POL00115409



| | |
|---|---|
| **Document Title:** | Branch Database High Level Design |
| **Document Type:** | High Level Design |
| **Release:** | HNG-X |
| **Abstract:** | This document describes the design of the Branch Database (BRDB), which is the data-store for the Branch Access Layer and also provides all of the transactional data to Legacy.<br>It also covers the BRDB-standby database, which provides a failover solution |
| **Document Status:** | APPROVED |
| **Author & Dept:** | Andy Beardmore |
| **Internal Distribution:** | *Gareth Jenkins, David Johns, Duncan Macdonald, David Harrison, Roger Barnes* |
| **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 |
|---|---|---|---|
| David Harrison | HNG-X Solution Design | | |
| Graham Allen | HNG-X Development | | |
| | | | |
| | | | |

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

POL00115409



# Document Control

## 0.1 Table of Contents

DOCUMENT CONTROL ..... 2

0.1 Table of Contents ..... 2
0.2 Document History ..... 6
0.3 Review Details ..... 6
0.4 Associated Documents (Internal & External) ..... 7
0.5 Abbreviations ..... 9
0.6 Glossary ..... 11
0.7 Changes Expected ..... 11
0.8 Accuracy ..... 12
0.9 Copyright ..... 12

1 INTRODUCTION ..... 13

1.1 Summary ..... 13
1.2 Scope ..... 13
1.3 Context within the Architecture ..... 14
1.4 Assumptions ..... 14

2 DESIGN PRINCIPLES ..... 16

3 REQUIREMENTS ..... 17

4 SUB-SYSTEM DESCRIPTION ..... 18

5 APPLICATION COMPONENTS ..... 22

5.1 Data Partitioning and Distribution ..... 22
5.2 Data Subject Areas ..... 28
5.3 System Interfaces ..... 37
5.4 Aggregation and De-normalisation ..... 60
5.5 Migration ..... 64
5.6 Training ..... 65
5.7 Branch Database Supportability ..... 66
5.8 Branch Support System Feed ..... 68
5.9 Branch Standby Database ..... 68
5.10 Post Hydra changes ..... 69

6 BUILDING BRANCH DATABASE ..... 73

6.1 Oracle Installation & Configuration ..... 73
6.2 Storage Management ..... 74
6.3 Database Components ..... 75
6.4 BRDB Database Build ..... 78
6.5 Network Configuration ..... 79
6.6 Replication to the SSC Support (History) Database ..... 80
6.7 Setting up Branch Standby Database ..... 81

POL00115409



6.8 Interfacing with non-Oracle Database Systems....81

7 BRDB HOST APPLICATION.... 83

7.1 Application Development.... 83
7.2 Host Processes.... 87

8 BRANCH STANDBY DATABASE SOLUTION....118

8.1 Standby Database Configuration.... 118
8.2 Setting up Branch Standby Database.... 119
8.3 Branch Standby Database Backups....120
8.4 Switching to BRDB-Standby....120

9 PLATFORMS.... 123

9.1 Branch Database....123
9.2 BRDB-Standby Database....124

10 NETWORKS....125

11 MANAGEABILITY.... 126

12 SYSTEM QUALITIES.... 127

12.1 Security.... 127
12.2 Availability.... 129
12.3 Usability.... 129
12.4 Performance....130
12.5 Backups.... 134
12.6 Failure & Restart/Recovery....137
12.7 Switch to BRDB-Standby.... 144
12.8 Potential for Change....144

13 IMPLEMENTATION....145

13.1 Summary....145
13.2 Delivery Units....145

14 APPLICATION DEVELOPMENT.... 147

15 TESTING AND VALIDATION....148

16 RISKS AND ASSUMPTIONS.... 149

17 REQUIREMENTS TRACEABILITY....150

18 APPENDIX A – TABLE AND INDEX DEFINITIONS.... 151

18.1 Live Schema objects shared with Training.... 151

POL00115409



19 APPENDIX B – VIEW DEFINITIONS....152

20 APPENDIX C – USER, ROLE SEQUENCE DEFINITIONS AND DATABASE LINKS....153

20.1 Linux User Definitions....153

21 APPENDIX D – METADATA AND STATIC REFERENCE DATA DEFINITION 164

21.1 Partition Management metadata....164
21.2 Housekeeping & Maintenance Metadata....169
21.3 Fad-Hash Mappings....183
21.4 Host Processing Metadata....183
21.5 Branch Specific Metadata....185

22 APPENDIX E – SUGGESTED ORACLE INITIALISATION PARAMETERS....194

22.1 Branch Database Parameters....194
22.2 Branch Standby Database Parameters....196
22.3 Role Reversal Parameter Changes....198



Figure – 1 Context within the Architecture....14
Figure – 2 Logical Data Model and Data Flows....20
Figure – 3 Concept of a Rolling Window....26
Figure – 4 Rolling Window implementation for a composite partitioned table....27
Figure – 5 Rolling Window implementation for a composite partitioned table....34
Figure – 6 Bulk Copy Schedule demonstrating Node Unavailability....43
Figure – 7 BRDB Transactions to TPS....44
Figure – 8 Hydra-specific feeds from TPS....47
Figure – 9 BRDB Transactions to APS....49
Figure – 10 BRDB Transactions to DRS....50
Figure – 11 BRDB interfaces with RDMC/RDDS....52
Figure – 12 Oracle components on each BRDB-Host node....73
Figure – 13 ASMB Instance Disk Group Layout....73
Figure – 14 Oracle components used on the BRDB – ACDB/BCDB interface....80
Figure – 15 Branch Database Execution Environment....112
Figure – 16 BRDB-Standby Platform Execution Environment....113
Figure – 17 BRDB Backup Approach....124
Figure – 18 Metadata logical data model....173

Table 1 – List of BRDB to/from Legacy Interfaces....39
Table 2 – Guidelines for Installing & Configuring Oracle....72
Table 3 – ASM Disk Group Usage Notes....74

POL00115409



Table 4 – Guidelines for Installing & Configuring Oracle....92
Table 5 – Branch Database Availability Targets....118
Table 6 – Branch Database Throughput Target....119
Table 7 – Branch Database Overall Volumes....119
Table 8 – Settlement Transaction Throughput & Volumes....119
Table 9 –Reporting Throughput & Volumes....120
Table 10 – Session Management Throughput & Volumes....120
Table 11 – Recovery Management Throughput & Volumes....120
Table 12 – Help Service Throughput & Volumes....120
Table 13 – Bulk Copy Throughputs....121
Table 14 – Audit File Throughputs....122
Table 17 – BRDB_TABLE_GROUPS Metadata....151
Table 18 – BRDB_PARTITIONED_TABLES Metadata....153
Table 19 – BRDB_TABLE_PARTITIONS Meta Data....154
Table 20 – BRDB_SUBPARTITION_RANGES Meta Data....155
Table 21 – BRDB_ARCHIVED_TABLES Metadata....160
Table 22 – BRDB_ANALYZED_OBJECTS Meta Data....161
Table 23 – BRDB_FILES_TO_HOUSEKEEP Meta Data....161
Table 24 – BRDB_PROCESSES Meta Data....162
Table 25 – BRDB_OPERATIONAL_INSTANCES Meta Data....163
Table 26 – BRDB_HOST_INTERFACE_FEEDS Meta Data....164
Table 27 – BRDB_HOST_AGGREGATIONS Meta Data....165
Table 28 – BRDB_BRANCH_ROLES Meta Data....165
Table 29 – BRDB_BRANCH_SERVICES Meta Data....166
Table 30 – BRDB_BRANCH_SERVICES Meta Data....167
Table 31 –Core BRDB Initialisation Parameters....168
Table 32 –Net Services specific Initialisation Parameters....168
Table 33 – Instance Tuning specific BRDB Initialisation Parameters....169
Table 34 – Database Tuning specific BRDB Initialisation Parameters....170
Table 35 – Standby Database specific BRDB Initialisation Parameters....170
Table 36 – Streams Replication specific BRDB Initialisation Parameters....171
Table 37 –BRDB-Standby Initialisation Parameters....172

POL00115409



## 0.2 Document History

| Version No. | Date | Summary of Changes and Reason for Issue | Associated Change - CP/PEAK/PPRR Reference |
|---|---|---|---|
| 0.1 | 22-Dec-2006 | First Draft (APP1) | |
| 0.2 | 06-APR-2007 | Incorporated Review Comments & issued second draft (APP1 & APP2) | |
| 0.3 | 09-Jun-2007 | Incorporated review comments, elaborated on host interface definitions and issued third draft (APP1 & APP2) | |
| 0.4 | 26-Sep-2007 | Incorporated review comments. Added Backup and Recovery (some sections incomplete) approach. | |
| 0.5 | 30-Oct-2007 | Updated as a result of review of development / testing for INT3. | |
| 0.6 | 03-Mar-2008 | Updated section for Data Aggregation | |
| 0.7 | 10-Mar-2008 | Updated Standby Database Section(s) | |
| 0.8&0.9 | 28-Dec-2008 | Various Changes for CP299 | CP299 |
| 0.10 | 10-Nov-2009 | Outstanding PEAKs and for review | |
| 1.0 | 17-Nov-2009 | Updated BRDB_FAD_HASH_OUTLET_MAPPINGS.csv and sent for Approval | |



## 0.3 Review Details

| | |
|---|---|
| Review Comments by : | 17th November 2009 |
| Review Comments to : | Andy Beardmore & RMGADocumentManagement GRO |

| Mandatory Review | |
|---|---|
| Role | Name |
| HNG-X Solution Design | David Harrison |
| HNG-X Development | Steve Goddard |
| Architecture | Jim Sweeting |
| SSC | Mik Peach |
| Business Continuity | Adam Parker |
| System Test | John Rogers |
| **Optional Review** | |
| Role | Name |
| HNG-X Architecture | Roger Barnes |
| HNG-X Architecture | Duncan Macdonald |
| Programme Manager | Alan D'Alvarez |
| Security and Risk Team | CSPOA.Security GRO |
| Applications Architecture | Dave Johns |
| Test Design | George Zolkiewka |

POL00115409



| | |
|---|---|
| Head of Service Management | Gaetan van Achte |
| Head of Service Change & Transition | Graham Welsh |
| Service Support | Kirsty Gallacher |
| Service Network | Ian Mills |
| Data Centre Migration | Geoff Butts |
| Integration | Peter Okely |
| SV&I Manager | Sheila Bamber |
| Tester | Hamish Munro |
| RV Manager | James Brett (POL) |
| VI & TE Manager | Mark Ascott |
| POL Design Authority | Ian Trundell (POL, via Document Control) |
| Core Services | Ed Ashford |
| System Qualities Architecture | Dave Chapman |
| Core Services | Andrew Gibson |
| Integrity Testing | Alan Child |
| Integrity Testing | Michael Welch |
| Business Architect | Gareth Jenkins |
| Development | Graham Allen |
| Architect | Jason Clark |
| Issued for Information – Please restrict this distribution list to a minimum | |
| Position/Role | Name |
| | |



( * ) = Reviewers that returned comments

## 0.4 Associated Documents (Internal & External)

| | Reference | Version | Date | Title | Source |
|---|---|---|---|---|---|
| [R1] | PGM/DCM/TEM/0001 (DO NOT REMOVE) | | | Fujitsu Services Post Office Account HNG-X Document Template | Dimensions |
| [R2] | PGM/DCM/TEM/0002 (DO NOT REMOVE) | | | | Dimensions |
| [R3] | ARC/APP/ARC/0008 | | | HNG-X Business Data Solution Architecture – Branch Database | Dimensions |
| [R4] | ARC/PER/ARC/0001 | | | HNG-X System Qualities Manual | Dimensions |
| [R5] | DES/APP/HLD/0005 | | | HNG-X RDDS Host Data High Level Design | Dimensions |
| [R6] | ARC/APP/ARC/0007 | | | HNG-X Batch Services Architecture | Dimensions |
| [R7] | DES/GEN/STD/0001 | | | Host Applications Design and Development Standards (HADDIS) | Dimensions |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| [R8] | DES/APP/HLD/0050 | | | Branch Access Layer High Level Design | Dimensions |
| [R9] | DEV/INF/LLD/0019 | | | Building Oracle 10gR2 on RHEL4 U4 AS 64-bit Low Level Design | Dimensions |
| [R10] | DES/PPS/HLD/0002 | | | Red Hat Linux 4 AS High Level Design for HNG-X | Dimensions |
| [R11] | DES/APP/HLD/0022 | | | Branch Database Sizing and Volume Spreadsheet | Dimensions |
| [R12] | DES/APP/HLD/0021 | | | Branch Database Scheduling High Level Design | Dimensions |
| [R13] | ARC/GEN/REP/0001 | | | HNG-X Glossary | Dimensions |
| [R14] | DES/APP/HLD/0023 | | | Branch Support Database High Level Design | Dimensions |
| [R15] | PGM/PAS/PRO/0003 | | | HNG-X Design and Build Methodology – Build and Unit Test process | Dimensions |
| [R16] | DES/APP/HLD/0027 | | | HNG-X TPS High Level Design | Dimensions |
| [R17] | DES/APP/IFS/0007 | | | Branch Database (BRDB) – Legacy Host Interface Specification | Dimensions |
| [R18] | DES/APP/AIS/0007 | | | HNG-X APS Host to Branch DB BRDB Interface Specification | Dimensions |
| [R19] | DES/APP/IFS/0001 | | | HNG-X RDDS Host to Branch DB BRDB Reference Data Interface Specification | Dimensions |
| [R20] | DES/APP/IFS/0002 | | | HNG-X: RDDS to Branch Database – Counters Reference Data and Memo Submission Interface Specification | Dimensions |
| [R21] | DES/APP/IFS/0010 | | | Branch Database – Branch Access Layer Interface Specification | Dimensions |
| [R22] | ARC/MIG/STG/0001 | | | HNG-X Migration Strategy | Dimensions |
| [R23] | ARC/SEC/ARC/0003 | | | HNG-X Technical Security Architecture | Dimensions |
| [R24] | ARC/SYM/ARC/0003 | | | HNG-X System and Estate Management Monitoring | Dimensions |
| [R25] | DES/SYM/HLD/0031 | | | Estate Management Database (EMDB) High Level Design | Dimensions |
| [R26] | ARC/NET/ARC/0001 | | | HNG-X Network Architecture | Dimensions |
| [R27] | ARC/SEC/ARC/0003 | | | HNG-X Security Architecture | Dimensions |
| [R28] | EP/IFS/005 | | | Horizon Counter – TPS Ongoing Migration Feed Interface Specification | PVCS |
| [R29] | DES/APP/HLD/0049 | | | HNG-X Generic Report Data Extract High Level Design | Dimensions |
| [R30] | EP/IFS/006 | | | Horizon – HNG-X In Day Migration Interface Specification | PVCS |
| [R31] | DES/APP/HLD/0070 | | | Host Application Monitoring High | Dimensions |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| | | | | Level Design | |
| [R32] | DES/APP/HLD/0005 | | | HNG-X RDDS Host High Level Design | Dimensions |
| [R33] | DES/APP/HLD/0004 | | | HNG-X RDMC Host High Level Design | Dimensions |
| [R34] | DES/SYM/IFS/0002 | | | ACDB/BCDB to Branch Database Interface Specification | Dimensions |
| [R35] | DES/SYM/HLD/0015 | | | HNG-X Backup and Recovery High Level Design | Dimensions |
| [R36] | DES/APP/LLD/0199 | | | Branch Database Schemas | Dimensions |
| [R37] | ARC/APP/ARC/0004 | | | HNG-X Architecture - Branch Access Layer | Dimensions |
| [R38] | DES/APP/HLD/0053 | | | HNG-X BAL Reporting High Level Design | Dimensions |
| [R39] | DES/APP/HLD/0121 | | | HNG-X HLD – BRDB Aggregations | Dimensions |
| [R40] | DES/APP/HLD/0120 | | | HNG-X HLD - BRDB Processing of the End of Day Migration Data | Dimensions |
| [R41] | DES/APP/HLD/0122 | | | HNG-X HLD - BRDB Data Retention | Dimensions |
| [R42] | DES/APP/HLD/0113 | | | HNG-X HLD - BRDB Processing of the In Day Migration Data | Dimensions |
| [R43] | DES/APP/SPG/0001 | | | Branch Database Support Guide | Dimensions |



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

## 0.5 Abbreviations

| Abbreviation | Definition |
|---|---|
| APEX | Oracle's Application Express (formerly known as XML DB) |
| APOP | Automated Payment Out-Pay |
| APS | Automated Payment System |
| ASM | Automatic Storage Management |
| BLOB | Binary Large Object |
| BRDB | Branch Database |
| CLOB | Character Large Object |
| CS | Customer Services |
| DMZ | De-militarised Zone |
| DR | Disaster Recovery |
| DRS | Data Reconciliation System |
| DWh | Data Warehouse |
| EM2 | Estate Management Version 2. Generic term used for HNG-X estate management changes. |
| ERwin | ERwin Data Modeller (CASE Tool) |

POL00115409



| | |
|---|---|
| ETU | Electronic Top-Up (of payment cards) |
| FAD | Financial Accounting Division [POL Outlet] |
| GB | Giga Bytes |
| GCS | Oracle Global Cache Service |
| GREV | Guaranteed Reversals |
| IAS | (Oracle) Internet Applications Server |
| IOT | Index Organised Tables |
| ITM | IBM Tivoli Monitoring Agent |
| HLD | High Level Design |
| JDBC | Java Database Connectivity |
| KB | Kilo Bytes |
| LCR | Logical Change Request (Oracle Streams data 'packet') |
| LFS | Logistics Feeder Service |
| LPAN | Logical Processor Area Network |
| LGWR | Log Writer process |
| LOB | (Oracle) Large Object (Datatype) |
| MB | Mega Bytes |
| NPS | Network Persistent Store |
| OAS | Oracle (Internet) Applications Server |
| OCFS | Oracle Cluster File System |
| OCR | Oracle Cluster Registry |
| ODBC | Microsoft's Open Database Connectivity |
| OEM | Oracle Enterprise Manager |
| OLTP | Online Transaction Processing |
| OMDB | Operation Management Database |
| Oracle 10gR2 | Oracle version 10gR2 |
| PAN | Processor Area Network |
| PGA | Program Global Area |
| POL-MIS | Post Office Limited – Management Information System |
| RAC | Real Application Cluster |
| RAD | IBM's Realtime Network Dashboard |
| RDMC | Reference Data Management Centre |
| RDDS | Reference Data Distribution Service |
| RHEL | Red Hat Enterprise Linux |
| RMAN | (Oracle) Recovery Manager |
| SAN | Storage Area Network |
| SGA | System Global Area |

POL00115409



| | |
|---|---|
| SLA | Service Level Aggrement |
| SOAP | Simple Object Access Protocol |
| SQL | Structured Query Language |
| SRDF | Symmetrix Remote Data Facility |
| SRS | System Requirement Specification |
| SYSMAN2 | Systems Management Environment (on the legacy Horizon environment) |
| T&T | Track and Trace |
| TES | Transaction Enquiry System |
| TESQA | Transaction Enquiry System Query Application |
| TPS | Transaction Processing System |
| TWS | Tivoli Workload Scheduler |
| XML | eXtensible Mark up Language |

## 0.6 Glossary

Also refer to the HNG-X Glossary [R13].

| Term | Definition |
|---|---|
| Branch Database | The centralised data store which will act as the data repository and transaction store in HNG-X. |
| Real Application Clusters | An Oracle Real Application Cluster is a group of loosely coupled computers that work together closely so that in many respects they can be viewed as though they are a single computer. Clusters are usually deployed to improve performance and/or availability over that provided by a single computer. |
| Branch Access Layer | The middle-tier that carries out the data storage, retrieval and transfer on behalf of the Counter. |
| Flashback | Flashback in the Oracle context refers to a feature by which users can access data from the table for a point in time in the recent past. Using similar technology it is also possible to rescue objects that may have been erroneously deleted without having to go to the backups. |
| Hydra | Phase covering the dual-running of Horizon and HNG-X |
| Outlets | Also known as Branches or Post Offices |

## 0.7 Changes Expected

| Changes |
|---|
| Changes to be expected for the next version of this document:<br>• HNG-X Release 2 CP's<br><br>Changes to be expected for a future version (may not be the next) of this document:<br>• Once the SRS is published, the requirements and requirement traceability sections will be updated. |

POL00115409



## 0.8 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.

## 0.9 Copyright

POL00115409



# 1 Introduction

## 1.1 Summary

This document describes the design of the Branch Database (BRDB), which will be built on a set of new database servers using Oracle's 10gR2 Real Application Cluster technology.

