwise "N" |
| AGENT_DIAGNOSTICS | | VARCHAR2(512) | For human consumption |
| AGENT_EVENT_TSMP | | DATE | See spreadsheet in [ REF NBXJNL \h ] |
| RECEIPT_DATE | | DATE | Concatenation of <LclDte> & <LclTme> |
| MESSAGE_TSMP | | DATE | From Date and Time attributes in [R1] or [C0] |
| ROUTING_AGENT_TSMP | | DATE | See spreadsheet in [ REF NBXJNL \h ]. When present it contains the same value as in the Transaction Status (see [ REF _Ref213226908 \w \h ]) |
| AGENT_SLA_INFO | | VARCHAR2(128) | Milliseconds, derived from the tick count. See spreadsheet in [ REF NBXJNL \h ] |

| Column Name | Null? | Type | Remarks |
|---|---|---|---|
| HORIZON_RESPONSE_CODE | | VARCHAR2(128) | Response Code after mapping, <RespCd> |
| LATE_REVERSAL_STATUS | | VARCHAR2(1) | "Y" if [E1] reversal is deemed to be 'late', otherwise omitted. See [ REF _Ref72660003 \r \h * MERGEFORMAT ] |
| REVERSAL_DELAY | | NUMBER(6) | Seconds. See [ REF _Ref72660003 \r \h * MERGEFORMAT ] |
| HORIZON_TRANSACTION_ID | | VARCHAR2(128) | <HTxnNum> from [R1] or [C0] |
| CLERK_ID | | VARCHAR2(128) | <User> from [R1] or [C0] |
| CLIENT_ID | | VARCHAR2(128) | <ClientId> from [R1] or [C0] |
| ISSUER_SCHEME_ID | | VARCHAR2(128) | <IssSchId> from [R1] or [C0] |
| ROUTING_GATEWAY | | VARCHAR2(128) | <RtngGwy> from [R1] or [C0] |
| TRANSACTION_TYPE | | VARCHAR2(128) | <TranType> from [R1] or [C0] |
| AMOUNT_REQUESTED | | VARCHAR2(128) | <ReqAmt> from [R1] or [C0] |
| MESSAGE | | VARCHAR2(2048) | Sanitised version of the message being audited. This will normally be an [R3] or an [A1] but can also be an [R1] or a [C0]. |
| PAN | | VARCHAR2(128) | Hashed PAN, H(PAN) |
| ENCRYPTED_PAN | | VARCHAR2(128) | Encrypted PAN Block, (PAN)PK |
| RETRIEVAL_REFERENCE_NUMBER | | VARCHAR2(12) | Retrieval Reference Number |
| TERMINAL_ID | | VARCHAR2(8) | GroupId and NodeId |
| CURRENCY | | VARCHAR2(128) | Currency Code |

**Table [ SEQ Table * ARABIC ]: Transaction Journal record**

The Transaction Journal contains a separate column, in general, for each field that occurs in the transactional messages to the FI_EE. This is for the benefit of the TES extraction process that populates the various tables in TES, which in turn is for the benefit of TESQA, the TES Query Application. Though none of this is planned for DCS. The obvious exceptions are the fields that are sensitive, such as the card's (clear) PAN or Expiry Date (see [ REF DCSHLD \h ]). The only sensitive data is in the ENCRYPTED_PAN and this is encrypted. Please refer to the spreadsheet in [ REF NBXJNL \h ] for full details of these fields and for which message types these fields are relevant.

#### 2.9.2.1 Journal types and subtypes

Each journal record is categorised by its Journal_Type and Journal_Subtype. This two-level categorisation is primarily for the benefit of the TES extraction process, as the Journal_Type is mostly sufficient for it to determine how to process the record. Note that some Journal_Types do not require division into subtypes, and that a Journal_Subtype of "Rpt" (= Repeat) means that the record does not contribute to the business outcome of the transaction.