The Branch Database will provide a centralised data-store for all counter transactions and events for branches and the data will be retained for a period that sufficiently covers the reporting and support requirements. The Branch data store will also support Counter Training Office (CTO) needs and provides a separate data access environment for support teams.

Counter access to the Branch database will be through a middle tier logically referred to as the Branch Access Layer (BAL) - Web Services in technical terms.

BAL will log all the transactional and event messages received from the Counter to the Branch Database. When necessary BAL will record transactions requiring third party authorisation prior to forwarding the authorisation request to third parties (external interfaces) so that they can be recovered after failure.

Branch Database will provide the existing host systems like TPS, APS etc with data feeds to replace the feeds that these systems currently get from Riposte Message Store via the agents. Near real-time and batch feeds will also be in place to transfer data from existing host systems to the Branch database for use by the counters.

All business centric transactional, event and access control information sent up by the counters will be audited in its original compressed XML format. The counter must obfuscate security sensitive information where necessary. The auditable data will be accessed by the audit server over NAS.

This document is an internal Fujitsu Services document. The level of detail in this document is intended to act as a baseline to Fujitsu Services developers and testers.

## 1.2 Scope

This document is the high level design of the database that will support BAL to store transactional and events data and facilitate report generation and recovery of failed on-line transactions. It also covers details required to support the HNG-X CTO and delivery of Reference data delivery to Counters.

This document covers the design of Branch database batch processes such as those required for interfacing with other host systems, audit file generation and general database housekeeping activities.

This document also covers the design of the BRDB-Standby database and the mechanism for keeping it in step with the live Branch database.

The Branch database host processes scheduling requirements are not covered in this document but are covered in [R12]. The physical database sizing is covered in [R11].

The document does not cover the design details of the Branch Support System (also referred to as SSC History database). This is covered in Branch Support System Design [R14].

The document does not cover details of either the BAL platform or BAL processes. These are described in the Branch Access Layer High Level Design [R8].

Also, this document does not cover the processes defined within the existing host systems domain that push or pull data to or from the Branch database respectively. These details will be present in the respective host systems High Level Design documents and in the Branch Database to host systems Interface Specification documents.

POL00115409



The details of platform build for the operating system RHEL4 are present in the Red Hat Linux 4 AS High Level Design [R10]. Details of building a platform environment for hosting the Branch Database are covered in [R9].

## 1.3 Context within the Architecture

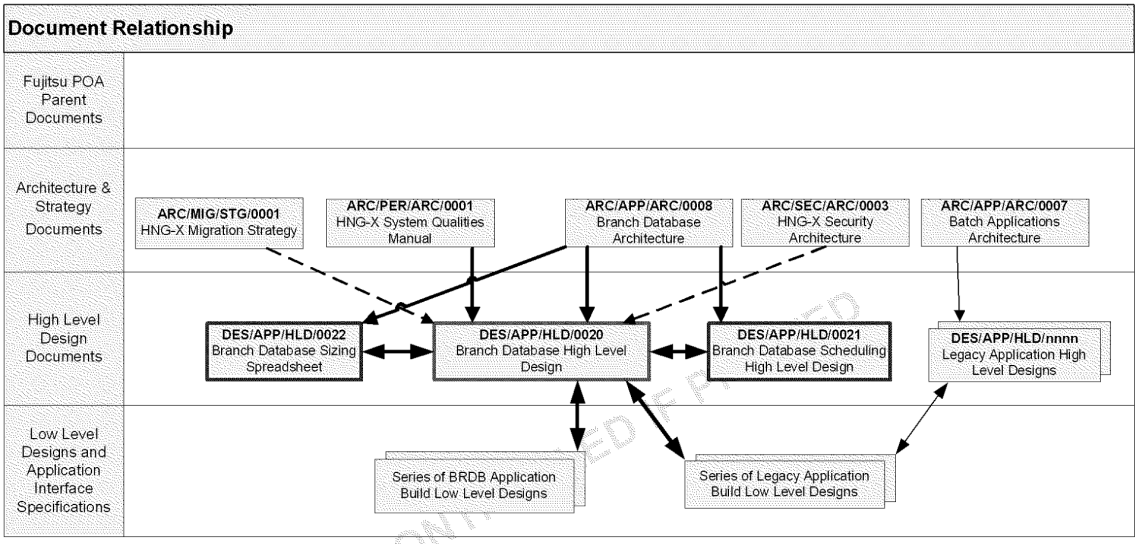



**Figure – 1 Context within the Architecture**

## 1.4 Assumptions

This design is based on the following assumptions and assertions.

1. Under all circumstances, BAL will always access the correct Oracle instance, as per the Fad-Hash to Oracle Instance mappings (defined later), regardless of the number of Branch Database nodes running.
2. BAL as far as possible will avoid the need to re-prepare and hence re-parse SQL statements. Where possible it must also use statement batching.
3. The Branch Access Layer will be responsible for deriving the value of the pseudo column “Trading-Date”. This date will not have any time part and will be derived by applying a metadata-driven delta to the Application Server clock date.

   The delta value will be 5hours i.e. the Trading Day will switch to the next calendar day at 7pm.
4. The clock of the Branch Access Layer rigs will not vary from the clocks of the Branch Database nodes by more than by 1 second. Further detail can be found in [R37].
5. All consumers of the Branch Database data will minimise the need to read non-outlet specific data from the Branch Database to reduce the cluster interconnect traffic. At the same time, outlet specific data reads will be qualified by the partitioning keys (Journal Date) and Fad-Hash to take advantage of the partitioning logic and reduce the load on the Branch Database.
6. The existing host systems interfaces will be simplified wherever possible in order to use the common host interface (discussed in 5.3.2) and thus lower the cost of development.

POL00115409



7. The flashback query feature in Oracle 10gR2 has a number of benefits around recovering the primary database on a standby platform and avoiding the re-configuration of streams. However using the 'flashback' feature has performance implications, which are as yet un-quantified. The working assumption is that this feature will be disabled.
8. It is assumed that the source and target systems for transactional data into the Branch Database are compliant with the PCI standards.
9. During the hydra phase, there will be means of identifying the up to date list of all Outlets that have migrated to HNG-X. The mechanism will be simple and involves running a SQL query against one or more tables within the Branch Database.
10. RDT requirement for Branch Database is a single node cluster (Test size). The database name and its operational requirements remain the same as those for any other test system.

POL00115409



# 2 Design Principles

- In line with other Fujitsu POA Host Systems the Branch Database design follows the standards defined by [R7].
- Fujitsu POA processes covering the full lifecycle are defined by [R15].
- The design is based on relational databases that use Oracle's RAC technology.
- The design is not object based.
- The design functional decomposition allows for a small sized development team in the order of no more than 8 developers.
- The design aims for reduced risk and cost through secure minimal change.
- The approaches highlighted in certain sections require prototyping effort to prove them. The scope of any prototyping effort needed is clearly marked in those sections.

POL00115409



# 3 Requirements

The SRS for this topic architecture captures incoming requirements. The requirements system has a skeleton copy of the document (section numbers only). The skeleton document identifies the section where the requirement is being addressed.

Section will be updated to reference the document once it is published.

POL00115409



# 4 Sub-System Description

The Branch Database (BRDB) system will provide a scalable, highly resilient (24 x7 availability) data-store for the Branch Access Layer and the Counter.

The BRDB system will run on four p-blades within the Fujitsu-Siemens' BladeFrame environment. The operating system will be a suitable patch-set of Red Hat Linux Enterprise Edition version 4 and will utilise a SAN-based storage solution.

The Branch Database will be implemented in the form of a 4-node Oracle Real Applications Cluster and database version will be a stable patch set of 10gR2. Clustering allows the database to provide the following features:

- **High Resilience:** If a database node fails, the other available nodes can take over with minimal effort on part of the consumer applications. The BRDB design allows for the system to continue running even if storage fails by providing a standby BRDB database (discussed later).
- **High Availability:** The database is a truly 24 x 7 system and has no scheduled regular downtime.
- **Workload Balancing:** A bespoke application-level load-balancing algorithm, which is discussed in the next section, ensures that the workload is distributed evenly across the available nodes of the cluster. In case of a node-failure, correct use of the load-balancing algorithm allows workload balancing across the remaining nodes.
- **High Scalability:** Oracle RAC technology allows nodes to be added or removed easily. For HNG-X although the Branch Database will not exploit this feature, there is a possibility of adding/removing nodes in future.



BRDB will provide storage to the Branch Access Layer to allow it to support the following business functionality from the Counter:

- **User and Session Management**
  Maintain Counter users, roles (Clerk, Manager etc) and stock units.
  Assist in user connection request authentication and log-on/log-off.
  Reference data version checking facility
  Record user access information for audit

- **Settlement Basket Capture**
  Record basket settlement request for audit
  Log settlement transactions for counter reporting and host systems interfaces

- **Reporting**
  Assist in generating transaction level and aggregated reports based on requests from the counter.
  Record report generation requests for audit and reporting purposes

- **External Authorisation Transaction Recovery**
  Provide persistence for potential recovery in case of failures
  Assist in transaction recovery

- **Track & Trace**
  Record Track & Trace transactions for onward routing to NPS

- **Counter Reference Data**
  Provide on-demand Reference data to the counter.

- **Cash and Stock Pouch Despatches, Advice Notices**

POL00115409



Inform counter on availability of such information.
Provide details to the counter on demand.
Record access requests for audit.

- **Delivery Notifications and Cash Declarations**
  Record notifications and declarations for audit and onward routing to existing host systems.

- **Desktop Memos**
  Notify Counter user on availability of memos and track their read status.
  Provide message details on demand and audit access requests.

The following figure shows the Branch Database data flows and a logical layout of the data structures:

POL00115409



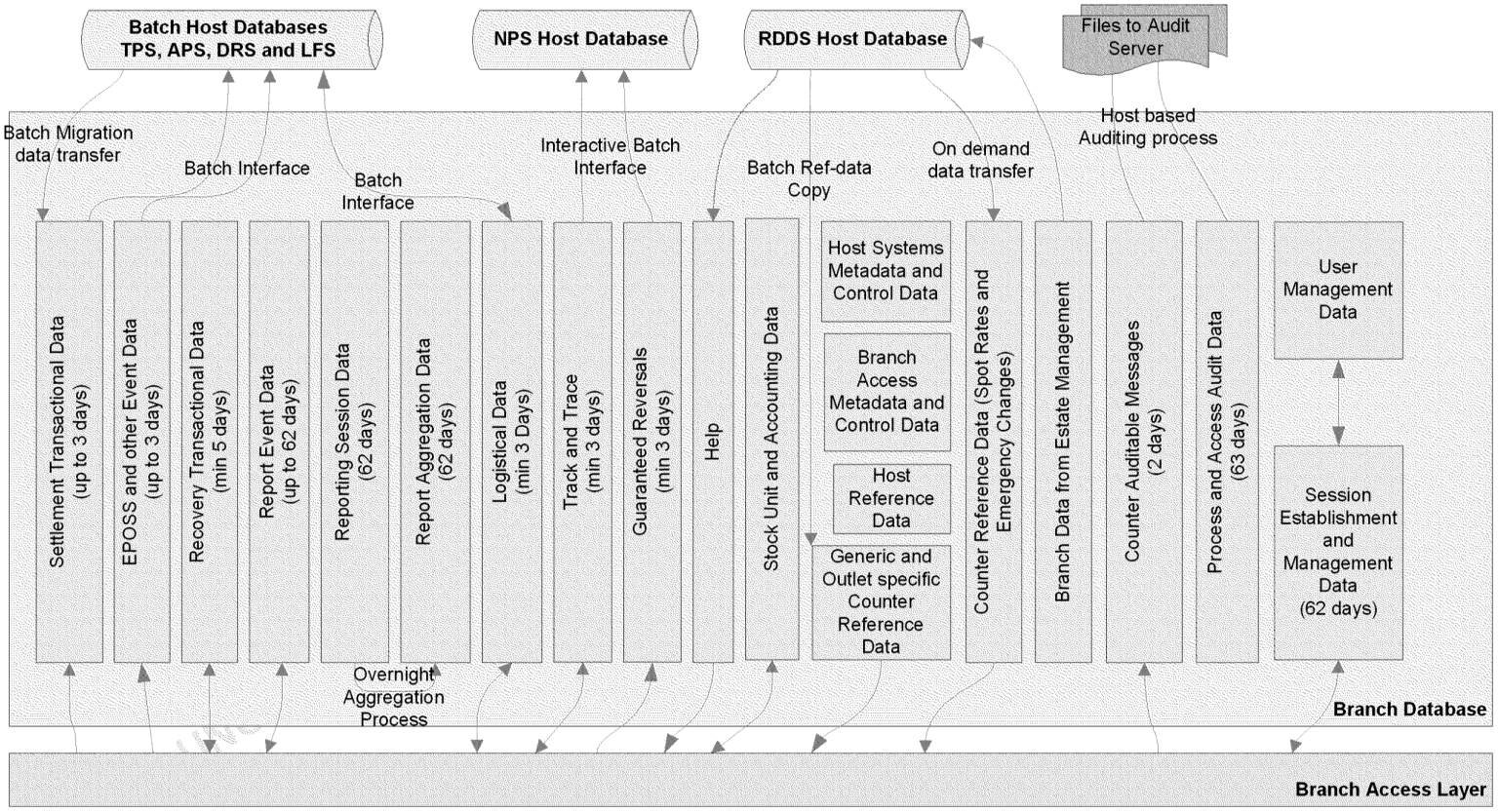



**Figure – 2 Logical Data Model and Data Flows**

POL00115409



The data in Transactional tables will be kept for up to three business days (three daily partitions). This is required to support the delivery of "must deliver" messages to existing host systems. Recovery data will be stored until the Counter marks it off as 'not needed' for transaction recovery.

Reporting transactional and aggregated data will be retained for two calendar months (62 daily partitions) to cover the longest possible trading period that an Outlet may have. All control and metadata tables will work off the same retention period of 62 days.

The BRDB system consists of a number of Linux-based host processes that are run by TWS scheduling software. These processes typically perform operations such as table and index partition management, interfacing with other Oracle-based host systems to provide data feeds to and from them, feeds to audit, file and table housekeeping and SLT report generation[1].

BRDB will provide existing batch applications such as APS, TPS, and DRS etc with transactional data, events, Cash declarations, Pouch deliveries, tracking and recovery specific data.

Batch applications will provide BRDB with Counter and Host reference data, desktop memos, transaction corrections and cash replenishment information.

Additionally during the dual-running (Hydra) phase, the TPS agent, via TPS batch applications, will provide the Branch Database with a daily feed. . This daily feed will have End-of-Day and In-Day migration

BRDB will run in archived redo-log mode to provide the facility of point in time recovery. RMAN will be used as the tool for performing backups and restore/recovery.

There will be a unidirectional real-time asynchronous feed of counter generated data from BRDB to the Branch Support (Database) System (BRSS).

There will also be another real-rime asynchronous feed of all Branch Database data to Standby Branch Database (BRDB-standby). This database will be used in the event of LIVE database failure as a result of data corruption or failures in the storage subsystem.



[1] The Branch Database is used for reference data SLA reporting and as there is limited bandwidth in the overnight batch schedule for running time-consuming jobs the Branch Support Database (BRSS) is used for other SLT reporting such as counter SLT reporting via counter audit data.

POL00115409



# 5 Application Components

## 5.1 Data Partitioning and Distribution

Database design for an Oracle Real Applications Clustered database used for high OLTP volumes, such as the Branch Database, needs to overcome the issue of Block Pinging, which could lead to serious performance degradation of the cluster.

Block Pinging refers to data or index blocks being transferred to and fro across the cluster interconnect (network that connects the RAC nodes). Block pinging causes serious performance issues due to the amount of waits generated (as processes have to wait for blocks to be transferred) and the additional network traffic that needs to flow across the cluster interconnects in order to negotiate the transfer of read-only copies or ownership of data blocks across.

Oracle suggests a number of design best practices for reducing block pinging. Almost all of the suggested best practices involve soft limiting the physical and logical data objects modified to a single instance. While this may not be practical for all objects and taken to the extreme will defeat the purpose of having a RAC database, is certainly highly recommended for objects holding high throughput OLTP transactions.

The concept of using a single RAC instance as the sole means of reading / writing transactional data has been implemented in the Branch Database at two levels, data partitioning using timestamps and/or mapping metadata and distributing partitioned data (on Fad-Hash) across tablespaces.



### 5.1.1 Data Partitioning using Mapping metadata

As outlined in [R3], the Branch Database will pre-allocate a “Hash” value ranging from 0 to 127 to each Outlet. The Hash allocation will be based on the volumes of data flowing from Outlets[2]. Once allocated, the Fad-Hash remains with the Outlet in the mapping metadata for the life of the Outlet in the Post Office branch network.

The Fad-Hash value will be used internally as the partition key within the Branch Database and also used to aid routing of particular FAD Hash values to specific Oracle Instances running within the Branch Database RAC.

The FAD Hash needs to be an integer with a range that best suits the partitioning of the RAC and the database. A range of 0-127 is considered suitable as it offers a balance between the complexities of deriving the mapping algorithm and fast data access based on full table scans across 1/128$^{th}$ of a day’s transactional / event data.

.

With a range of FAD Hash values coming from the data centre, the Branch Access Layer can very simply route the messages to one of the Oracle RAC Instances (nodes[3]) depending on the value of the FAD Hash. A simple lookup table will cross-reference the FAD Hash value to the identifier of the Oracle RAC Instance. However, there are instances where the simple look-up table is insufficient:

- ◊ Complications occur when one of the Oracle RAC Instances fails or one (or more) of the Application Servers fails to connect to an Oracle RAC Instance.

[2] A study of data volumes over a period of three months (Jul-2006 to Sep-2006) has been conducted and the Outlets have been distributed across the 128 “hash” buckets. This has since been revisited Oct-2009.

[3] Node refers to the Server where the Branch Database instance is running. Node-Id is a number ranging from 1 to 4 that has been statically assigned to each node.

POL00115409



◊ Design needs to allow for adding nodes to the Oracle RAC or removing nodes from the Oracle RAC in order to vary the processing power according to the timings of the peak loadings. This is considered unlikely in practical terms and is included for completeness.

The challenges presented by both of the above situations are similar and the solution to the problems must be the same.

The solution is that each FAD Hash must be able to be routed to *any* Oracle RAC Instance but will have a preference as to which one it should be routed to as long as that node is available. As soon as the preferred Oracle RAC Instance is unavailable, then the FAD Hash number must have associated with it a secondary preferred Oracle RAC Instance to which it is routed. Taking this to the logical extreme, each FAD Hash number must have priority-based routing to all Oracle RAC Instances. When the preferred node is not available, then the next-in-line preferred option is used.

In a configuration that allows a maximum of 8 Oracle RAC Instances, each FAD Hash value will have a mapping to each node, each of them with a different priority.

When the Node that a FAD Hash is mapped-to is unavailable for any reason, then the FAD Hash will map to the Node that is available at the next highest priority. The trick is to ensure that, regardless of which Nodes are available or unavailable, the balancing of the FAD Hash values across remaining Nodes is relatively balanced. Reducing or increasing the number of Nodes thereby retains a balanced and performance system.

The embedded spreadsheet[4] contains a matrix of FAD Hash values, Priorities (1-8) and the FAD Hash/Priority Node mapping. This provides a fairly good balance of FAD Hash/Node distribution regardless of how many RAC Instances are available.



"Fad-Hash Spreadsheet.xls"

Worksheet “PriorityMatrix” gives an easily visible FAD Hash to Oracle Instance mapping in priority order:

Worksheet “PriorityList” shows the same data in a table format.

Extracting the data from worksheet “PriorityList” into an Oracle table provides the information with which the Application Servers can determine the Instance (Node) to which a message from any FAD Hash should be routed.

If this data were loaded into a database table named BRDB_FAD_HASH_INSTANCE_MAPPING with following columns:

| Column | Type | Comment |
|---|---|---|
| FAD_Hash | Number (3) | The FAD Hash value that will accompany each communication between the Counter and the Data Centre |
| Instance_Id | Number (2) | The Instance to which the FAD Hash value maps at this priority level |
| Priority | Number (2) | The priority value of this Hash->Instance mapping. Priority 1 is highest and 8 is lowest. |
| | | |

[4] Note that this spreadsheet contains 8 levels of priority. Since the priority number maps on to a node / database instance number, This spreadsheet can only be implemented on an 8-node RAC. This version of the spreadsheet should only be used to understand the concepts. Currently 4-node RAC.

Refer to Section 21.3.3 for the correct mapping spreadsheet to be used for HNG-X.

POL00115409



As long as all Instances are available, then the following SQL returns the FAD Hash to Instance mapping:

```
SELECT fh.Instance_Id, fh.FAD_Hash
FROM   brdb_fad_hash_instance_mapping fh,
       (SELECT      FAD_Hash, MIN (Priority) Priority
        FROM        brdb_fad_hash_node_mapping
        GROUP BY    FAD_Hash) fhs
WHERE  fh.FAD_Hash = fhs.FAD_Hash
AND    fh.priority = fhs.priority;
```

#### 5.1.1.1 Handling Node Unavailability

However, we need a means of indicating the Nodes that are unavailable. This list of unavailable Nodes will need to be populated whenever Instances are removed from the cluster. This list of available nodes will also need to be updated whenever an Instance has been operationally removed from the cluster.

The solution is to maintain a single table that store the list of instances and whether they are operationally available or not. Let us assume that the table is called ***brdb_operational_instances***. Also assume that the tables have a single column each called "Node". Then by restricting the selection to the contents of brdb_operational_instances, the following SQL gets us the FAD Hash to Instance mapping:

```
SELECT fh.Insatnce_Id, fh.FAD_Hash
FROM   FAD_Hash fh,
          (SELECT   FAD_Hash, MIN (Priority) Priority
              FROM   brdb_fad_hash_instance_mapping
              WHERE  Instance IN (
                           SELECT   Node
                           FROM     brdb_operational_instances)
              GROUP BY FAD_Hash) fhs
WHERE  fh.FAD_Hash = fhs.FAD_Hash
AND    fh.priority = fhs.priority;
```



There are two scenarios:

- ◊ The schedule or an operator dictates that an Oracle RAC Instance (Node) has now been added or removed from the RAC
- ◊ An Oracle RAC Instance (or Node) fails or is restored following failure
- ◊ An single (or multiple) Application Service fails to connect to an Oracle RAC Instance

The first scenario will involve operations updating brdb_operational_instances.

The second scenario will be handled by a Host-based utility that uses the Oracle Fast Application Notification (FAN) feature. FAN is a feature that filters and publishes high priority events to specific targets. For Branch Database, the FAN will run the Host utility, which will in turn update an entry in the brdb_operational_instance table.

In the third scenario, when a single Application Server fails to connect to a particular node, then the actions to be taken include:

- ➢ Retry for up to three times with a configurable pause of *nn* seconds to allow any momentary glitch type errors.
- ➢ Use an alternate JDBC connection that has failover nodes defined so that the connection is established to an available node, to re-fetch the mapping details and using the new mappings, retry connecting to the node once.

  When the Application Server needs to re-evaluate its FAD Hash mapping by re-reading from the brdb_fad_hash_node_mapping table in the following manner:

```
SELECT fh.FAD_Hash, fh.Node
```

POL00115409



```
FROM   FAD_Hash fh,
       (SELECT      FAD_Hash, MIN (Priority) Priority
        FROM        brdb_fad_hash_node_mapping
        WHERE       Node IN (SELECT     Node
                             FROM       brdb_operational_nodes
                                        WHERE      is_available = 'Y'
                             )
       GROUP BY FAD_Hash) fhs
WHERE  fh.FAD_Hash = fhs.FAD_Hash
AND    fh.priority = fhs.priority;
```

- Otherwise bail out with error.

Note that once a node that was down comes back up, its entry in ***brdb_operational_instances*** needs to be updated. This will be done by an automated process as part of the overnight batch.

As long as the FAD Hash algorithm provides a relatively balanced distribution of transaction volumes across Hash values, then the described approach provides a flexible and resilient mechanism by which the load balancing by Outlet can be achieved within the Application Service layer.



## 5.1.2 Distributing Fad-Hash across tablespaces

One of the main performance bottleneck issues with any RAC database is un-necessary 'chatter' of data across the cluster interconnects. The data partitioning based on Fad-Hash approach mentioned earlier will help alleviate the problem for incoming data to the Branch Database. However the problem will still persist when fetching data out of the database unless preventive action is taken.

If a service in the Branch Access Layer connects to the Branch Database using one node and attempts to fetch a number of blocks of data for an Outlet, Oracle will implement its own load balancing across the cluster by distributing the ownership of data blocks amongst the four instances. This happens regardless of whether the Fad-hash algorithm has been used to fetch the data from the 'correct' instance.

If the blocks are manipulated using the same single instance, Oracle in the background will negotiate the ownership of the blocks across instances before updating them. This causes un-necessary overhead in manipulating the data and may result in performance degradation.

This problem can be minimised in the following manner:

- If the data for each Fad-Hash resides in its own data-file, Oracle's load balancing avoids distributing data across instances. This however is not fail-safe and relies on Oracle detecting that data belonging to certain data files is not used by more than one instance.
- .

The Fad-Hash based partitions or sub-partitions will be created in their own tablespaces. There will be 128 data tablespaces, where each tablespace will hold Fad-Hash based partitions or sub-partitions belonging to different tables. The extent sizes may differ from table to table.

For the second approach, please refer to Section on **Instance Tuning** for further details.

## 5.1.3 'Rolling Window' Data Storage

There is a requirement for a number of BRDB transactional and aggregation tables to retain data over a period of days. This may be because there is a business need for the data to remain in BRDB for that period of time or we may simply want to remain the data for a larger period that required in order to minimise the possibility of losing critical data before it has been, say, forwarded to existing host systems or copied across to the audit server.

POL00115409



In order to satisfy such requirements with minimal impact on performance, a number of transactional, audit and reporting tables will be implemented in the form of a 'Rolling Window' by making use of Oracle's range partitioning.

The following diagram explains the rolling window concept by taking an example of a table with a three-day rolling window on successive dates of 22nd and 23rd December:

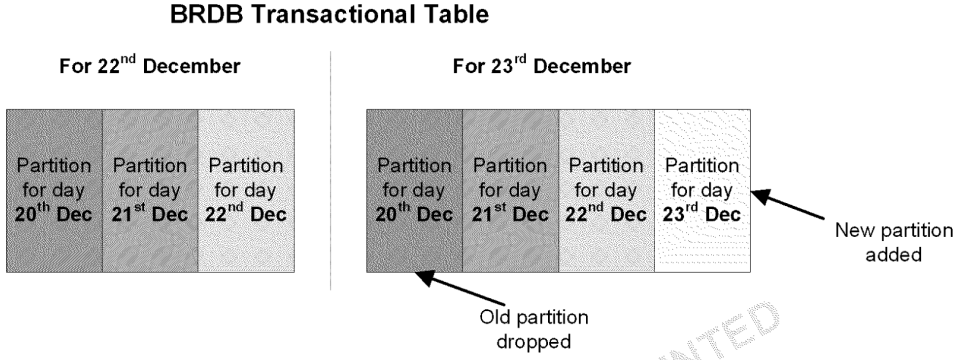



**Figure – 3 Concept of a Rolling Window**

The addition of a new partition to the table on 23rd December along with the simultaneous drop of the now aged-out partition of 20th Dec, results in the 'Rolling Window' to roll forward by a day.

There are a number of advantages of using this approach:

- Minimal performance impact of storing data for larger periods of time as approach takes advantage of Oracle's partition pruning.
- Ease of maintenance as aged-out partitions can be dropped or archived off without any potentially messy data movements.
- Metadata driven implementation of the rolling window maintenance software (see Section 7.2.1) ensures ease of changing window retention period.

For the very high volume OLTP tables, a combination of the 'Rolling Window' concept will be used along side Fad-Hashes distributed across tablespaces. This will result in a composite partitioned table with range-partitions on Business/Transactional Date and list-sub partitions on the Fad-Hash value.

The following figure explains the same pictorially:

POL00115409



**BRDB Transactional Table**

**For 22nd December**

| | Partition for day 20th Dec | Partition for day 21st Dec | Partition for day 22nd Dec |
|---|---|---|---|
| BRDB_FH_PART_000_DATA | FH - 0 | FH - 0 | FH - 0 |
| BRDB_FH_PART_001_DATA | FH - 1 | FH - 1 | FH - 1 |
| BRDB_FH_PART_002_DATA | FH - 2 | FH - 2 | FH - 2 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| BRDB_FH_PART_127_DATA | FH - 127 | FH - 127 | FH - 127 |

**Figure – 4 Rolling Window implementation for a composite partitioned table**

Note that in all of the examples in this section, partitions are shown to be created on the day that they will be needed. In reality, the BRDB Start of Day process will create partitions one day in advance. This is to ensure if partitions could not be created for any reason, Branch trading can continue while support investigates the problem.



## 5.1.4 Fad-Hash Maintenance

### 5.1.4.1 Pre HNG-X Go-Live

One of the disadvantages of having a mapping metadata driven approach for data partitioning is the need for keeping the metadata up to date. The analysis for deriving the mappings has been done quite early on in the development lifecycle and it is reasonable to expect some additions to the list of Outlets by the time HNG-X goes Live. Hence an exercise of deriving the Fad-Hash mappings for the new Outlets needs to be carried out just before going Live. This exercise should be based on analysis of the Outlet volumes in order to maintain an even balance across the Fad-Hashes and ensure that the Fad-Hash values are correctly balanced when HNG-X goes Live. The new Fad-Hash to Outlet mappings will need to be added as a delta to the Fad-Hash-Node-Mapping table. Existing Fad-Hash to Outlet mappings should not be changed.

### 5.1.4.2 Post HNG-X Go-Live

Once the Branch Database goes Live, any new Outlets joining up to the Network will need their mappings derived and present in the Fad-Hash-Node-Mapping table before the Outlet can be allowed to trade or in fact even before any outlet-specific reference data arrives in BRDB. This is because the Transactional, Event and Audit tables as well as some Ref-Data tables, amongst others, will be partitioned on Fad-Hash and without the Fad-Hash value no data can be logged in these tables.

Since the Outlet will be new to the estate, no transactional volume analysis will be possible; hence the Outlet will be assigned with a Fad-Hash value as per the following algorithm.

POL00115409



New outlets are made available by POL many weeks in advance. Fad-Hash value will assigned when the Branch information feed from EMDB is processed.

It is possible that over time the Fad-Hash to Outlet mappings will not be as evenly balanced in terms of Outlet transaction volumes as at the time of going Live. This is acceptable and an allowance of 20% has been made in the performance analysis done for the Branch Database. This allowance is on the higher side and it is not expected for the transactional volumes to be out of balance by more than a few percent.

Over a period of time, some Outlets may cease trading and eventually disappear from the Post Office Network and Reference Data. For simplicity, the Fad-Hash to Node mappings for such Outlets will never be deleted. This approach will reduce the cost of maintenance of the mapping mechanism and keep the solution simple. There will be a no impact on performance due to this.

## 5.2 Data Subject Areas

This section describes in brief each of the data subject areas depicted in Figure – 2 Logical Data Model and Data Flows.



### 5.2.1 Transactional Store

The transaction information from the basket Settlement XML message needs to be parsed and written to tables. This information is passed on to existing host systems, as a once or more-times-a-day batch feed. .

The existing host systems that read transactional data from the Branch Database are APS, DRS and TPS.

TPS and APS systems will read AP transactional data from the same source as part of the HNG-X development. The Branch Database APS tables will therefore carry the attributes that are the union of both the TPS requirements and the APS requirements. The data collected in the branch database needs to be no richer in attributes than the data required for servicing the TPS/APS applications (and their interfaces).

#### 5.2.1.1 Transactional Store Layout and Usage

The transaction store is sub-divided into various tables structured along the lines of the table structures of the interface tables in existing host systems.

Each basket settlement XML message will be broken down based on the type of message and messages will be inserted into their corresponding tables. Refer to the Branch DB – Branch Access Layer IFS [R21] for further details.

The significant tables that constitute the transaction store are:

**BRDB_RX_EPOSS_TRANSACTIONS** Stores details of all EPOSS and Settlement transaction parts. Each EPOS transaction in a customer session will result in one record being inserted into the EPOSS transactions table. The settlement transaction will also result in a record being inserted into the EPOSS-transactions table.

**BRDB_RX_APS_TRANSACTIONS** Stores details of all APS transaction parts.

Each AP transaction in a customer session will result in one record being inserted into the APS transactions table.

POL00115409



| | |
|---|---|
| **BRDB_RX_NWB_TRANSACTIONS** | Stores Network Banking & E-Top Up transaction parts.<br>Each Network Banking / E-Top Up transaction in a customer session will result in one record being inserted into the NWB transactions table. |
| **BRDB_RX_DCS_TRANSACTIONS** | Stores Debit and Credit Card transaction parts.<br>Each Debit Card transaction in a customer session will result in one record being inserted into the DCS transactions table. |
| **BRDB_RX_BUREAU_TRANSACTIONS** | Stores Bureau de Change transaction parts.<br>Each Bureau de Change transaction in a customer session will result in one record being inserted into the BUREAU transactions table. |
| **BRDB_RX_REPORT_SESSION_DATA** | Stores a subset of all transaction parts in a settlement message.<br>Every EPOS, AP, Banking + E-Top Up, Debit Card, Bureau and Settlement transaction in a customer session will result in one record each being inserted into the Report-Session-Data table. |



#### 5.2.1.2 Transactional Store Storage & Partitioning

All Transaction store tables are distributed across 128 table partitions based on the Fad-Hash value present in each transaction. The tables are partitioned on a Date value that relates to the time of the transaction.

Tables have been implemented in the form of a 'Rolling Window' as described in Section 5.1.3.

### 5.2.2 Message Journal

Every auditable request made by the counter will be logged in the message journal before the request is actioned by the BAL, The message journal performs two functions, firstly it provides auditing facility and secondly it provides a duplicate checking facility to prevent counter messages that may have been resent from being reprocessed.

The message journal will also contain messages to reflect any balancing/correcting business transactions that may have been inserted by support to assist the Post Office branch business. Refer to Section 5.7.2 for more details on balancing transactions.

In order to ensure the integrity and completeness of the audit store, business transactions involving auditable transactions will not result in a 'Success' response if they could not be logged in the message journal for any reason other than being a duplicate.

All auditable messages logged during a calendar day will be made available to the audit system in uncompressed form as a part of Branch Database batch overnight processing. However, the audit file will be compressed to reduce disk space and network utilisation when copying files to the audit system.

#### 5.2.2.1 Message Journal Layout and Usage

The message journal is implemented in the form of a single Oracle table named **BRDB_RX_MESSAGE_JOURNAL**. Uniqueness is controlled at the level of a Branch counter using a dense sequence known as the Journal-Sequence-Number.

POL00115409



#### 5.2.2.2 Message Journal Storage & Partitioning

All Message Journal table records are distributed across 128 table partitions based on the Fad-Hash value present in each auditable request. The tables are partitioned on a Date value that relates to the time of the journal message at the counter.

Message Journal will be implemented in the form of a 'Rolling Window' as described in Section 5.1.3.

### 5.2.3 Events

Events that are raised by the counter typically fall into two categories: first is where the counter raises an event for an action to be performed and the second is where the counter raises the event to report on an action that it has performed.

Events of the first type will typically be in the form of 'requests' from the counter that results in an interaction with the BAL while events of the second type will typically arrive along with unrelated requests such as Settlement..

All events will be logged in the message journal table while the Report Events will be logged in the BRDB_RX_REP_EVENT_DATA and will be identifiable by the Event-Id.

A subset of the other events referred to as EPOSS events will be stored in the table BRDB_RX_EPOSS_EVENTS and passed on to the TPS batch application for onward routing to the Post Office MIS. Another subset containing SLA measurement events will be passed on to the Data Warehouse batch application.



#### 5.2.3.1 Event Store Layout and Usage

The significant tables that constitute the event store are:

| | |
|---|---|
| **BRDB_RX_EPOSS_EVENTS** | Stores details of all events that need to be passed onto the TPS Host.<br><br>Data is passed onto TPS Host on the same business day and is retained in the Branch Database for up to 3 days (3 daily partitions worth of data are retained). |
| **BRDB_RX_REP_EVENT_DATA** | Stores details of all report events. The reason report events are stored separately from other events is so that they can be retained for a longer time period (62 days) to cover requests for any reports for the current and previous trading period. |

#### 5.2.3.2 Event Store Storage & Partitioning

All Event tables are distributed across 128 table partitions based on the Fad-Hash value present in each event message. The tables are partitioned on a Date value that relates to the date and time of the Event.

Tables have been implemented in the form of a 'Rolling Window' as described in Section 5.1.3.

### 5.2.4 Transaction Recovery

Core details of recoverable transactions will be stored in the Branch Database to allow for their recovery in the event of a failure. Examples of recoverable transactions include some externally authorised transactions such as Network Banking.

Recoverable transactions are recorded in the Branch Database during a customer settlement session as a request for the transaction is sent up from the Counter. In the event of the transaction failing due to

POL00115409



any reason e,g, comms issues, counter PC crash etc, the transaction recovery entry is used to initiate the recovery.

If the recoverable transaction succeeds, then as a part of the basket settlement processing, any recoverable transactions for the customer session are marked as processed.

#### 5.2.4.1 Transaction Recovery Store Layout and Usage

The transaction recovery store is in the form of a single table named **BRDB_RX_RECOVERY_TRANSACTIONS**.

#### 5.2.4.2 Transaction Recovery Store Storage & Partitioning

All Transaction Recovery tables are distributed across 128 table partitions based on the Fad-Hash value present in each recoverable transaction message. The tables are partitioned on Receipt Date value that relates to the time of the transaction.

Tables have been implemented in the form of a 'Rolling Window' as described in Section 5.1.3.

### 5.2.5 Reporting & Cut-offs

Data required for reporting is stored at two levels; firstly at a transaction level where core information such as Product-Id, Amount and Quantity is stored for each settlement transaction and secondly at an aggregated level that summarises the Outlet's transactions for each day at product level.

Transaction level details are stored synchronously when processing settlements. Along with the core transaction details, certain types of transactions such as AP or Banking will result in additional information being stored such as AP transaction references, Banking customer Sort Code etc.

Aggregated information will be summarised by a batch overnight host process.

Cut-off markers and details record all report cut-offs registered at the Counter. Cut-offs is recorded synchronously with cut-off events coming up from the counter.

#### 5.2.5.1 Reporting & Cut-off Store Layout and Usage

Significant tables that constitute reporting and cut-off data store are listed below. Further details can be found in [R38].

| | |
|---|---|
| **BRDB_RX_REP_SESSION_DATA** | Stores a subset of all transaction parts in a settlement message.<br><br>Every EPOS, AP, Banking + E-topup, Debit Card, Bureau and Settlement transaction in a customer session will result in one record each being inserted into the Report-Session-Data table. |
| **BRDB_CUTOFF_MARKERS** | Stores a marker for each Stock Unit Cut-off report. The marker is in the form of the highest journal-sequence-number for a cut-off report at a Branch Counter.<br><br>Table is updated by a batch-overnight aggregation process. |
| **BRDB_CUTOFF_DETAILS** | Stores cut-off details for reports at the level of a Stock Unit within a Branch.<br><br>Table is updated by a batch-overnight aggregation process. |
| **BRDB_CUTOFF_TOTALS** | Stores cut-off totals for reports at the Branch level. |

POL00115409



| | |
|---|---|
| **BRDB_SU_OPENING_BALANCE** | Stores the brought forward Cash and Stock position for each Product in every Stock Unit for a Branch.<br><br>Table is updated daily to populate it with the new stock position. |

#### 5.2.5.2 Reporting and Cut-off Storage & Partitioning

The report session data table is distributed across 128 table partitions based on the Fad-Hash value present in each settlement transaction. The tables are partitioned on the Receipt-Date value that relates to the time of the transaction. This table is implemented in the form of a 'Rolling Window' as described in Section 5.1.3.