The list of Journal_Types is as follows. Other types may readily be introduced. Refer to the spreadsheet in [ REF NBXJNL \h ] for information on Journal_Subtypes.

| Journal_Type | Category | Remarks |
|---|---|---|
| R1 | [R1] message | The only [R1] messages that are audited in this category are those that would not otherwise be audited, i.e. those which are discarded because they are stale or a duplicate. |
| R3 | [R3] message | Audited immediately after generating the [R3], before it is transmitted to the FI_EE |
| A3 | Agent-initiated [A3] message | [R3]/[A1] timed out; or unable to generate the [R3] |
| A1 | [A1] message | Audited soon after receipt of the [A1] from the FI_EE |
| C0 | [C0] message | The only [C0] messages that are audited in this category are those where the [E1] cannot or should not be generated. Normally, the details of the [C0] are included in the log record for the [E1] |
| E1 | [E1] message | Audited immediately after generating the [E1] Reversal, before it is transmitted to the FI_EE |
| E2 | [E2] message | Audited soon after receipt of the [E2] Reversal Response from the FI_EE |
| EV | Events | Miscellaneous events |
| NMPI_STS | PI Status | Audited when the status of a PI (or SSE handler) changes status |
| TCP_STS | Connection Status | Audited when the status of a socket connection changes ststus |

**Table [ SEQ Table * ARABIC ]: Journal_Types in the Transaction Journal**

### 2.9.3 TMS_RX_MGT_JOURNAL

A single **Management Journal table** is shared by all Authorisation agents, and used by the DCS Authorisation Agents to audit (log) all PI (aka SSE Handler) and Connection Statuses, and to record external commands. The network management message entries are also written to the Transaction Journal but are targetted at a different audience: SYSMAN rather than TES. The spreadsheet embedded in [ REF NBXJNL \h ] identifies the different classes of journal record.

The entries which apply to DCS are identified by having FI_TYPE = 'DCS', and since there is only one logical DCS agent will have `ENQUIRY_ENGINE = 'DCS' as well.`

The TRANSACTION_ID is derived using an ORDERed (not CACHEd) Oracle sequence TMS_MGT_JOURNAL_SEQ. There will be a unique index on this column.

The columns present are those columns from TMS_RX_TXN_JOURNAL needed for network management messages, with a couple of extra columns as the table is shared between all Agent instances.

| Column Name | Null? | Type | Remarks |
|---|---|---|---|
| TRANSACTION_ID | NOT NULL | DATE | From Oracle sequence |
| ENQUIRY_ENGINE | NOT NULL | VARCHAR2(6) | "DCS" for DCS |
| FI_TYPE | NOT NULL | VARCHAR2(6) | "DCS" for DCS |
| SERVICE_NAME | NOT NULL | VARCHAR2(32) | Leading "TMS" omitted |
| AGENT_HOST | NOT NULL | VARCHAR2(32) | |
| INSERT_TSMP | NOT NULL | DATE default SYSDATE | |
| JOURNAL_TYPE | NOT NULL | VARCHAR2(8) | See [ REF _Ref70736989 \r \p \h * MERGEFORMAT ] |
| JOURNAL_SUBTYPE | | VARCHAR2(8) | See [ REF _Ref70736989 \r \p \h * MERGEFORMAT ] |
| PI | | VARCHAR2(6) | PI (aka SSE Handler) name |
| AGENT_DATA | | VARCHAR2(2048) | Type-value pairs of parameters, present for a few journal records |
| AGENT_DIAGNOSTICS | | VARCHAR2(512) | For human consumption |
| AGENT_EVENT_TSMP | | DATE | See spreadsheet in [ REF NBXJNL \h ] |
| CLERK_ID | | VARCHAR2(128) | Only for external commands |
| COMMAND | | VARCHAR2(50) | Only for external commands |
| FROM_TSMP | | DATE | Optional command parameter |
| TO_TSMP | | DATE | Optional command parameter |
| PARAMETER_TEXT | | VARCHAR2(200) | Optional command parameter |

### 2.9.4 TMS_RX_C0_REVERSALS

Not used for DCS.

### 2.9.5 TMS_TX_NBX_CONFIGURATION

The DCS authorisation agent will share the NBX Configuration Parameters table described in [ REF NBSLLD \h ] with the other Authorisation agents.