All aggregation and stock tables will be list partitioned with 128 partitions based on the Fad-Hash value in each record. The table partitions will be static.

### 5.2.6 Logistical Store

The logistical support data stored in the Branch Database can be classified into upstream and downstream data. The upstream data includes cash declarations, pouch delivery and pouch collection details while the downstream data includes planned orders and replenishment delivery details.

The upstream data is passed by the Counter as requests that are logged in the Branch Database from where they are transferred to heritage system LFS as a part of the batch overnight schedule.

Downstream data is transferred from the LFS into Branch Database via a batch interface. As soon as the data reaches Branch Database, it will be made available to the Counter however the counter will need to initiate a request for accessing the data.



#### 5.2.6.1 Logistical Store Layout and Usage

| | |
|---|---|
| **BRDB_BRANCH_DECL**<br>**BRDB_BRANCH_DECL_ITEM**<br>**BRDB_BRANCH_DECL_MARKER** | Stores the headers and details for all Cash and Stock declarations.<br><br>Table is populated by the BAL. Data is also read by a Branch Database summarisation process for aggregating and onward routing of data to legacy host system LFS. |
| **BRDB_RX_POUCH_COLL_HEADER** | Stores the headers and details for all Pouch Collections. |
| **BRDB_RX_POUCH_COLL_DETAILS** | Table is populated by the BAL and records are then updated as they get transferred to the LFS system via a batch feed. |
| **BRDB_RX_POUCH_DEL_HEADER** | Stores the headers and details for all Pouch Deliveries. |
| **BRDB_RX_POUCH_DEL_DETAILS** | Table is populated by the BAL and records are then updated as they get transferred to the LFS system via a batch feed. |
| **LFS_PLO_HEADER** | Stores the headers and details for all Planned Orders. |
| **LFS_PLO_DETAILS** | Table is populated by a batch copy process using records read from LFS. The BAL on behalf of the Counter reads records as requested for by the Counter. |
| **LFS_RDC_HEADER**<br>**LFS_RDC_DETAILS** | Stores the headers and details for all Replenishment Deliveries.<br><br>Table is populated by a batch copy process using records read from LFS. The BAL on behalf of the Counter reads records as requested for by the Counter. |

POL00115409



#### 5.2.6.2 Logistical Store Storage & Partitioning

All upstream and downstream logistical data tables will be list partitioned with 128 partitions based on the Fad-Hash value in each record. The table partitions will be static.

## 5.2.7 Branch Information

The Branch information is available in the Branch Database table BRDB_BRANCH_INFO and BRDB_BRANCH_NODE_INFO. Primary source for this data is Estate management through the EMDB feed. Below are the details for implementing this feed. .

The functionality will be implemented as a PL/SQL package PKG_BRDB_EMDB_INTERFACE in the style of all the other non-source-updating interfaces.

The interface will be invoked by TWS once a day as part of the normal schedule.

TWS will call the BRDBX003.sh programs in the same way as other feeds.

The package will interact with the process control framework as other feeds.

The feed meta data will allow the interface to be re-runable more than once a day.

The schema needs updating to put a primary key onto the following tables

EMDB_POST_OFFICE (branch_code)

EMDB_MANAGED_NODE (branch_code, node_id)



We will process the EMDB_POST_OFFICE records first.

All rows in the EMDB_POST_OFFICE will be processed each day.

If a matching row already exists in the BRDB_BRANCH_INFO then

update the columns ip_subnet, number_of_counters, suspend_distribution, mobility_type, service_type,

cto_flag and extended_information with a simple overwrite.

If suspended_distribution_flag then set branch_status as ‘Closed’

else

if it doesn’t exist then

obtain BRDB_CURR_BRANCH_ACCOUNTING_CODE from BRDB_SYSTEM_PARAMETERS.

fad-hash = MOD(BRANCH_ACCOUNTING_CODE,128)

create new record into the BRDB_BRANCH_INFO table.

insert a new row into the BRDB_FAD_HASH_OUTLET_MAPPING table

Update BRDB_CURR_BRANCH_ACCOUNTING_CODE

endif;

On error, log error message in normal way,stop processing and exit with error code set.

We will process the EMDB_MANAGED_NODE records second.

All rows in the EMDB_MANAGED_NODE will be processed each day.

Look up the fad_hash for each branch_code from the BRDB_BRANCH_INFO table.

If a matching row already exists in the BRDB_BRANCH_NODE_INFO then

update the columns ip_address_1, extended_information, suspend_distribution_flag with a simple overwrite.

POL00115409



*Set ip_address to null for nodes where suspend_distribution flag is set*

else

if it doesn't exist then

if the parent branch CTO_flag is TRUE then

get (and later increment) the next unused branch accounting code FOR EACH COUNTER in the range 910000 to 919999

else

branch_accounting_code = branch_code

endif;

*Set ip_address to null for nodes where suspend_distribution flag is set*

create new record in the BRDB_BRANCH_NODE_INFO table.

endif;

*Ensure that all branch info records also have a DEF stock unit in BRDB_BRANCK_STOCK_UNITS*

*Ensure that branches in fad hash outlet mapping are also in* BRDB_TXN_CORR TOOL_CTL



On error, log error message in normal way,stop processing and exit with error code set.

commit at the end.

## 5.2.8 Stock Unit and Accounting

A number of Stock Unit and Accounting related objects will reside in the Branch Database. The transaction summary objects will be populated by the Branch Database overnight aggregation process.

All balances stored in the database will be derived by the Counter.

### 5.2.8.1 Logistical Store Layout and Usage

**BRDB_DAILY_SUMMARY**

Table stores a non-cumulative summarised view of settlement transactions for each Trading Day. The aggregation level is at Trading-Date, Journal Date, Fad-Hash, Branch Accounting Code, TP, BP, Stock Unit, Product Id and Transaction Mode.

Data is populated by Host aggregation process and is used as an intermediate step towards constructing the Daily-Cumulative summary.

POL00115409



| | |
|---|---|
| **BRDB_DAILY_CUMULATIVE_SUMMARY** | Table stores a cumulative summarised view of settlement transactions for each Trading Day. The aggregation level is at Trading-Date, Journal Date, Fad-Hash, Branch Accounting Code, TP, BP, Stock Unit, Product Id and Transaction Mode.<br><br>Data is populated by Host aggregation process and is used for all accounting related counter reports and as a part of the SU and Branch rollover processing. |
| **BRDB_SU_OPENING_BALANCE** | Tables store the opening and closing balances at Fad-Hash, Branch Accounting Code, Stock Unit, Trading Period, Balance Period and Product Id.<br><br>Table is populated by the BAL using figures sent up by the Counter as a part of the SU rollover process. |
| **BRDB_BRANCH_OPENING_BALANCE** | Tables store the opening and closing balances at Fad-Hash, Branch Accounting Code, Trading Period and Product Id.<br><br>Table is populated by the BAL using figures sent up by the Counter as a part of the Branch rollover process. |
| **BRDB_BRANCH_STOCK_UNITS** | Table stores the Stock Unit details.<br><br>Data is populated by the BAL based on information passed up by the Counter. |



#### 5.2.8.2 Logistical Store Storage & Partitioning

The Daily-Summary and Daily-Cumulative-Summary are composite partitioned; range-partitioned on Trading-Date & list-partitioned on Fad-Hash. The Opening and Closing Balance tables are list-partitioned on Fad-Hash. The data retention is two Trading-Periods for all of these tables.

Stock Units are retained until they have been deleted. All actions performed on Stock Units are additionally journalised into the Stock Unit History table.

### 5.2.9 Reference Data

The Branch database requires both dynamic and static reference data to support its operations. Dynamic reference data is sourced from RDDS database.

Static data is populated as part of the schema build process.

Dynamic reference data is required by the Counter, Branch Access Layer and the Branch database host processes.

As directed by the Branch DB Architecture [R3], the source system RDDS will initiate the transfer of reference data by making a call to the to the Branch database when a new version of the data is available by invoking the BRDB harvester.

The harvester or data copy mechanism will run on the Branch database node 1 and will run a number of pre-defined SQL statements that will copy the reference data updates from RDDS into the Branch Database.

Dynamic reference data can be further sub-divided into Counter reference data and the reference data used by the Branch Access Layer / Branch Database Host processes.

POL00115409



#### 5.2.9.1 Counter Reference Data

Dynamic counter reference data covers emergency reference data changes (both Counter specific and general) and changes to exchange rates.

Data will be loaded in the Branch Database incrementally from RDDS and the data-load will be triggered by the RDDS schedule.

##### 5.2.9.1.1 Counter Reference Data Layout and Usage

All of the following tables are populated by the RDDS – BRDB batch interface.

| | |
|---|---|
| **RDDS_PACKAGE_TYPE** | Table maintains a list of all package types. Examples of package types are “MAIN”, “SPOTRATES”, “MARGINS” and “OTHER”. |
| **RDDS_DELIVERY_TYPE** | Table maintains a list of delivery types. Examples include “COMMON”, “PINPAD” etc. |
| **RDDS_PACKAGE** | Table defines each reference data package that is delivered down to the counters. |
| **RDDS_DELIVERY** | Table defines each of the reference data deliverables down to the counter. |
| **RDDS_PACKAGE_CONTENT** | Table contains the actual reference data file that will be delivered down to the counters. |



#### 5.2.9.2 Counter Reference Data Storage & Partitioning

All Counter reference data tables are non-partitioned.

#### 5.2.9.3 Host Reference Data

The common host systems interface (discussed later in Section 5.3) will access RDDS for reference data via a set of views and will refresh the data in equivalent tables owned by the Branch-DB schema owner.

The Reporting service within the Branch Access Layer needs to be able to access reference data going back 60 days. RDDS generally manages the retention of it’s data in BRDB and so the interface can be kept simple by deleting records from the branch database tables prior to populating them with new reference data on each day.

Note that a single commit should be used following deletion and population with new data. In this way, should the RDDS be unavailable or the copy process fail in any way, the previous days data will still be available to allow operation of the BRDB host processes and BAL.

There can be multiple versions of reference data items in each of the RDDS views. The most recent version of reference data for a key value e.g. Product-Id can be obtained by using the record with highest Version-Number and whose Valid-From and Valid-To Dates cover the date required.

##### 5.2.9.3.1 Host Reference Data Layout and Usage

All of the following tables are populated by the RDDS – BRDB batch interface.

| | |
|---|---|
| **RDDS_BRANCHES** | Provides details of all Post Office Branches including training branches. It also provides a history of Branch reference data with each instance having an effective start date and end date (optional). |

POL00115409



| | |
|---|---|
| **RDDS_PRODUCTS** | Table provides details of all Post Office products. Table also contains a history of product reference data with each instance having an effective start date and end date (optional). |
| **RDDS_ACCOUNTING_NODES** | The table provides details of all the accounting nodes that make up the reporting hierarchies. There is one main reporting hierarchy (top node 3017) defined.<br><br>The table contains a history of accounting node reference data with each instance having an effective start date and end date (optional). |
| **RDDS_BRANCH_OPENING_PERIODS** | The table provides current details (including opening timings) only of all Post Office Branches including training branches. |

The interface between RDDS and BRDB is described in the Branch Database – RDDS Interface Specification [R19].

#### 5.2.9.4 Host Reference Data Storage & Partitioning

All Host reference data tables are non-partitioned.



## 5.3 System Interfaces

### 5.3.1 Overview

At HNG-X, transactional data is captured in the Branch database in real-time when providing services such as sale of goods at Post Office branches. The existing Horizon solution consists of a number of host systems that require the captured data in the same or modified form at various times during the day.

The existing host systems also interface with the PO back office systems and the Reference Data service and these systems need to transfer data to the Branch database for onward routing via the Branch Access Layer to the Post Office branches as and when necessary.

The following table summarises the Branch Database interface with various host systems.

| No | Description | Source System | Target System | Frequency | Volumes |
|---|---|---|---|---|---|
| 1. | Transfer AP, EPOS, Banking, E-topup, Debit/Credit Card and Bureau transactions to TPS. Also transfer EPOSS Events to TPS. | BRDB | TPS | Once per day | Very High |
| 2. | Transfer aggregated transaction totals to TPS | BRDB | TPS | Once per day | Medium |
| 3. | Transfer C12 for Banking, E-topup and Debit/Credit card transactions to DRS | BRDB | DRS | Once per day | High |
| 4. | Transfer AP transaction details to APS | BRDB | APS | Once per day | Medium |
| 5. | Transfer Track & Trace transactions from BRDB to NPS-Host | BRDB | NPS | Interactive feed once every 1 min | Medium |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| 6. | Transfer Guaranteed Reversals from BRDB to NPS-Host | BRDB | NPS | Interactive feed once every 1 min | Low |
| 7. | Transfer Planned Order details from LFS to BRDB | LFS | BRDB | On Demand. Normally once per day but could be more in case of late deliveries. | Low |
| 8. | Transfer Replenishment Delivery details from LFS to BRDB | LFS | BRDB | On Demand. Up to 16 per day. | Low |
| 9. | Transfer Cash Declaration details from BRDB to LFS | BRDB | LFS | On Demand. 2 per day | Low |
| 10. | Transfer Pouch Delivery details from BRDB to LFS | BRDB | LFS | On Demand. 2 or more per day | Low |
| 11. | Transfer Pouch Collection details from BRDB to LFS | BRDB | LFS | On Demand. 2 or more per day | Low |
| 12. | Transfer emergency common or Outlet Specific Reference Data, Spot Rates and Desktop Memos from RDDS to BRDB | RDDS | BRDB | On Demand. Typically Once per day but may be more | Low |
| 13. | Transfer Branch, Product, Accounting Node and helpdesk-specific from RDDS to BRDB | RDDS | BRDB | Once per day | Low |
| 14. | Transfer up to date HNG-X Outlet information from BRDB to RDMC | BRDB | RDMC | Once a day | Low |
| 15. | Transfer up to date HNG-X Outlet information from BRDB to RDDS | BRDB | RDDS | Once a day | Low |
| 16. | TPS Transaction Corrections to BRDB | TPS | BRDB | Once per day | Very Low |
| 17. | Transfer ongoing Outlet information from TPS to BRDB (hydra only) | TPS | BRDB | Once per day | Very High |
| 18. | Transfer ongoing reconciliation information from TPS to BRDB (hydra only) | TPS | BRDB | Once per day | Low |
| 19. | Transfer 'in-day' Outlet information from TPS to BRDB (hydra only) | TPS | BRDB | Once per day | Medium |
| 20. | Branch & Counter status feed from SYSMAN2 to BRDB | SYSMAN2 (OMDB) | BRDB | Once a day | Very Low |
| 21. | HNG-X Counter Network Information | ACDB | BRDB | Once a day | Very Low |
| 22. | Branch and Counter Migration Status to ACDB (Hydra only) | BRDB | ACDB | Once a day | Very Low |
| 23. | *Branch detail feed to Helpdesk System* | *BRDB* | *Dispatch1* | *Once a day* | *Low* |



**Table 1 – List of BRDB to/from Legacy Interfaces**

Considering the large number of interfaces to be developed, there is a strong need for identifying a common approach for defining these interfaces and a common mechanism for implementing them.

POL00115409



The main driver for this is cost as aggressive costing of the HNG-X programme presents the challenge of cost reduction by minimising design and development effort. A common approach also provides the means of incorporating the odd interface that may need to be introduced as a late design change but in the HNG-X timescales, at relatively low cost.

A true generic host interface is not proposed because although such an interface is architecturally clean and presents a truly flexible means for interfacing batch systems, the high design and development effort involved in constructing such a mechanism would reduce the cost benefit of developing the common interface.

## 5.3.2 Identifying Common Functionality

### 5.3.2.1 Data Copy Approach

The common interface relies on the fact that all of the interfaces listed earlier fall into two types:

- Data primed for copying to existing host systems is simply copied across the interface using the SQL statement `INSERT INTO... SELECT FROM...`
- Data is fetched from source table/s with either row-locking or ROWIDs to 'remember' them (`SELECT FROM... WHERE...`), copied across the interface (`INSERT INTO... SELECT FROM... WHERE...`) and the remembered records are updated in the source table to mark as 'copied' (`UPDATE... WHERE...`).



Interfaces involving large amounts of data and are executed once per business day fall into the former category while those that copy across smaller amounts of data or are executed more than once per day fall into the later category. One exception to this is that some of the interfaces transferring reference data use the first type of approach with an additional preceding step of clearing down the target table/s.

Interfaces that fall into the second category need to run multiple SQL statements to implement them. There is also the need to 'remember' the rows affected by a SQL statement so that they can be marked off as transferred in a final SQL statement. Although the most efficient way of doing this is to 'remember' the Oracle ROWIDs of source records that are transferred across the interface, at Oracle 10gR2, there is a vulnerability of this approach because of the potential use of 'flashback'. There are no plans to use 'flashback' in the Branch Database.

Oracle guarantees that once assigned, the ROWID of any record remains constant (provided 'flashback' feature was not used) unless the row length increases by more than the allowance made for increases in the block, which then causes Oracle to migrate the entire row to a different block and assign a new ROWID. The other reason of ROWIDs changing is table restructuring through export/import or other such means. Since both of these activities are not expected to happen while the batch copy process is executing, ROWIDs can safely be used to mark records as transferred.

Another potential approach is to programmatically remember the records fetched, and use them when updating the table to set the flag. Although this approach does not have the vulnerabilities present for using ROWIDs, it is also not as performant as using ROWIDs.

To facilitate implementation of interfaces that require the source tables to be updated based on the ROWIDs of records transferred, a temporary[5] working table with the following structure will be delivered for each Host interface:

Table Name: **WKG_***Name of the Host interface as defined in BRDB_HOST_INTERFACE_FEEDS*

| Column Name | Datatype & Size | Mandatory | Comments |
|---|---|---|---|
| FAD_HASH | Number (3) | Yes | Stores the Hash-value of the Branch that the record |

[5] Table may not be defined as "temporary" in Oracle terms but its usage will be similar to that for typical temporary tables.

POL00115409



| | | | |
|---|---|---|---|
| | | | belongs to. Used as a key for table partitioning. |
| SRC_ROWID | ROWID | Yes | Stores the ROWID of the source record that is being transferred to the corresponding table in the target database. |
| SOFT_DELETED_YN | Varchar2 (1) | Yes | Values are "Y" (Yes) and "N" (No).<br>It is used to exclude rows from the update of the source table - because the data has not been loaded into the target table (due to a data error). |

The table is list-partitioned on Fad-Hash with 128 list partitions all spread across 128 tablespaces. Table access is by Fad-Hash as described in Section 5.1.

#### 5.3.2.2 Fad-Hash aware Table Access

All partitioned tables in the Branch Database must be accessed is a fad-hash aware manner i.e. Fad-Hash (sub) partitions of the table must be accessed using the correct instance. The view BRDB_FAD_HASH_CURRENT_INSTANCE provides the 'current' Fad-Hash to instance mappings for all available instances.



#### 5.3.2.3 Handling Exceptions

Exceptions encountered by interfaces that are driven through the Branch Database would be handled with a common approach. If an Oracle exception occurs while transferring data across the interface, the exception will be checked to see if it were a data-related exception. A simple way of doing so is to log a list of exception codes for data-related exceptions in a static reference data table named BRDB_ORACLE_ERROR_CODES and to compare the exception that is raised by the interface against this list.

Data related exceptions should not cause the interface to fail. The reason for the error is likely to be that a few records would have violated a constraint of the target table. These records should be copied to an exceptions table and rest of the interface should be processed. For interfaces involving marking the source records as 'processed' by setting a flag/date timestamp, the erroneous records should not be marked as 'processed'. This gives support the chance to correct the data in the input tables before the next scheduled run of the interface so that the data gets transferred.

Data exceptions should be logged in a common table named BRDB_HOST_INTERFACE_EXCEPTIONS unless otherwise stated e.g. unless there is a pre-defined exceptions table in batch host applications. Each erroneous record should be logged as a separate row in the table. Against each row, the Oracle error code and details should also be logged.

There is a limit on the number of records each interface can log into this table. This will be driven to a BRDB system parameter and the initial value should be set to 1000.

If the interface has encountered one or more data exceptions, then on completion of processing an Operational exception should be logged in order to highlight to support the presence of data exception details in the process logs and error tables.

All non data-related errors such as system errors or connection issues should result in the job exiting with a failure.

#### 5.3.2.4 Metadata driven solution

One of the approaches suggested by the high level design is using a metadata-driven approach for defining the interfaces. Such an approach provides the necessary flexibility for adding, changing or removing interface definitions. Refer to for details

POL00115409



### 5.3.3 Failures and Restartability

The logical processing unit for all interfaces involving Fad-Hash partitioned / sub-partitioned tables as either the source or target is a Fad-Hash value. The use of HADDIS-compliant process-control measures will ensure that when the data for a Fad-Hash value has been transferred successfully, process-control will record the fact and prevent the same partition from being re-processed in the event of a restart. Use of process-control also ensures that in the event of a restart after failure, the failed Fad-Hash is reprocessed.

For interfaces that do not involve Fad-Hash partitioned / sub-partitioned tables, the logical process unit is the entire interface. In the event of a restart after failure, the interface is re-processed.

Note that for the restartability to work correctly, the interface must be able to rollback any changes made or data transferred before the failure occurred.

#### 5.3.3.1 Handling Node Unavailability

One of the advantages of using an Oracle RAC database is that the database is available for use even when one or more nodes (servers hosting the Oracle instances) are down. Nodes could be unavailable for a variety of reasons from OS crashes to planned maintenance work.

Every batch interface that copies transactional data along with the copy job schedule needs to handle unavailability of one or more nodes. A node could become unavailable before the interface starts executing or while it is running. Both types of situations must be handled and the copy job schedule must complete using the available nodes to copy the data for the pending Fad-Hash values across.

Node unavailability needs to be handled differently for jobs that are controlled at Fad-Hash level from the jobs that run as interactive batch feeds e.g. NPS T&T and GREV interfaces.



**Fad-Hash Level Controlled Jobs**

Jobs that are controlled at Fad-Hash level are the ones that typically run at defined times during the day and for whom it is absolutely essential for all of the transactions to be transferred across the interface in that run of job.

The solution proposed below is simple to implement and works within the constraints of the job scheduling software TWS.

The batch interface will handle failures such as Oracle instance going down by bailing out with a specific error code, say 99. Although the scheduling software will not be able to distinguish such errors from other application failures, which typically result in error code of 1, the error code of 99 will aid supportability.

The scheduler waits for all instances of the copy job to either finish execution (successfully or otherwise) or for a small time period e.g. 15 minutes to elapse from the start of the schedule, before running a ‘check’ job which checks if all of the possible Fad-Hash values (128 of them) have been processed successfully by the interface job and if so it returns a Success. If one or more Fad-Hash partitions have not been processed at all or the copy has not completed successfully, the check job returns a Failure.

If the ‘check’ job returns a Failure, the scheduler invokes ‘recovery’ batch interface jobs on all of the nodes with exactly the same parameters as before to allow the copy mechanism to process the pending Fad-Hash values.

This approach relies on the Oracle Fast Application Notification utility to quickly detect and record the fact that the node/instance is down in the table ***brdb_operational_instances***. The Fad-Hash filtering SQLs that are run by the batch interface will return a different set of Fad-Hashes for each instance. The new set of fad-hashes will include those that normally belong to the failed instance but have been redirected to the alternate nodes.

It is prudent to run the ‘check’ job again to ensure that all Fad-Hashes have been covered. If the check job fails again, a high-priority alert will be raised to allow the support lines to intervene.

POL00115409



**Example**

The recovery schedule structures can be best explained through an example. For the purpose of this example only, it is assumed that the common batch interface described in  is implemented. If an alternate method was used, the approach outlined by this example is still valid.

The full command used to invoke the routine Bulk-copy (batch interface) process is expected to be:

```
BulkCopy.sh Interface-Name Business-Date Node-Id
```

Let us assume that the overnight transaction copy interface from Branch Database to TPS is called "BRDB_TXNS_TO_TPS". **`BulkCopy.sh is actually script BRDBX003.sh.`**

With business date as 20th Nov 2009, the scheduler will run four instances of the Bulk-copy process for this interface:

On **Node1** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 1`**"

On **Node2** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 2`**"

On **Node3** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 3`**"

On **Node4** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 4`**"

If the job on **Node3** fails during execution and an error code of 99 is returned back to the scheduler, once the conditions described earlier are satisfied, the check job is run and if it returns a failure, four further recovery jobs are fired by the scheduler. Each job will be passed exactly the same parameters as before.

On **Node1** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 1`**"

On **Node2** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 2`**"

On **Node3** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 3`**"

On **Node4** execution command is "**`BRDBX003.sh BRDB_TXNS_TO_TPS 20061120 4`**"

The recovery jobs will attempt to re-copy the data for all fad-hashes that have the respective nodes as priority – 1 node and will also cover all those fad-hashes whose priority – 1 node is the failed node but the priority – 2 node is the node the recovery job instance runs on.

The information is depicted diagrammatically below for failure of Node-3:

POL00115409



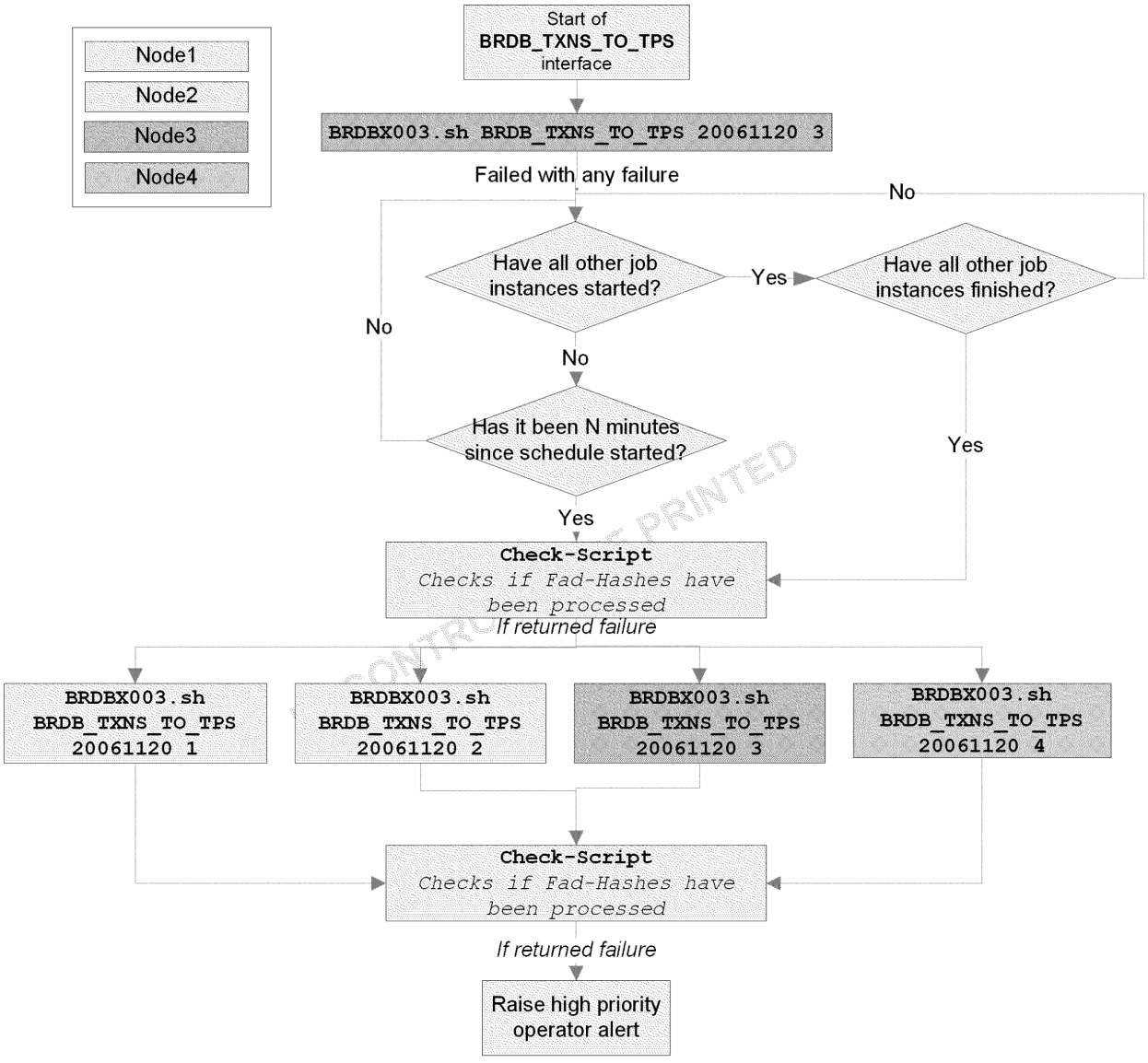



**Figure – 6 Bulk Copy Schedule demonstrating Node Unavailability**

The above-mentioned approach is applicable for nodes that become unavailable before or during execution of the Bulk copy process.

The changes required for implementing the proposed changes for handling node availability to the common bulk-copy mechanism described earlier are minimal.

**Interactive Batch Jobs**

Jobs that run as an interactive batch feed are the ones that are either run at frequent intervals by the scheduler or run in the foreground and wake up after pausing for a short period of time.

For such jobs, if some of the data is not copied across in a particular run due to a non-persistent error, there is always the next run to catch up with the data copy.

POL00115409



The solution proposed for such jobs in the event of a failure is to rerun the job instance after pausing for a short period of time to allow the transient error such as a network glitch to disappear. If the job instance fails again, raise a high priority alert to allow the support teams to quickly respond to, resolve the problem and reinstate the interactive batch feed.

## 5.3.4 Host Interfaces

### 5.3.4.1 TPS

The BRDB – TPS batch interface will perform the following functions:

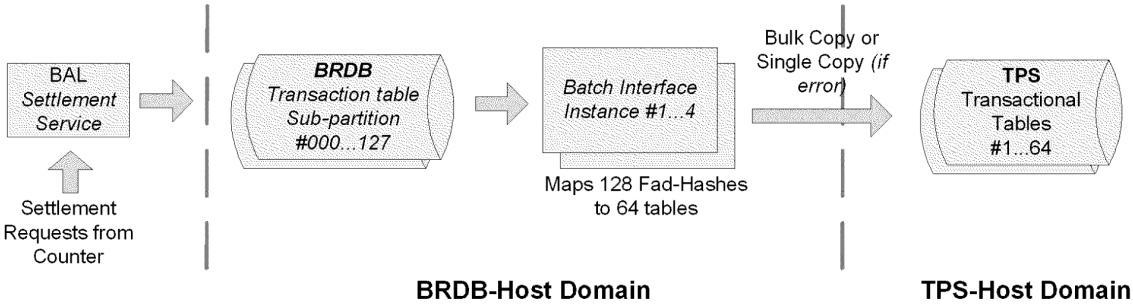
- **BRDB Transactions to TPS:** Once a day transfer of transactional data and EPOSS Events from BRDB to TPS to allow TPS to perform HR-SAP and POL-FS summarisations and onward route the transactional and event data to POL-MIS.
- **BRDB Transaction Totals to TPS:** Once a day transfer of aggregated transaction totals from BRDB to TPS to allow TPS to perform reconciliation with similar totals derived from the transactional feed delivered from the Branch Database to TPS.
- **BRDB Cut-off Summaries to TPS:** Once a day transfer of counter report cut-off summary information from BRDB to TPS to allow TPS to pass these on to the POL-FS.
- **TPS Transaction Corrections to BRDB:** Once a day transfer of Transaction corrections from TPS to BRDB. These messages will be relayed by the Branch Access Layer to the Counter, which uses them to perform accounting corrections.

In addition for the duration of the pilot, TPS will receive a new **Migration-specific feed** of all bits of information that it does not receive from the TPS harvester e.g. current stock position, non-EPOSS events, T&T sack details, cut-off data for cut-off reports etc. This information will be collated in TPS for all outlets that have not migrated to HNG-X and relayed to the Branch Database in the form of once a day type feeds.

#### 5.3.4.1.1 BRDB Transactions to TPS

This batch feed will run after the logical event "Start of BRDB-Batch[6]" has occurred.

This feed is characterised by very high data volumes however the implementation is simple as is depicted by the following diagram:




[6] This will be implemented in the form of a scheduler job which either creates a flag-file or sets a scheduler variable to TRUE. The time is expected to be 19:00 or thereabouts. Refer to the Scheduling High Level Design [R12] for details.

POL00115409



**Figure – 7 BRDB Transactions to TPS**

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances running for each of the following transaction types:

- Automated Payment Transactions
- Electronic Point of Sale Transactions
- Network Banking & E-Topup Transactions
- Debit Card Transactions
- Bureau De Change Transactions
- EPOS Events
- Summary of Cut Off Events
- Transaction Totals

For each transaction type from above, the interface steps involve:

- Fetching records for the current day's Trading Date from the appropriate Fad-Hash sub-partition in the BRDB source table where the TRANSFERRED_FROM_LEGACY_YN flag value is set as "N".
- Inserting records into the appropriately numbered target table in TPS.

Copied records need not be marked in the source BRDB tables as 'processed'.

The 128 Fad-Hash sub-partitions in BRDB will be mapped to 64 TPS partitions as per the following:

```
Fad-Hash #0 => TPS Table #1
Fad-Hash #1 => TPS Table #1
Fad-Hash #2 => TPS Table #2
Fad-Hash #3 => TPS Table #2
Fad-Hash #4 => TPS Table #3
…
Fad-Hash #127 => TPS Table #64
```



Any records that could not be copied across to TPS due to a data-related exception (as opposed to other exceptions such as system level or connection related) will be logged in the TPS Harvester Exceptions table (TMS_HARVESTER_EXCEPTIONS).

The source and target data attribute mappings / derivations are defined in the Branch Database – TPS Interface Specification [R17].

### 5.3.4.1.2 BRDB Transaction Totals to TPS

The batch feed will run after the logical event "Start of BRDB-Batch" has occurred.

This feed is characterised by medium data volumes and simple implementation.

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances of the interface with each instance running on a Branch Database node.

The interface steps involve:

- Fetching records for the current day's Trading Date from the appropriate Fad-Hash sub-partition in the BRDB source table BRDB_TX_TRANSACTION_TOTALS) where the TRANSFERRED_FROM_LEGACY_YN flag value is set as 'N'.

POL00115409



- Inserting records into the un-partitioned target table in TPS (TMS_RX_TRANSACTION_TOTALS).

Copied records need not be marked in the source BRDB tables as 'processed'.

The 128 Fad-Hash sub-partitions in BRDB will be mapped to a single TPS transaction table.

The source and target data attribute mappings / derivations are defined in the Branch Database – Legacy Host Interface Specification[R17].

#### 5.3.4.1.3 Cut Off Summaries to TPS

The batch feed will run after the logical event "Start of BRDB-Batch" has occurred.

This feed is characterised by medium data volumes and simple implementation.

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances of the interface with each instance running on a Branch Database node.

The interface steps involve:

- Fetching records for the current day's Trading Date from the appropriate Fad-Hash partition in the BRDB source table BRDB_RX_CUT_OFF_SUMMARIES.
- Inserting records into the appropriately numbered target table in TPS.



Copied records need not be marked in the source BRDB tables as 'processed'.

The 128 Fad-Hash sub-partitions in BRDB will be mapped to 64 TPS partitions as per the following:

```
Fad-Hash #0 => TPS Table #1
Fad-Hash #1 => TPS Table #1
Fad-Hash #2 => TPS Table #2
Fad-Hash #3 => TPS Table #2
Fad-Hash #4 => TPS Table #3
…
Fad-Hash #127 => TPS Table #64
```

Any records that could not be copied across to TPS due to a data-related exception (as opposed to other exceptions such as system level or connection related) will be logged in the TPS Harvester Exceptions table (TMS_HARVESTER_EXCEPTIONS).

The source and target data attribute mappings / derivations are defined in the Branch Database – TPS Interface Specification [R17].

#### 5.3.4.1.4 TPS Transaction Corrections to BRDB

This batch feed is initiated by the TPS-Host batch schedule and typically runs at the tail end of the TPS schedule TPS_TC, which run in the early hours of the day.

This feed is characterised by very low data volumes and a simple implementation. The interface is implemented in the form of a single job instance. The instance runs on a pre-defined BRDB-Host node and node failure recovery actions involve running on other nodes.

The interface steps for copying transaction corrections across from TPS to BRDB involve:

- Fetching all transaction correction records from TPS table TMS_TX_TPS_TC_DETAIL, which have not been copied across to BRDB (indicated by HNGX_ACTIONED_IND values of Null / 'N'), from TPS.
- Inserting records into the target table (TPS_TXN_CORRECTION_DETAILS) in BRDB.

POL00115409



- Updating the copied records in TPS to mark them as 'copied' by setting the HNGX_ACTIONED_IND flag value to 'Y'.

The source and target data attribute mapping / derivation are defined in the Branch Database – Legacy Host Interface Specification [R17].

### 5.3.4.1.5 Hydra-specific Migration Feeds from TPS

These batch feeds will run after the logical event "Start of BRDB-Batch" has occurred and after the completion of the execution of the migration-specific TPS harvester. Refer to [R12] for details of schedule dependencies.

These feeds are characterised by a varied range of volumes (low to very high) and the number of tables affected within the Branch Database. The following diagram presents a pictorial view of the interface:

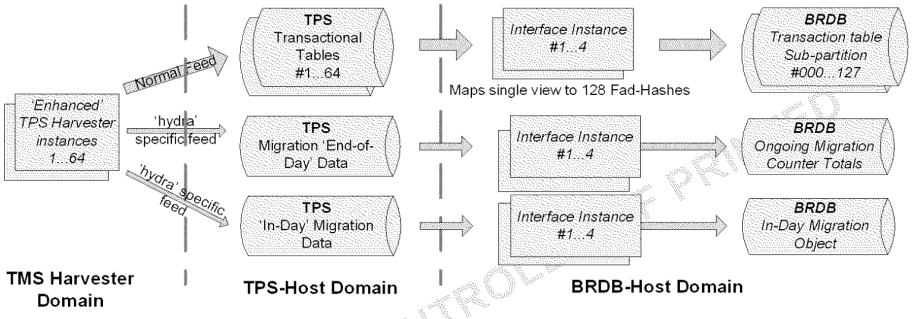



**Figure – 8 Hydra-specific feeds from TPS**

#### Transactional Data

The ongoing hydra-specific interface for transactional data is initiated by the TPS operational schedule and implemented in the form of four instances running for each of the following transaction types:

- Automated Payment Transactions
- Electronic Point of Sale Transactions
- Network Banking & E-Topup Transactions
- Debit Card Transactions
- Bureau De Change Transactions

For each transaction type from above, the interface steps involve:

- Fetching records for the current day's Trading Date from the appropriate partition in the TPS source table.
- Inserting records into the target table in BRDB while populating the Fad-Hash value for the Branch.
- Inserting a subset of the record into the BRDB table BRDB_RX_REP_SESSION_DATA to facilitate counter reports, while populating the Fad-Hash value for the Branch.
- Inserts into the BRDB transactional table should match inserts into BRDB_RX_REP_SESSION_DATA. Any data exception should cause the failed record to be not written to both the tables.
- Set the TRANSFERRED_FROM_LEGACY_YN flag in the BRDB tables to "Y" for all transferred transactions.

POL00115409



Copied records need not be marked in the source TPS tables as 'processed'.

There is no defined rule for mapping the 64 TPS partitions to the 128 Fad-Hash sub-partitions in BRDB and it is recommended that each of the four interface instances use the TPS overall view to fetch records for specific Fad-Hash values.

**Reconciliation Data**

The ongoing hydra-specific interface for reconciliation totals from the counter is initiated by the TPS operational schedule and implemented in the form of a single instance of the interface.

The steps to be executed by the interface step are:

- Fetch records for the current day's Trading Date from the migration preparation TPS source table TMS_RX_HNGX_MIGRATION_PREP.
- Insert those records into the target table in BRDB TPS_HYDRA_RECON_TOTALS while enriching it with the Fad-Hash value for the Branch.

Copied records should be marked in the source TPS table as 'processed'.

The source and target tables are both non-partitioned.



**In-Day Migration Data**

The in-day hydra-specific interface for the migration object from the counter is initiated by the TPS operational schedule and implemented in the form of a single instance of the interface.