The entries relating to the DCS agent will be identified by having the APPLIES_TO column set to 'DCS'. They are listed in [ REF DCSHLD \h ].

The version of configuration parameter actually used for DCS is controlled by the NPX_CONFIG_VERSION entry in the NPS_SYSTEM_PARAMETERS table which has its USED_BY column set to 'DCS'. The configuration parameter value used will be the one with the highest value of PARAMETER_VERSION which does not exceed this NPX_CONFIG_VERSION value.

### 2.9.6 TMS_RX_NBX_HB

There will be a separate **Heartbeat table** for DCS, used by each DCS Authorisation Agent instance to communicate with its Partner, the other Agent instance in a Resilient Agent Pair.

These tables will be similar to those for the other Authorisation agents (see [ REF NBSLLD \h ]) but will have their ENQUIRY_ENGINE column set to 'DCS'.

### 2.9.7 TMS_RX_NBX_HB_HISTORY

The **Heartbeat History table** is used by the DCS Authorisation Agents in the same way as by the other Authorisation agents (see [ REF NBSLLD \h ]).

The entries which apply to DCS are identified by having ENQUIRY_ENGINE = 'DCS', as there is only one logical DCS agent.

### 2.9.8 TMS_RX_NBX_STATS

The **NBX Statistics table** is used by the DCS Authorisation Agents in the same way as by the other Authorisation agents (see [ REF NBSLLD \h ]) to record statistics that apply to an Agent as a whole, and in particular to the SSE as a whole.

Statistics apply to the particular agent instances given by SERVICE_NAME and AGENT_HOST.

### 2.9.9 TMS_RX_NBX_PI_STATS

The **NBX PI Statistics table** is used by the DCS Authorisation Agents in the same way as by the other Authorisation agents (see [ REF NBSLLD \h ]) to record statistics that apply to individual PIs or, for DCS, an SSE Handler.

Statistics apply to the particular agent instances given by SERVICE_NAME and AGENT_HOST. Within the agent instance the PI column identifies the statistics for the individual PIs.

### 2.9.10 NPS_OPERATOR_COMMANDS

The DCS Authorisation agents will share the single **Operator Commands table** used for passing operator commands to the Authorisation Agents (see [ REF NBSLLD \h ]).

The entries for the DCS agent will have the FI_EE column set to 'DCS' as there is only one logical DCS agent.

### 2.9.11 NPS_SYSTEM_PARAMETERS

The DCS Authorisation agents will share the single **NPS System Parameters table** used by the Authorisation Agents to hold persistent information that has to be passed from an active Agent instance to one that is taking over from it. The table is described in [ REF NBSLLD \h ].

The entries applying to DCS are identified by having USED_BY = 'DCS' as there is only one logical DCS agent.

The only entries in the table which are used by the DCS agent are NBX_CONFIG_VERSION and NBX_ADMIN_STATUS.

### 2.9.12 TMS_RX_TID_MESSAGE_NUM

This table is used to hold the latest Message Number value that has been used in a message sent to e-pay; there will be an entry for each TID (counter) for which a message has been sent.

Two authorisation agents (DCS, ETS) require a Message Number to be saved, each uses its own table, the name distinguished by the suffix _DCS and _ETS respectively. The agents access the table via the common synonym, as described in section [ REF _Ref210812355 \r \h ], which does not have the suffix.

| Column Name | Null? | Type | Remarks |
|---|---|---|---|
| TID | NOT NULL | VARCHAR2(8) | TIS, identifying a counter |
| MESSAGE_NUM | NOT NULL | NUMBER(12,2) | Last number used in [R3] sent to SSE |
| INSERT_TSMP | | DATE | |
| UPDATE_TSMP | | DATE | |

## 2.10 FI Type Message Formats

See the AIS

[ REF SSEAIS \h ]

and Mapping documents, currently

[ REF DCSHLD \h ]

### 2.10.1 Message Mapping

This section merely provides some additional notes.

#### 2.10.1.1 [R1] to [R3]

No further notes.

#### 2.10.1.2 [C0] to [E1]

In DCS, a [C0] is the only trigger for generating an [E1] and contains the values required for generating it.

#### 2.10.1.3 [A1] to [A3]

Prior to the mapping there is some validation and conversion of the data necessary. In particular the [A1] is checked for a correct Message Type and the RespCd needs to be derived from the [A1]'s Acquirer

Response Code field. The mapping for this derivation is described in [ REF _Ref210715022 \w \h ]; it is defined by the value of the registry parameter `FI_RESPCD_MAP`.

#### 2.10.1.3.1 RespCd Values Allocated to Agents

The Response Code values for DCS-detected conditions are given in [ REF DCSHLD \h ] and summarised in the table below:

| Agent Reason for Failure | RespCd |
|---|---|
| EE Down (operational) | 60 |
| EE Closed (administrative) | 61 |
| EE Timeout | 62 |
| Unexpected Message from SSE | 63 NOT USED |
| Invalid [A1] | 64 |
| Orphan or late [A1]/[E2] | 65 |
| Bad MID/TID | 66 |
| Invalid [R1] | 37 |
| Operational Problem at Agent | 38 |

**Table [ SEQ Table * ARABIC ]: Response Code values for DCS-detected conditions**

# 3 APPENDIX I –HOSTS AND SERVICES

## 3.1 Hosts and Services

### 3.1.1 Connection to Streamline

Refer to [ REF DCSHLD \h * MERGEFORMAT ] section 4 'Connectivity' and section 4.2 'Connections to the SSE' for the description of the connections to Streamline. These sections define the number of EE_IO threads, known as SSE Handlers (`EE_IO.Count`) and the number of connections per thread (`EE_IO.SocketConcurrency`). The number of Virtual Addresses (VA = an IP address and port combination) per thread (`VA_COUNT`) for DCS is 1. The target PI identifier (%T) and the index for the target VAs (%V) are substituted into strings supplied via the registry items `SOCKET_HOST` and `SOCKET_SERVICE`. After substitution the strings provide the appropriate host and service names. The service file (c:\winnt\system32\drivers\etc\services) is used to resolve the service names to port numbers. The DNS maps the host names to Streamline virtual IP addresses.

### 3.1.2 Connections for DCS Authorisation Agents to listen for OSR

Value of **LISTEN HOST** is ComputerName

Only 1 port number needed. The number follows on from those used by the NBS Authorisation Agents (see [ REF NBSLLD \h * MERGEFORMAT ]).

| | GetR1X Thread | Cluster Id (%C) | Service (Port) name in LISTENS registry | Port |
|---|---|---|---|---|
| DCS | 0 | 5 | TMSNX_%I_CL5 | N+30 |

**Table 41: Service information for listening for OSR (HNG-X)**

# 4 APPENDIX II - HEARTBEAT THREAD

This is as for the NBS Authorisation agents described in [ REF NBSLLD \h ].

In addition the thread calls Prepare_SSE_RespCd() during initialisation to prepare a linked list to map the response codes returned by the FI, see [ REF _Ref213228193 \w \h ].

Also the TMS_RX_TID_MESSAGE_NUM table (see [ REF _Ref161989025 \r \h ]) is read just before the agent becomes active.

# 5 APPENDIX III - E1RETRY THREAD

Not required for DCS.

# 6 APPENDIX IV - JOURNAL PARTITION MANAGEMENT

This is as for the NBS Authorisation agents described in [ REF NBSLLD \h ].

# 7 APPENDIX VI – State Transitions Overview

The HLD [ REF DCSHLD \h ] gives a specification of the transaction states, and the way the DCS [R1], [C0], [A1] and [E2] messages progress through these states.

The progress of these messages is controlled by a trigger value associated with the message. The value of this trigger summarises the systems view of the state of the message at any point.

This section gives an overview of the normal flow of messages through the system and how their transaction *state* and *trigger* values change.

**PreEE Thread**

An [R1] starts in the PreEE thread with *Trigger_R1* and with assumed transaction state *No_Status_Entry*.

If the message is processed successfully and an [R3] produced it is recorded in the NPS with transaction state *R3_Auth* and forwarded to the EE_IO thread.