This batch feed will bring across all of the information necessary for a just-migrated Horizon Branch to be able to operate under HNG-X. This includes the Branch's cash & stock positions, Users, Roles and Stock Units, sack details for the various Remittance outs and cut-off events for cut-off reports.

The steps to be executed by the interface step are:

- Fetch records for the current day's Trading Date from the in-day migration TPS source table (TMS_RX_HNGX_MIGRATION_DAY).
- Insert those records into the target table TPS_HYDRA_INDAY_DATA in BRDB while enriching it with the Fad-Hash value for the Branch.

Copied records should be marked in the source TPS table as 'processed'.

The source and target tables are both non-partitioned.

The source and target data attribute mappings / derivations are defined in the Branch Database – Legacy Host Interface Specification [R17].

### 5.3.4.2 APS

The BRDB – APS batch interface will perform one function (**BRDB Transactions to APS**) of once a day transfer of AP transactional data from Branch Database to APS to allow APS to onward route the transactional data to the AP Clients.

#### 5.3.4.2.1 BRDB Transactions to APS

The batch feed will run after the logical event "Start of BRDB-Batch" has occurred.

This feed is characterised by high data volumes however the implementation is simple as is depicted by the following diagram:

POL00115409



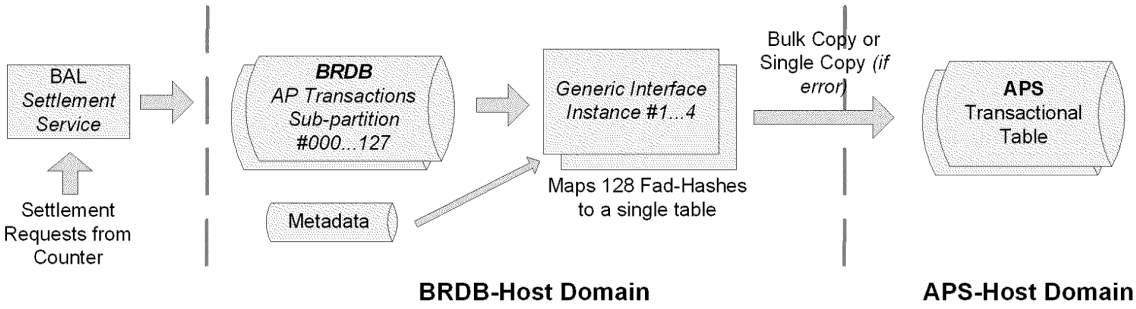



**Figure – 9 BRDB Transactions to APS**

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances of the generic bulk copy interface running for each of the following transaction type:

- Automated Payment Transactions

The interface steps involve:

- Fetching records for the current day's Trading Date from the appropriate Fad-Hash sub-partition in the BRDB source table (BRDB_RX_APS_TRANSACTIONS) where the TRANSFERRED_FROM_LEGACY_YN flag value is set as 'N'.
- Inserting records into the un-partitioned target table in APS (TMS_RX_APS_MCBC_TXNS).

Copied records need not be marked in the source BRDB tables as 'processed'.

The 128 Fad-Hash sub-partitions in BRDB will be mapped to a single APS transaction table.

The source and target data attribute mappings / derivations are defined in the Branch Database – Legacy Host Interface Specification[R17].



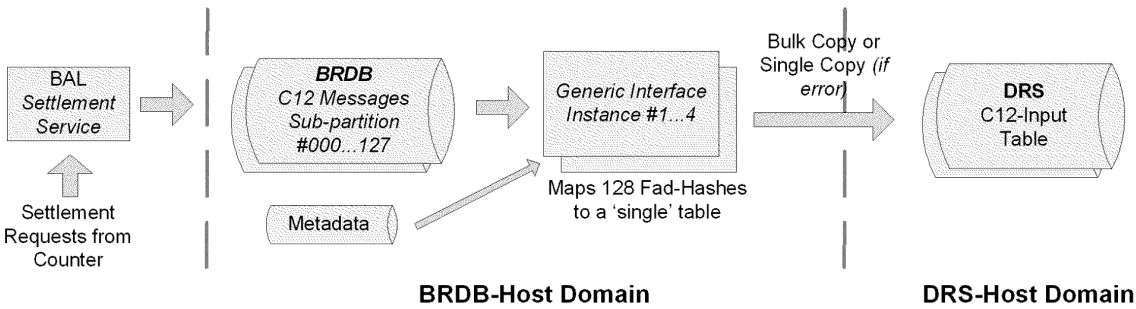
### 5.3.4.3 DRS

The BRDB – DRS batch interface will perform one function (**BRDB Transactions to DRS**) of once a day transfer of C12 confirmation messages from the Branch Database to DRS to allow DRS to reconcile the transaction parts and in case of Debit-Card C12 messages, to onward route to streamline.

#### 5.3.4.3.1 BRDB Transactions to DRS

The batch feed will run after the logical event "Start of BRDB-Batch" has occurred.

This feed is characterised by high data volumes however the implementation is simple as is depicted by the following diagram:



POL00115409



**Figure – 10 BRDB Transactions to DRS**

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances of the generic bulk copy interface running for each of the following transaction types:

- C12 confirmation messages for Network Banking and E-topups
- C12 confirmation messages for Debit Card

The interface steps involve:

- Fetching records for the current day's Trading Date from the appropriate Fad-Hash sub-partition in the BRDB source table.
- Inserting records into the target table in DRS. Even though the target table is hash-partitioned, it needs to be accessed at table level as opposed to partition level. Even distribution of transactions across all eight partitions in the DRS input table can be achieved by using the DRS sequence TMS_REC_SEQ.

Any records that could not be copied across to DRS due to a data-related exception (as opposed to other exceptions such as system level or connection related) will be logged in the DRS Input Exceptions table (DRS_C12_INP_EXCEPTIONS).

The source and target data attribute mappings / derivations are defined in the Branch Database – Legacy Host Interface Specification[R17].



#### 5.3.4.4 LFS

The BRDB – LFS batch interface will perform the following two functions:

- **BRDB messages to LFS:** Twice a day transfer of Cash Declarations and hourly transfer of Pouch Collection and Pouch Delivery details from BRDB to LFS for onward routing to SAPADS.
- **LFS messages to BRDB:** Transfer of Planned Orders (typically once per day) & Replenishment Delivery details (up to 16 times a day) from LFS to BRDB. These messages will be relayed by the Branch Access Layer to the Counter in response to requests from the Counter.

##### 5.3.4.4.1 BRDB messages to LFS

The batch feed will run at regular intervals throughout the day depending on the type of feed..

This feed is characterised by low data volumes and simple implementation.

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances of the generic bulk copy interface running for each of the following types of messages:

- Cash Declarations
- Pouch Deliveries
- Pouch Collections

The interface steps involve:

- Fetching all messages that have not been copied across i.e. LFS-Delivered-Timestamp is not set, for the above-mentioned types from BRDB.
- Inserting those records into the target tables in LFS. Inserts of detail records need to match header records. Any data exception in header/detail record should result in corresponding detail/header records to not be copied across. For such records, the LFS-Delivered-Timestamp should remain un-set.
- Updating the LFS-Delivered-Timestamp for all copied records in BRDB to mark them as 'copied'.

POL00115409



For each message type, the above steps need to form a single commit transaction to ensure data integrity and a consistent restart point. Messages belonging to all Outlets need to be copied across regardless of whether the Outlet is trading under Horizon or HNG-X.

The source and target tables and data attribute mapping / derivation are defined in the Branch Database – Legacy Host Interface Specification [R17].

#### 5.3.4.4.2 LFS messages to BRDB

The batch feed will run at the tail end of the PLO, RDC file processing schedules in LFS. Since these LFS schedules run on-demand, the copy jobs may run within core hours.

This feed is characterised by low data volumes and simple implementation.

It is implemented in the form of four instances (one on each node) of the generic bulk copy interface for each of the following types of messages:

- Planned Orders
- Replenishment Delivery Messages

The interface steps involve:

- Fetching all messages that have not been copied across i.e. HNGX-Actioned-Tsmp flag is not set, for the above-mentioned types from LFS.
- Inserting records into the target tables in BRDB. Inserts of detail records need to match header records. Any data exception in header/detail record should result in corresponding detail/header records to not be copied across. For such records, the LFS-Delivered-Timestamp should remain un-set.
- Updating the HNGX-Actioned-Tsmp flag to SYSTIMESTAMP for all copied records in LFS to mark them as 'copied'.



The above steps need to form a single commit transaction to ensure data integrity. Messages belonging to all Outlets need to be copied across regardless of the Outlet is trading under Horizon or HNG-X.

The source and target tables and data attribute mapping / derivation are defined in the Branch Database – Legacy Host Interface Specification [R17].

### 5.3.4.5 RDDS

The BRDB – RDDS batch interface will perform the following functions:

- **BRDB Reference Data from RDMC/RDDS:** Once a day transfer of Reference Data to BRDB. The interface will transfer data from RDDS to be used by BRDB-Host & the BAL for internal processing and will also transfer Outlet opening times from RDMC to be supplied in future to the helpdesk system.
- **BRDB Host Reference Data to RDDS:** Once a day transfer of Estate Management Data from BRDB to RDDS. This data will be used by RDDS for its internal processing.
- **BRDB Host Reference Data to RDMC:** Once a day transfer of Estate Management Data from BRDB to RDMC. This data will be used by RDMC for its internal processing.
- **Counter Reference data from RDDS:** Once a day (fixed) plus on-demand transfer of counter specific reference data from RDDS to BRDB. This includes data such as Bureau De Change spot rates, emergency counter ref-data changes and desktop memos.

The following diagram depicts the data flows between RDMC/RDDS and BRDB.

POL00115409



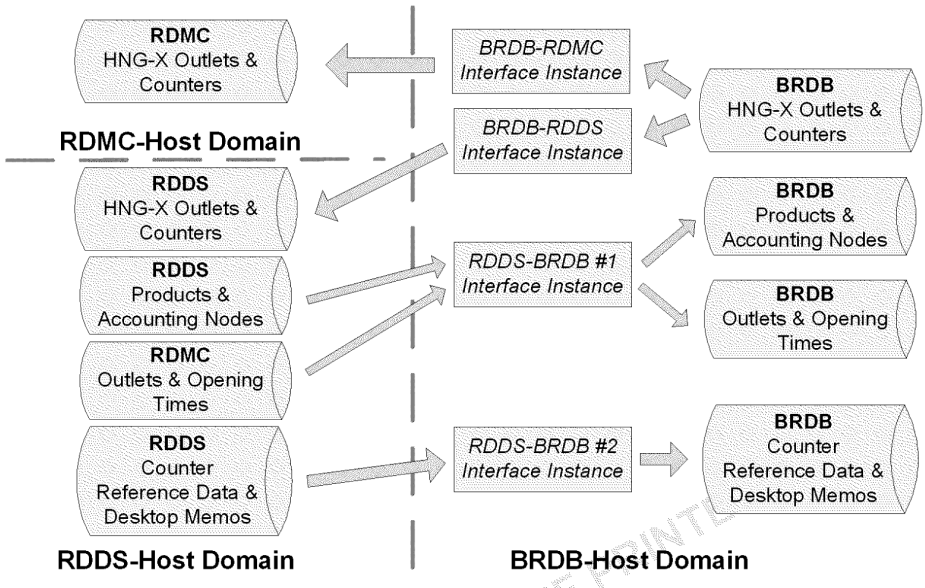



**Figure – 11 BRDB interfaces with RDMC/RDDS**



### 5.3.4.5.1 BRDB Host Reference Data from RDMC/RDDS

The RDDS–to–BRDB host feed will populate the Branch Database with Products, Accounting Periods and Branch information. The interface will also populate the Branch Database with Counter Report definitions.

Additionally the interface will also populate the Branch Opening timings information for use by the helpdesk system from the RDMC database.

The batch feed will run once a day and is triggered by the RDDS system operational schedule.

This feed is characterised by very low data volumes and a simple implementation. The interface is implemented in the form of a single instance copy job. The process unit is at the interface level i.e. all interface steps will be executed as a part of a single commit.

This interface covers the following Branch Database reference data tables:

RDDS_BRANCHES

RDDS_ACCOUNTING_NODES

RDDS_PRODUCTS

RDDS_BRANCH_OPENING_PERIODS

The instance will run on a pre-defined BRDB-Host node and connect to the instance using bequeath protocol. BRDB node/instance failure recovery actions involve a single retry by re-running on one of the other available nodes by using the BRDB service. The scheduler will provide an extra command-line parameter to facilitate a rerun. RDDS node/instance failure will not result in a rerun.

The interface steps for copying host reference data to BRDB involve:

- Deleting all host reference data that is about to be refreshed from the source system.
- Fetching new reference data from RDMC/RDDS and inserting into the target tables in BRDB.

POL00115409



- Deriving the current day's de-normalised view of the Accounting node to Product mappings. This de-normalisation is described in detail below.

  **Accounting Node – Product De-normalisation**

  The table BRDB_ACC_NODE_PRODUCT_MAPPING provides a de-normalised materialized view that identifies each product with all of the accounting nodes to which it is related at each level in the hierarchy.

  For each accounting node, all the products that either directly or indirectly maps to it are extracted from the BRDB_CURRENT_ACCOUNTING_NODES view and inserted into the BRDB_ACC_NODE_PRODUCT_MAPPING table.

  The materialized view should be refreshed once its source tables in BRDB have been refreshed.

The source and target tables and data attribute mapping / derivation are defined in the Branch Database – RDMC/RDDS Interface Specification [R19].

#### 5.3.4.5.2 BRDB Host Reference Data to RDDS

The BRDB–to–RDDS host feed will populate RDDS with up to date status information on Branches using data provided by Estate Management. The interface will be owned by RDDS-Host and is discussed in the RDDS High Level Design [R32].



#### 5.3.4.5.3 BRDB Host Reference Data to RDMC

The BRDB–to–RDMC host feed will populate RDMC with up to date status information on Branches. The contents and mechanics of the interface are owned by the RDMC-Host and are discussed in RDMC High Level Design [R33].

#### 5.3.4.5.4 Counter Reference Data from RDDS

This on-demand batch feed will run at various times during the day and will be driven by the RDDS system operational schedule.

This feed is characterised by low data volumes but a more complicated implementation. The interface is implemented in the form of a single instance, which will run on a pre-defined BRDB-Host node and node failure recovery actions involve running the same job on one of the other nodes.

This interface covers the following Branch Database counter reference data tables:

RDDS_PACKAGE_TYPE

RDDS_PACKAGE

RDDS_PACKAGE_CONTENT

RDDS_DELIVERY_TYPE

RDDS_DELIVERY

**Interface Steps**

This interface is different from the common interface functionality identified in Section 5.3.2 hence this section describes the interface steps involved in detail.

The BRDB interface process will be invoked by an RDDS-host process via TWS. The BRDB Interface will be passed in the command-line arguments that are mandatory for the interface i.e. Interface-Name, Business Date and Node-Id.

POL00115409



The BRDB interface will access the RDDS table RD_TRANSFER_CONTROL to read the list of unprocessed Package-Types i.e. records where status is 0 (not-started) or 1 (Started but not completed). Records will be fetched in chronological order using the creation_date column.

For each Package-Type fetched, the BRDB interface will update the RD_TRANSFER_CONTROL table to set the status to 1 and lock the record and will use the Package-Type (e.g. SPOTRATES, MAIN etc) as an argument[7].

The BRDB interface will identify the mismatches between the above-mentioned Branch Database tables and their corresponding tables in RDDS by issuing SQL queries. Queries will be based on Primary-Key values and may involve joining parent tables where necessary to filter on the input package-type argument.

SQL Template #1 to identify records to be inserted into the Branch Database table:

```
SELECT          Primary-Key-values
FROM            RDDS-table-name(s)
WHERE           package-type = 'Input-package-type'
(optional AND…)
MINUS
SELECT          Primary-Key-values
FROM            BRDB-table-name(s)
WHERE           package-type = 'Input-package-type'
(optional AND…)
```

SQL Template #2 to identify records to be deleted from the Branch Database table:

```
SELECT          Primary-Key-values
FROM            BRDB-table-name(s)
WHERE           package-type = 'Input-package-type'
(optional AND…)
MINUS
SELECT          Primary-Key-values
FROM            RDDS-table-name(s)
WHERE           package-type = 'Input-package-type'
(optional AND…)
```



The above-mentioned templates apply to all of the BRDB counter reference data tables with the exception of RDDS_PACKAGE and RDDS_PACKAGE_TYPE.

For RDDS_PACKAGE the Select-list needs to include the column EXPIRY_DATE in addition to the primary keys for the table. This is to cover scenarios where emergency changes to the RDDS-equivalent tables involve expiring the current version of counter reference data and replacing it with a new version. Note that Expiry-Date column can have Null values.

The RDDS_PACKAGE_TYPE table will not be refreshed on a nightly basis. It will be refreshed whenever the value of system parameter called "*Feed-Name-in-uppercase*_REFRESH_YN" is set to "Y".

Once all the changes have been made to the counter Reference-Data, the transfer-control table will be updated to set the status to 2 and record the completion-date. All records for the reference data change will either be committed or rolled back (in case of an error) as one unit alongside the transfer control update. In case of a failure, the Transfer-Control status for that package type must be left at its current state and an operational exception must be raised to make support aware of the failure. However the process should exit with success.

The BRDB interface process will also be capable of checking and using the value of system parameter called "*Feed-Name-in-uppercase*_REFRESH_YN" of "Y" to indicate a complete refresh of all data in BRDB Counter reference data tables. Complete refresh involved deleting all records from the target tables in BRDB and refreshing with data from source tables in RDDS. The system parameter value will

[7] With one exception as discussed later in this section.

POL00115409



usually be set to "N" is only meant to be used by Support in exceptional circumstances and this will be covered in more detail in the Branch Database Support Guide.

The TWS implementation of this job for Spot-Rates is in the form of an on-demand schedule that is triggered by the successful completion of the RDDS TWS job that processes the Counter Reference Data into the RDDS interface tables. The TWS implementation of the job for all packages other than 'Spot-Rates' is in the form of a job that typically runs once a day but whose execution is dependent on the completion of the RDDS TWS job that processes Counter Reference Data and makes it available in the interface tables in RDDS.

The source and target data attribute mapping / derivation is driven off metadata and is defined in the Branch Database – RDDS Counter Reference Data and Memos Interface Specification [R20].

#### 5.3.4.5.5 Desktop Memos from RDDS

This on-demand batch feed will run at various times during the day and will be driven by the RDDS system operational schedule.

This feed is characterised by very low data volumes and a simple implementation. The interface is implemented in the form of a single instance, which will run on a pre-defined BRDB-Host node and node failure recovery actions involve running the same job on one of the other nodes.

This interface covers the following Branch Database desktop memo tables:

RDDS_DESKTOP_MEMO

RDDS_DESKTOP_MEMO_DISTR

**Interface Steps**

This interface is similar to the common interface functionality identified in Section 5.3.2 but with some variations.

The BRDB Memo interface process will be invoked by an RDDS-host process. The BRDB Memo interface will be passed in the command-line arguments that are mandatory for the interface i.e. Interface-Name, Business Date and Node-Id.

The BRDB interface will access the RDDS table MEMO_TRANSFER_CONTROL to read the list of unprocessed Memo-Ids i.e. where status is 0 (not-started) or 1 (Started but not completed). Records will be fetched in chronological order using the creation_date column.

For each Memo-Id fetched, the BRDB interface will update the MEMO_TRANSFER_CONTROL table to set the status to 1, lock the record and will use the Memo-Id to fetch the memo and its distribution details from RDDS and populate the corresponding BRDB tables. Once memo details are copied across, the transfer-control table will be updated to set the status to 2 and record the completion-date. All records for the 'Memo-Id' will either be committed or rolled back (in case of an error) as one unit along with the transfer control update.

If the Memo was already present in the Branch Database, the interface will raise & return an exception and will not attempt to process the memo.

The BRDB interface process will also be capable of checking and using the value of system parameter called "*Feed-Name-in-uppercase*_BRDB_REFRESH_YN" of "Y" to indicate a complete refresh of all data in BRDB Desktop Memo tables. Complete refresh involved deleting all records from the target tables in BRDB and refreshing with data from source tables in RDDS. The system parameter value will be defaulted to "N" and is only meant to be used by Support in exceptional circumstances and this will be covered in more detail in the Branch Database Support Guide.