If the message cannot be successfully processed the trigger is changed to reflect the reason, one of *Bad_R1*, *R3_EE_Closed* or *R3_No_PI*. It is recorded in the NPS with transaction state *R3_Not_Sent* and a failure [A3] returned to the counter.

Alternatively if the message is found to be stale the trigger is changed to *Trigger_Stale_R1* and it is discarded without being recorded in the NPS in any way (apart from statistics).

A [C0] starts in the PreEE thread with *Trigger_C0*.

The transaction state is read from the NPS. Only if the transaction state is *Authorised* or *A1_Timed_Out* is it worth processing the [C0] to produce an [E1] otherwise, it is either ignored, or treated as a duplicate. In particular it will be ignored if the transaction state was *Declined* or *R3_Not_Sent*.

If the message is processed successfully and an [E1] produced it is recorded in the NPS with transaction state *Reversing* and forwarded to the EE_IO thread.

If the message is too old to be sent to the SSE or if it cannot be processed the trigger is changed to *Trigger_Old_Reversal* or *Trigger_Bad_C0* it is journalised and discarded.

Alternatively if the message is found to be stale the trigger is changed to *Trigger_Stale_C0* and it is discarded without being recorded in the NPS in any way (apart from statistics).

**EE_IO Thread**

An [R1]/[R3] arrives with *Trigger_R1* and transaction state *R3_Auth*.

If it cannot be sent to the SSE the trigger is set to *Trigger_Failed_To_Send_R3* and the message is sent to the Exceptions thread to set the transaction state to *R3_Not_Sent*.

If it is sent to the SSE the trigger is set to *Trigger_At_FI*.

A [C0]/[E1] arrives witht *Trigger_C0* and transaction state *Reversing*.

If it cannot be sent to the SSE the trigger is set to *Trigger_Failed_To_Send_E1* and the message is sent to the Exceptions thread to set the transaction state to *Reversal_Failed*.

If it is sent to the SSE the trigger is set to *Trigger_At_FI*.

If a message sent to the SSE times out the trigger is set to *Trigger_A1_Timeout* or *Trigger_E1_Timeout* and it is sent to the exceptions thread where the transaction state is changed to *A1_Timed_Out* or *Reversal_Failed*.

When a reply is received back from the SSE the type of the reply [A3] or [E2] cannot be determined from the format of the reply, but only by matching the TID in the reply against the TIDs of messages with outstanding replies. As in the Horizon agent the hash table is used searching first for an “R1” entry and, if this fails, a “C0” entry, and having found such an entry the SSE Message Number and Trigger value are also checked. The consequence is that at most one SSE reply can be matched against a message sent, and hence its transaction, so replies will not be recognised as duplicates.

If a match is found the trigger is set to *Trigger_A1* or *Trigger_E2*, and if no match is found to *Trigger_Orphan_A1_E2*.

For a *Trigger_Orphan_A1_E2* we don’t have sufficient information to identify a transaction so we cannot write a status entry. So the best that can be done is to journalise it.

For *Trigger_A1* or *Trigger_E2* the message is passed to the PostEE thread.

**PostEE Thread**

An [A1] arrives with *Trigger_A1* and transaction state *R3_Auth*.

If the [A1] indicates success the trigger is set to *Trigger_A1_OK* and the new transaction state recorded as *Authorised*.

If the [A1] indicates rejection the trigger is set to *Trigger_A1_Decline* and the new transaction state recorded as *Declined*.

Otherwise if the [A1] cannot be processed the trigger is set to *Trigger_Bad_A1* and the new transaction state recorded as *Discrepency*.

In the first 2 cases but not the last an [A3] is returned to the counter.

An [E2] arrives with *Trigger_E2* and transaction state *Reversing*.

If the [E2] indicates success the trigger is set to *Trigger_E2_OK* and the new transaction state recorded as *Reversed*.

If the [E2] indicates rejection the trigger is set to *Trigger_E2_Decline* and the transaction state recorded as *Reversal_Failed*.

Otherwise if the [E2] cannot be processed the trigger is set to *Trigger_Bad_E2* and the transaction state recorded as *Reversal_Failed*.

In none of these cases is a reply returned to the counter.