The TWS implementation of this job is in the form of an on-demand schedule that is triggered by the successful completion of the RDDS TWS job that makes the memos available in the RDDS interfacing

POL00115409



tables. A successful completion of the Memo copy to Branch Database will trigger the memo de-normalisation job (see Section 5.4.5.1).

The source and target data attribute mapping / derivation is driven off metadata and is defined in the Branch Database – RDDS Counter Reference Data and Memos Interface Specification [R20].

#### 5.3.4.6 NPS

The BRDB – NPS batch interface will perform one function (**BRDB Transactions to NPS**) of a interactive batch transfer feed of Track & Trace (T&T) and Guaranteed Reversal (GREV) messages from the Branch Database to NPS to allow NPS to forward the T&T messages as SOAP requests to POL EDG and to use the GREV messages to ensure that the FIs and POA financial reconciliation are aware of the reversal.

##### 5.3.4.1.1 BRDB messages to NPS

The interface will be implemented in the form of an interactive batch feed that will run both within and outside of core business hours.

This feed is characterised by medium/high data volumes and a more complex implementation due to the interactive nature of the feed.

It is driven by the Branch Database system operational schedule and is implemented in the form of four instances of the generic bulk copy interface running for each of the following types of messages:

- Track & Trace
- Guaranteed Reversals



The interface steps involve:

- Checking for a signal to exit processing. The signal is sent by an updating the value of System Parameter named <Feed-Name>_STOP_YN (see 7.2.11 for details). There will be separate system parameter flags for GREV and T&T interfaces.
- Fetching all messages that have not been copied across i.e. NPS-Delivered-Timestamp is NULL, for the above-mentioned record types from BRDB.
- Inserting records into the target tables in NPS.
- Updating the copied records in BRDB to mark them as 'copied' by set the value of NPS-Delivered-Timestamp to SYSTIMESTAMP.
- Checking for a signal to exit processing as before.
- 'Sleep' for a metadata-defined time period[8] before waking up and repeating the above-mentioned steps.

The Track & Trace feed to NPS will connect to the NPSSERVICE1_2 service and the Guaranteed Reversals feed will connect to NPSSERVICE2_1.

The source and target tables and data attribute mapping / derivation are defined in the Branch Database – Legacy Host Interface Specification [R17].

[8] Sleep time in seconds will be stored in BRDB-System-Parameters table. This needs to be queried once only, at the start of the processing.

POL00115409



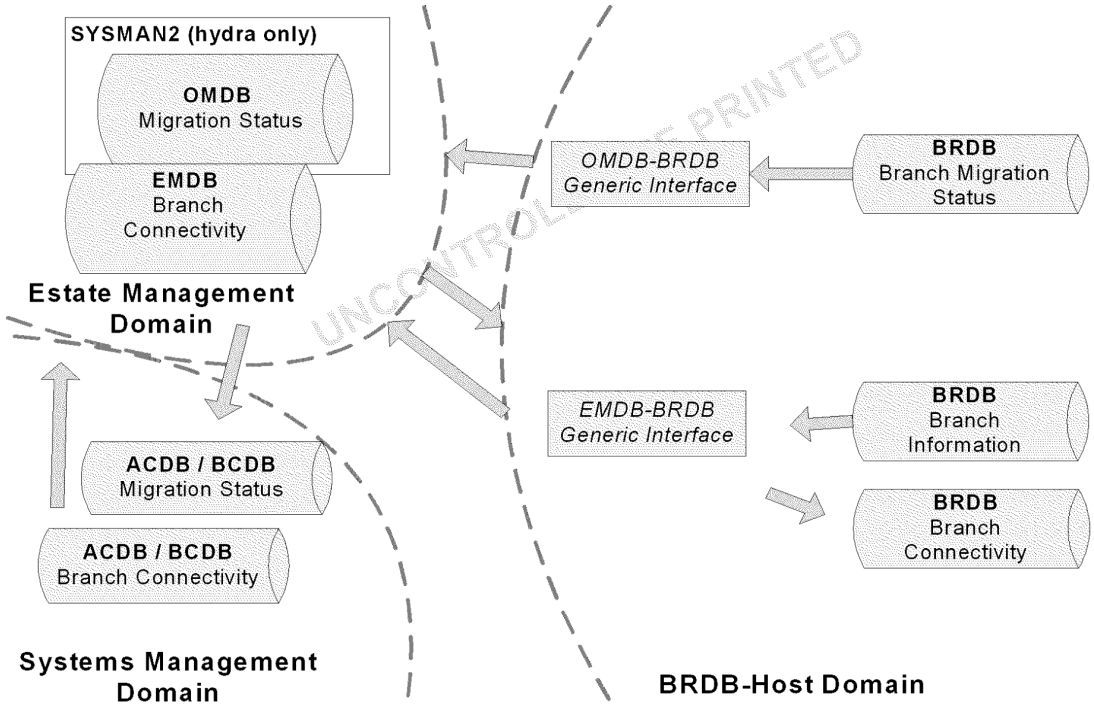
## 5.3.5 Estate and Systems Management Interfaces

The Branch Database will interface with various Estate and Systems Management systems to read networking information about Branches and Counters within Branches and to transfer the details of HNG-X Migration Confirmation to Systems Management via EMDB.

Networking information involves reading details such as the Connection type available at the Branch and the full IP address of each counter in an HNG-X branch. For the period up to a point before the start of Windows-XP Rollout, the information will be sourced from ACDB via EMDB. After that the information will be sourced from a new system called BCDB via EMDB. BRDB also provides EMDB with the definitive view of branch information via a view on the reference data tables and BRDB_BRANCH_INFO.

Additionally for the duration of Hydra, the Branch Database needs to provide ACDB / BCDB via OMDB with information on Branches that have reached the point of no return in HNG-X migration.

The following diagram depicts the Branch Database interfaces with Estate and Systems Management.




### 5.3.5.1 EMDB Branch Connectivity to BRDB

The interface will be implemented as an instance of the generic interface for feeds going out of the EMDB system. The feed to BRDB will be in the form of an infrequent repopulation of the corresponding tables in EMDB and will be initiated and owned by EMDB.

The Branch Database tables populated by this interface are:

EMDB_POST_OFFICE

POL00115409



EMDB_MANAGED_NODE

The interface steps involve:

- Deleting all Branch Connectivity data in the Branch Database EMDB target tables, as they are about to be refreshed from the EMDB source tables.
- Repopulate the EMDB tables within Branch Database.

Refer to the EMDB HLD [R25] for more details.

A job to populate and update the BRDB_BRANCH_INFO and BRDB_BRANCH_NODE_INFO tables will run as part of the TWS batch schedule BRDB_FROM_EMDB. The implementation of the BRDB_EMDB_INTERFACE process is described in section 5.1.4.2.



#### 5.3.5.2 BRDB Outlet Migration Status to ACDB (hydra only)

The interface will be implemented in will run once a day outside of core business hours.

This feed is characterised by very low data volumes and a more complex implementation due to the interactive nature of the feed and the fact that an Oracle database is accessed by SQL*Server.

It copies HNG-X Branch trading status across to ACDB via OMDB (also see CP4928).

This interface is Hydra only.

The source and target tables and data attribute mapping / derivation are defined in the Branch Database – ACDB/BCDB Interface Specification [R34].

### 5.3.6 HNG-X Migration Schedule from CS

The Branch Database will receive a once-a-day view of the day's view of the Migration schedule. The schedule is owned by the Post Office and is maintained by the Post Office / CS and is stored in RDDS.

Branch Database has access via a database link of the RDDS migration schedule which is stored in RD_MIGRATING_BRANCHES. This is then used by the BRDBX034 hydra_prep_recon package, after start of day, to obtain the list of branches migrating within the next 14 days via procedure Get_Migrating_Branches. See HNG-X RDDS Host High Level Design [R32].

For the processing of migration data please see BRDB Processing of the End of Day Migration Data [R40] and BRDB Processing of the In Day Migration Data [R42].

### 5.3.7 Branch Access Layer Interface

Branch Database receives XML-based requests from Post Office branches through the Data Access Service in the Branch Access Layer.

The application interface specification between the Branch Access Layer's Data Access Service and the Branch Database is defined in [R21].

POL00115409



## 5.4 Aggregation and De-normalisation

### 5.4.1 Overview

The Branch Database receives transactional, control and reference data all through the day. Some of the accounting, reporting and operational requirements necessitate the data to be aggregated / de-normalised as a part of (usually but not always) overnight processing. The aggregations for the Legacy Host systems are emulating reconciliation totals that were generated by the Horizon Counter. These aggregations need to be carried out for all transactions for a trading day as soon as possible after the trading day has ended. Given that a Trading Day can span multiple Journal Dates, these aggregations will not be confined to a single partition. However a Trading Date cannot start earlier than the Journal Date which is one day earlier than the Trading Date and this fact can be used to constrain the scanning of the BRDB tables. Such aggregation must be carried out prior to the data being transferred to the Legacy Host systems. Aggregation for TPS and APS Hosts are independent of each other.

Given that the Trading Day ends at 19:00 (local time - as defined by the BAL), then these processes can run soon after that time. Their actual running will be controlled by TWS and the tasks should be scheduled at around 19:15 each night. The process that copies the day's transactions to the Legacy Hosts should be dependent upon the corresponding aggregation processes.

The aggregations for Reporting are based on the data in a single partition based on Journal Date which is based on the clocks at the counter. Since Journal Date is held in UTC time, such aggregation cannot take place until after 01:00 (which is when the date changes in BST). To allow for any counter drift, the aggregation process should not start until at least 01:15. Again these processes will be scheduled by TWS. Once all the aggregation for reporting is complete, the System Parameter LastDailySummaryDate is updated to the current Business Date to indicate to the reporting processes that the aggregation is complete.

Given the way that data is partitioned using fad_hash, it is possible to run a number of parallel aggregation processes. A minimum of 4 such processes should be run (one for each Oracle Instance).

Further details on aggregations can be found in [R39]

### 5.4.2 Metadata Driven Mechanism

The aggregations / de-normalisations outlined earlier can be best performed by following a metadata driven approach. This allows for ease of implementation through the use of common code constructs and provides sufficient flexibility in tweaking the SQL with reduced development impact and delivery overheads.

The approach relies on the following characteristics of the operations being performed:

- The source and target tables reside in the Branch Database
- In the event of a failure, the operation needs to be backed out. No partial runs are allowed. Where Fad-Hash partitioned tables are involved, the atomicity is at Fad-Hash level.
- If more than one SQL statements are required to perform the logical operation, the multiple SQL statements will be defined in metadata and will be run as a single commit unit.

The metadata table that will drive the aggregations / de-normalisations is called BRDB_HOST_AGGREGATIONS. The following table lists some of the important attributes of the table:

| Column Name | Data type & Size | Mandatory | Key | Description |
|---|---|---|---|---|
| Aggregation-Name | Varchar2 (50) | Yes | Yes | Identifies the aggregation / denormalisation SQL statement. This will be passed on |

POL00115409



| | | | | |
|---|---|---|---|---|
| | | | | command-line as the first input parameter to the generic aggregation module as described in 7.2.7. |
| SQL-Id | Number (3) | Yes | Yes | Uniquely identifies each aggregation SQL for an Aggregation / Denormalisation. It is assumed that scripts with the same aggregation name, but different SQL ids will have the same value for use_fad_hash. |
| Aggregation-SQL / Aggregation-SQL-continued | Varchar2 (4000) each | Yes / No | No | Stores the Aggregation / Denormalisation SQL. If the SQL is larger than 4000 characters, the Aggregation-SQL-continued column is used to store the remainder.<br>All SQLs will make use of the Input Trading/Business Date and Node/Instance Id input parameters as described in the generic aggregation module as described in Section 7.2.7. It is left to the relevant low level designs to define how the parameters will be substituted into the Aggregation SQLs. |
| Use Fad-Hash | Varchar2 (1) | Yes | No | Indicates whether the aggregation to be performed needs to be Fad-Hash aware. If the flag value is set, the aggregation module will use the input Node-Id parameter (see Section 7.2.7) to identify the Fad-Hash values for the Node and run the Aggregation SQL for each Fad-Hash value.<br>Note that if there are more than one SQLs associated with an aggregation, the value of this flag for the first SQL applies to all others. |



The Aggregation / Denormalisation SQL definition may contain one or more of the following placeholders, which needs to be substituted by the Aggregation Routine, typically with parameter values passed on the command-line. Note that these are case sensitive and must be delimited by the # sign:

| | |
|---|---|
| #BUSINESS-DATE# | This is a date in the format YYYYMMDD.<br>It indicates the business day of aggregation and may either be a calendar day or the trading day. |
| #INSTANCE-ID# | Numeric value of up to 2 digits indicates the Branch Database Node/Instance Id to use to run the Aggregation / Denormalisation. |
| #FAD-HASH# | Numeric value of up to 3 digits indicating the Fad-Hash to filter on. Note that if the Use-Fad-Hash flag in the brdb_host_aggregations table is not set, this placeholder should not be present in the SQL. If found to be present, an exception should be raised. |
| #OPTIONAL-PARAM1# | Alphanumeric value of up to 100 characters long. |
| #OPTIONAL-PARAM2# | Alphanumeric value of up to 100 characters long. |
| #OPTIONAL-PARAM3# | Alphanumeric value of up to 100 characters long. |

#### 5.4.2.1 Process Distribution Across Nodes

The actual implementation of all SQLs that involve Fad-Hash based processing needs to take into account Fad-Hash – Instance mappings to perform the aggregation on each Branch Database node.

POL00115409



Aggregations can either be done once for each Fad-Hash i.e. in 128 steps or just once to cover all the Fad-Hashes that 'belong' to the node.

It is recommended that where possible, aggregations should be done at Fad-Hash level. However for some aggregations, it may be sufficient to run four instances of the SQL, one on each node. The Fad-Hash filter can be applied in the form of an additional *where clause*, as shown below:

```
WHERE  …
AND    fad_hash IN (    SELECT  fad_hash
                        FROM    brdb_fad_hash_current_instance
                        WHERE   instance_id = :current_instance_id)
```

Use of parallel query should be made where appropriate.

#### 5.4.2.2 Table Access by Fad-Hash based on Node availability

As with most operations on the Branch Database involving transactions, all partitioned tables in the Branch Database must be accessed is a fad-hash aware manner i.e. Fad-Hash (sub) partitions of the table must be accessed using the correct instance. The view BRDB_FAD_HASH_CURRENT_INSTANCE provides the 'current' Fad-Hash to instance mappings for all available instances.



#### 5.4.2.3 Handling Exceptions

As the operations do not handle data errors differently from system errors, any exception that occurs while attempting to run the SQL that performs the action will result in the process performing an Oracle Rollback and exiting with failure.

### 5.4.3 Failures and Restartability

The logical processing unit for all tables involving Fad-Hash based data is at a Fad-Hash level. The use of HADDIS-compliant process-control measures will ensure that when the data for a Fad-Hash value has been transferred successfully, process-control will record the fact and prevent the same partition from being re-processed in the event of a restart. Use of process-control also ensures that in the event of a restart after failure, the failed Fad-Hash is reprocessed.

For operations that do not involve Fad-Hash partitioned / sub-partitioned tables, the logical process unit is the entire operation. In the event of a restart after failure, the action is performed again from the beginning.

Note that for the restartability to work correctly, the operation must be able to rollback any data changes that were made before the failure occurred.

#### 5.4.3.1 Handling Node Unavailability

Node/Instance unavailability is handled in the same manner as for Batch Interfaces – See Section 5.3.3.1.

POL00115409



## 5.4.4 Host Aggregations

## 5.4.5 Host De-normalisations

### 5.4.5.1 Desktop Memo De-normalisation

Desktop memos are supplied by the RDDS application and the data is provided in two tables:

RDDS_DESKTOP_MEMO – Provides the definition of the Desktop Memo and the User-Role to whom it is targeted.

RDDS_DESKTOP_MEMO_DISTR – Provides the distribution list for the memo in the form of Branch Accounting Codes.

The memos need to be made visible to every user belonging to the User-Role in each Branch from the memo distribution list. There is a requirement to track whether the memo has been read by the user.

The de-normalisation simply populates a new table with the Memo-Id and the user list. This data is stored in the table BRDB_DESKTOP_MEMO_USER_DISTR. This action needs to run on the completion of the job that copies across desktop memo details from RDDS. There may be more than one runs of this job during the business day.

The algorithm for de-normalising the memo distribution is:

```
SELECT      branch_accounting_code,
            fad_hash,
            memo_id,
            branch_user,
            brdb_received_timestamp,
            SYS_CONTEXT('USERENV', 'INSTANCE_NAME')
FROM        rdds_desktop_memo rdm,
            rdds_desktop_memo_distr rdmd,
            brdb_branch_user_roles bbur
WHERE       rdm.memo_id = rdmd.memo_id
AND         rdmd.branch_accounting_code = bbur.branch_accounting_code
AND         rdm.user_group = bbur.branch_user_role
```

The de-normalisation will be available in three forms.

The first is at an individual Memo-Id level, which will de-normalise the memo for all Branches, as defined by the distribution. To implement this, the SQL will contain an additional “AND memo_id = #OPTIONAL-PARAM1#”. The Optional-Parameter-1 will be passed on command-line to the Aggregation module. This de-normalisation will be used on an ongoing basis to process a new memo coming from Reference data.

The second is at an individual Branch level and will cover all existing memos in the Branch Database. To implement this, the SQL will contain an additional argument “AND branch_accounting_code = #OPTIONAL-PARAM1#”. The Optional-Parameter-1 will be passed on command-line to the Aggregation module. This de-normalisation will be used for hydra as a part of the ‘In-Day’ migration processing and also on an ongoing basis as a step in setting up a new Branch in the Branch database.

The third form is at a gross level and will cover all existing memos in the Branch Database. The difference between its SQL and the SQLs used by the first and second form will be the absence of the argument that limits on any specific Memo-Id or Branch-Accounting-Code. The SQL will remain as shown in the template earlier. This will be used once as a part of the preparation of the Branch Database for the first business day only and thereafter only under exceptional circumstances.

The three forms of de-normalisation will be represented as separate sets of metadata in the host aggregations table.

POL00115409



Since the volumes involved are very low (a few thousand), processing should be at process level and not fad-hash level with the exception of the aggregation labelled DEN_DESKTOP_MEMO_REFRESH, which will be at Fad-Hash level.

## 5.5 Migration

As per the Migration Strategy [R22], there will be a significantly large period of time after data centre migration, during which Horizon Outlets will progressively move across to HNG-X. This time period is known as 'Hydra' or dual running.

In order to facilitate the migration of Horizon Outlets to HNG-X, the Branch Database needs to be kept up to date with the Horizon transactional and reconciliation data until the Outlet moves across to HNG-X. This is known as the 'ongoing' migration feed.

There is also a need to 'feed' the Branch database with critical access (users, roles, privileges) and stock unit information as a part of the Outlet switchover to HNG-X. This is known as the 'in-day' migration feed and is a one-off feed for each Outlet migrating to HNG-X.

This section discusses the migration feeds from a Branch Database perspective.

Before a Horizon Outlet can migrate to HNG-X, the following categories of Branch specific data needs to be available in the Branch Database:

**End of Day Migration Feed**

- Transaction log covering all transactions (ideally) for the past 42 days or as a minimum, all transactions since the beginning the Outlet trading period and the Outlet remuneration period, whichever starts earlier.
- A reconciliation feed from Horizon counters covering the same duration as the transaction log (from above).

[R40] describes the processing required within the Branch Database for above.

**In-day Migration Feed**

- Usernames and roles of all existing Outlet end users
- Stock Unit Definitions
- Current Cash and stock positions
- Suspense Accounts
- Transaction Correction statuses of all TCs targeted at the Branch over a period of 42 days (lifecycle of the message store)
- Track & Trace "Sack" details
- REM Sack details.
- Cut off data for cut off reports
- Accounting information such as the current Trading Period & Balance Period
- The starting values of various Sequence Numbers such as Journal Sequence and AP Transaction Sequence

[R41] describes the processing required within the Branch Database for above.

The End of Day and In-day migration feeds will be sourced from TPS-Host. The transfer mechanism is discussed in Section 5.3.4.1.5 in further detail. For the processing of migration data please see BRDB

POL00115409



Processing of the End of Day Migration Data [R40] and BRDB Processing of the In Day Migration Data [R42].

## 5.6 Training

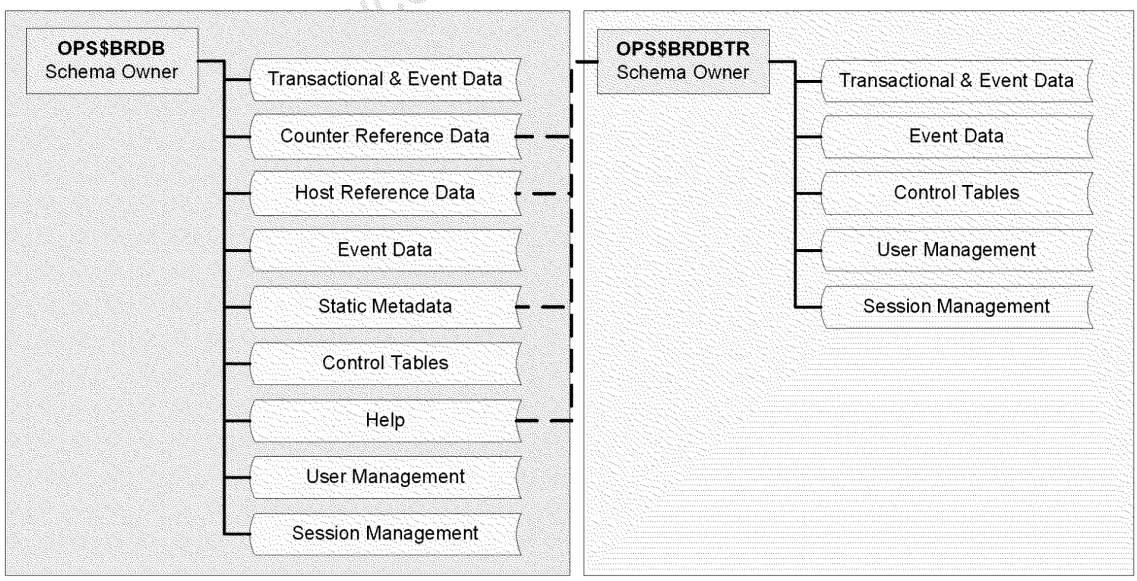
The Branch database will store a copy of part of the "live" schema under a different ("training") schema owner to support the training requirements. This second copy of the schema will be identical to the first copy with a few exceptions:

- All tables that have been implemented as a 'rolling window' e.g. the message journal, will contain just a single partition created for a date in the distant future e.g. 31st December 3000. This will allow the training schema to be operational without any overheads of partition management.
- The training schema will not contain any host or counter reference data objects but will be given a read-only view (through private synonyms) on the corresponding tables in the live schema. The same applies for any static metadata objects.
- All training schema objects will share the same tablespaces as the live schema objects but the actual table sizes will be much smaller (no more than 1% of the Live size).

The training schema will be accessible to the Branch Access Layer via a separate set of users – See Appendix C – User, Role Sequence Definitions and Database Links for details.

Table in thetraining schema will not be housekept, data will be deleted as part of training reset.

The following figure depicts the shared and exclusive schema objects for the Training users.




### 5.6.1 Training Metadata Setup

Training sessions will be setup under a (virtual) Branch-Accounting-Code. This is a numeric value that is statically assigned to a CTO Branch-Code and Node-Id combination. POL will confirm a static range of values to be used for these virtual Branch Accounting Codes. Any new Branch Node that is set up in the table BRDB_BRANCH_NODE_INFO will be assigned a virtual Branch-Accounting-Code if the Branch

POL00115409



can be identified as a CTO branch. The de-normalised Is-CTO-Office flag will be set to 'Y' to indicate that the Branch is a CTO office. Virtual Branch-Accounting-Code are assigned when processing the Estate Management feed.:

### 5.6.2 Training Session Setup

The establishment of a new training session requires the BAL to do an extra lookup of the BRDB_BRANCH_NODE_INFO table to fetch the Branch-Accounting-Code to be used for that training session.

As a part of the creation of the training session, the training schema needs to be pre-populated with transactions, events and accounting information. This will be done by the BAL either using SQL statements or through a stored procedure.

### 5.6.3 Training Session Housekeeping

Tables in the training schema will not be housekept. Data will be deleted using a stored procedure when training session is reset by the trainer. Given low data volume this approach does not have any adverse affect on sizing.



## 5.7 Branch Database Supportability

### 5.7.1 Support Interface

The Branch Database provides the various support teams with a simple SQL based interface with the access control managed using Oracle roles.

Any routine support queries should be addressed using the Branch Support System. The Branch Database should be used only in the following cases:

- The Branch Support System (BRSS) is down / inaccessible or the replication between the Branch Database and Branch Support System is down / or too far behind.
- The nature of query requires use of the Branch Database only e.g. querying the AUD$ table for access logs.

Even when the Branch Database is used for support queries, any queries that access tables that are partitioned or sub-partitioned on Fad-Hash will need to use the correct instance for fetching the data. Failure to do so will increase the traffic across the cluster interconnects and could impact operational performance of the Branch Database.

There is a need to reduce the impact of any potential inappropriate access/queries by support team members. Support teams will be restricted to accessing the Branch Database only under an OCP. This is a change in process and will be defined as a requirement in the Branch Database Support Guide.

### 5.7.2 Inserting Balancing Transactions

There is a requirement that the SSC will have ability to insert balancing transactions into the persistent objects of the Branch Database. There are reasons for SSC having to do so e.g. to rectify erroneous accounting data that may have been logged as a result of a bug in the Counter / BAL.

SSC will have privileges of only inserting balancing / correcting transactions to relevant tables in the database. SSC will not have any privileges to update or delete records in the database.

POL00115409



Any writes by the SSC to BRDB must be audited. The mechanism for inserting a correction record must ensure that the auditing of that action performed must be atomic. There also needs a level of obfuscation to ensure that the audit mechanism is robust.

The above-mentioned requirements suggest that there is a need for a correction tool to be delivered which performs the correction, audits it and saves both changes.

A simple low-cost solution for the tool is to provide a Linux shell based utility, which calls a PL/SQL package to perform the changes. The package will allow inserts to the following transactional tables in the Branch Database Live schema with the exception of the Message Journal. All inserts will be audited in the table BRDB_TXN_CORR_TOOL_JOURNAL.

| Object Name | Object Type | Ins | Sel | Upd | Del | Oth |
|---|---|---|---|---|---|---|
| BRDB_RX_BUREAU_TRANSACTIONS | Table | X | | | | |
| BRDB_RX_EPOSS_TRANSACTIONS | Table | X | | | | |
| BRDB_RX_APS_TRANSACTIONS | Table | X | | | | |
| BRDB_RX_EPOSS_EVENTS | Table | X | | | | |
| BRDB_RX_NWB_TRANSACTIONS | Table | X | | | | |
| BRDB_RX_REP_SESSION_DATA | Table | X | | | | |
| BRDB_RX_DCS_TRANSACTIONS | Table | X | | | | |
| BRDB_RX_REP_EVENT_DATA | Table | X | | | | |
| BRDB_RX_CUT_OFF_SUMMARIES | Table | X | | | | |



The values for some of the columns in the Message Journal will need to be derived in order to insert the audit record. The following list shows the derived and non-derived values for the Transaction Correction Tool Message-Journal entry:

| No | Column Name | Value |
|---|---|---|
| 1. | FAD_HASH | Fad-Hash value for the Branch as derived from BRDB_FAD_HASH_OUTLET_MAPPING |
| 2. | BRANCH_ACCOUNTING_CODE | The Branch-Accounting-Code of the Branch for which the correction transaction is required. |
| 3. | NODE_ID | Hard-coded to “99” |
| 4. | JOURNAL_DATE | Date & Time of record insertion into the Message Journal. Use Oracle built-in SYSDATE. |
| 5. | JOURNAL_SEQ_NUMBER | . This is the next sequence number for Node id ‘99’. |
| 6. | SUPPORT_TOOL_USERNAME | SUPPORTTOOLUSER |
| 7. | INSERT_TIMESTAMP | Timestamp of record creation. Use Oracle built-in SYSTIMESTAMP |
| 8. | APP_SERVER_NODE_NAME | ’999’ |
| 9. | BRDB_INSTANCE_NAME | Instance name of the Branch Database Instance. |
| 10. | JOURNAL_XML | The format of the data written to this column is:<br>“<?xml version="1.0" encoding="UTF-8"?><Support_Insert><br><Unix User> Name of the Linux User executing the script </Unix_User><br><Oracle User> Oracle Username that executes the package </Oracle_User><br><Sql> SQL Statement being executed </Sql></Support_Insert>” |

POL00115409



| | | |
|---|---|---|
| 11. | BRANCH_CODE | Same value as for BRANCH_ACCOUNTING_CODE |

Section 7.2.11 describes the correction tool in detail.

### 5.7.3 Support Monitoring

The Branch Database environment needs to be actively monitored by support on three fronts: the RHEL Operating System, Oracle and application. The monitoring facilities offered for HNG-X application systems require the use of different tools to monitor the Branch Database environment.

All monitoring requirements for the Branch Database are discussed in the Host Application Monitoring High Level Design [R31].

## 5.8 Branch Support System Feed

The Branch Database will replicate transactional, event and other relevant data to the Branch Support System (BRSS). The BRSS will act as the primary point of access to such data for the various technical and support streams. The data needs to be replicated on a near real-time basis and in a manner that results in no data loss under normal operational circumstances.

The mechanism chosen for this replication is Oracle Streams. Streams uses three stages for propagating data: Capture, Propagate and Apply. Of these, Capture and Propagate will occur on the Branch Database and Apply on the Branch Support System.



### 5.8.1 Capture Changes in Branch Database

The Oracle Streams capture background process will reside within the Branch Database and capture all DDL and DML events on objects (except temporary and working tables) owned by the Oracle user OPS$BRDB. The capture involves running a Streams background process that progressively scans the online and archived redo logs (as required) to capture changes to data into events known as Log Change Records (LCR) and enqueue them into a queue. The capture process will make use of a dedicated queue for streams replication of type SYS.AnyData.

The Capture process will use rules defined using Oracle supplied PL/SQL based utilities to ensure that all DDL and DML operations on objects residing in the OPS$BRDB schema will be propagated from the Branch Database to the Branch Support System. Some DDL operations such as ALTER TABLE DROP PARTITION and CREATE OR REPLACE PACKAGE must not applied on the Branch Support System.

### 5.8.2 Propagate Changes to Branch Support System

The Oracle Streams stage and propagate background process will reside within the Branch Database and will propagate all LCRs events that are present in a named dedicated queue of type SYS.AnyData to a destination queue of the same type in the BRSS database.

The propagation can propagate or discard events based on any rules that may be defined. Examples of such rules include propagation of events belonging to selective schema objects, propagation of events based on whether they relate to DDL or DML operations etc. No rules will be defined at the propagation stage for the data replication from the Branch Database.

## 5.9 Branch Standby Database

Please refer to Section 8 for details on the Branch Standby Solution.

POL00115409



## 5.10 Post Hydra changes

This section lists the changes required to the Branch Database as a part of the post hydra phase.

Alongside each change, the document lists whether the change is mandatory or optional and wherever relevant, the implications of not making the change.

### 5.10.1 Schema Level Changes

The removal of any object specified in this section must be accompanied by the removal of its public / private synonyms.

**Tables to be removed (all Optional)**

BRDB_HYDRA_CT_TOTALS
BRDB_HYDRA_CT_MODE_TOTALS
BRDB_HYDRA_MIGRATION_SCHED
BRDB_HYDRA_EXCEPTIONS
BRDB_HYDRA_TPS_FILTER
BRDB_HYDRA_TPS_FILTER_ERRORS
TPS_HYDRA_INDAY_DATA
TPS_HYDRA_RECON_TOTALS
TMP_HYDRA_INDAY_XML
TMP_HYDRA_INDAY_XML_BARCODES
TMP_HYDRA_INDAY_XML_BDC
TMP_HYDRA_INDAY_XML_CUTOFF
TMP_HYDRA_INDAY_XML_CUTOFF_TOT
TMP_HYDRA_INDAY_XML_EOSPROMPTS
TMP_HYDRA_INDAY_XML_EPOSS
TMP_HYDRA_INDAY_XML_EPOSSCAP
TMP_HYDRA_INDAY_XML_HWM
TMP_HYDRA_INDAY_XML_LFS_IN
TMP_HYDRA_INDAY_XML_LFS_OUT
TMP_HYDRA_INDAY_XML_PINPAD
TMP_HYDRA_INDAY_XML_REPORTS
TMP_HYDRA_INDAY_XML_ROLLOVER
TMP_HYDRA_INDAY_XML_SCAN
TMP_HYDRA_INDAY_XML_STOCK
TMP_HYDRA_INDAY_XML_TC
TMP_HYDRA_INDAY_XML_USER_ROLES
TMP_HYDRA_INDAY_XML_USERS
TMP_HYDRA_RECON_XML
TMP_HYDRA_RECON_XML_PS_P
TMP_HYDRA_RECON_XML_PS_P_MD
BRDB_MIG_REPORT_DATA
BRDB_MIG_REPORT_DIFF

**View to be removed (optional)**

OMDBUSER.MV_TRADING_POSITION_DIFF

POL00115409



**Sequences to be removed (optional)**

BRDB_HYDRA_EXCP_SEQ
TPS_HYDRA_INDAY_SEQ
TMP_HYDRA_RECON_SEQ

**PL/SQL Objects to be removed (all Optional)**

All PL/SQL objects that correspond to the following feeds should be removed:

| | |
|---|---|
| BRDB_EPOSS_TXN_FROM_TPS: | Transfer EPOS transactions from TPS to BRDB (Hydra) |
| BRDB_APS_TXN_FROM_TPS: | Transfer AP transactions from TPS to BRDB (Hydra) |
| BRDB_BDC_TXN_FROM_TPS: | Transfer Bureau transactions from TPS to BRDB (Hydra) |
| BRDB_NWB_TXN_FROM_TPS: | Transfer Banking & ETU transactions from TPS to BRDB (Hydra) |
| BRDB_DCS_TXN_FROM_TPS: | Transfer Debit Card transactions from TPS to BRDB (Hydra) |
| BRDB_RECON_XML_FROM_TPS: | Transfer ongoing Reconciliation blob from TPS to BRDB (Hydra) |
| BRDB_INDAY_XML_FROM_TPS: | Transfer In-day Migration blob from TPS to BRDB (Hydra) |

The following packages are implicated, along with associated synonyms

Pkg_brdb_hydra_inday
Pkg_brdb_hydra_prep_recon
Pkg_brdb_hydra_recon
Pkg_brdb_inday_xml_from_tps
Pkg_brdb_recon_xml_from_tps
Pkg_brdb_nwb_txn_from_tps
Pkg_brdb_eposs_txn_from_tps
Pkg_brdb_dcs_txn_from_tps
Pkg_brdb_bdc_txn_from_tps
Pkg_brdb_aps_txn_from_tps



## 5.10.2 Metadata Changes

**Metadata to be removed (Mandatory)**

Metadata referring to all the tables listed in Section 0.1.1 needs to be removed from all of the following tables:

BRDB_ANALYZED_OBJECTS
BRDB_ARCHIVED_TABLES
BRDB_SUBPARTITION_RANGES
BRDB_PARTITIONED_TABLES
BRDB_PARTITIONED_INDEXES
BRDB_PARTITION_CREATES
BRDB_PARTITION_STATUS_HISTORY
BRDB_TABLE_PARTITIONS
BRDB_TABLE_GROUPS
BRDB_SQL_HINTS

**Metadata to be removed (Optional)**

Metadata referring to the following interfaces to be removed from BRDB_HOST_INTERFACE_FEEDS along with corresponding child rows from BRDB_HOST_INTERFACE_FEED_EXCP

BRDB_EPOSS_TXN_FROM_TPS

BRDB_APS_TXN_FROM_TPS

BRDB_BDC_TXN_FROM_TPS

BRDB_NWB_TXN_FROM_TPS

POL00115409



BRDB_DCS_TXN_FROM_TPS

BRDB_RECON_XML_FROM_TPS

BRDB_INDAY_XML_FROM_TPS

Metadata referring to process_name like BRDBX030%, BRDBX033% and BRDBX034 needs to be removed from BRDB_PROCESSES table. Data from BRDB_PROCESS_CONTROL and BRDB_PROCESS_AUDIT for the deleted BRDB_PROCESSES rows.

Rows from brdb_system_parameters for the following parameter_name

DEBUG_LEVEL_FOR_BRDBX030
DEBUG_LEVEL_FOR_BRDBX033
DEBUG_LEVEL_FOR_BRDBX034
BRDBX034_LOOK_AHEAD
BRDB_HYDRA_PREP_RECON_DEBUG_LEVEL
BRDB_HYDRA_FILTER
BRDB_HYDRA_RECON_DEBUG_LEVEL
BRDB_HYDRA_INDAY_DEBUG_LEVEL
BRDB_HYDRA_INDAY_BATCH_SIZE
BRDB_HYDRA_INDAY_CONTROL_VERSION
BRDB_APS_TXN_FROM_TPS_DEBUG_LEVEL
BRDB_APS_TXN_FROM_TPS_BATCH_SIZE
BRDB_APS_TXN_FROM_TPS_MAX_DATA_ERRORS
BRDB_BDC_TXN_FROM_TPS_DEBUG_LEVEL
BRDB_BDC_TXN_FROM_TPS_BATCH_SIZE
BRDB_BDC_TXN_FROM_TPS_MAX_DATA_ERRORS
BRDB_DCS_TXN_FROM_TPS_DEBUG_LEVEL
BRDB_DCS_TXN_FROM_TPS_BATCH_SIZE
BRDB_DCS_TXN_FROM_TPS_MAX_DATA_ERRORS
BRDB_EPOSS_TXN_FROM_TPS_DEBUG_LEVEL
BRDB_EPOSS_TXN_FROM_TPS_BATCH_SIZE
BRDB_EPOSS_TXN_FROM_TPS_MAX_DATA_ERRORS
BRDB_INDAY_XML_FROM_TPS_DEBUG_LEVEL
BRDB_INDAY_XML_FROM_TPS_BATCH_SIZE
BRDB_INDAY_XML_FROM_TPS_MAX_DATA_ERRORS
BRDB_NWB_TXN_FROM_TPS_DEBUG_LEVEL
BRDB_NWB_TXN_FROM_TPS_BATCH_SIZE
BRDB_NWB_TXN_FROM_TPS_MAX_DATA_ERRORS
BRDB_RECON_XML_FROM_TPS_DEBUG_LEVEL
BRDB_RECON_XML_FROM_TPS_BATCH_SIZE
BRDB_RECON_XML_FROM_TPS_MAX_DATA_ERRORS



### 5.1.3 Process/Module Changes

**Modules to be decommissioned (optional):**

| | |
|---|---|
| $BRDB_SH/BRDBX030.sh | Hydra specific XML Data Processor |
| $BRDB_SH/BRDBX033.sh | Hydra specific XML Data Processor |
| $BRDB_SH/BRDBX034.sh | Hydra specific XML Data Processor |

### 5.1.4 Scheduling Changes

All scheduling changes are discussed in the Scheduling Specification [R12].

POL00115409



# 6 Building Branch Database

## 6.1 Oracle Installation & Configuration

This section briefly describes the software components and installation guidelines for the Branch database (BRDB) on Linux.

The BRDB-Host platform will contain the following components:

- Oracle Clusterware version 10.2 (suitably stable patchset)
- OCFS2 version 1.2 (suitably stable patchset)
- Oracle Enterprise Edition 10.2 (suitably stable patchset) with partitioning & RAC option

Installation guidelines for the above-mentioned software components are as follows:

| Topic | Description |
|---|---|
| **RHEL** | ➢ The additional RPM packages required for an Oracle 10gR2 RAC cluster should be installed in addition to the standard Linux build from [R10]. These packages add development tools such as C/C++ compilers and libraries, glib and zlib libraries, libaio and openmotif.<br>➢ The hang-check timer should be enabled as per Oracle's recommendations.<br>➢ Rsh, rlogin and rexec must be enabled across all four of the BRDB-Host nodes.<br>➢ Kernel parameters such as minimum & maximum shared memory, semaphores and maximum file handles should be adequately set to allow a single Oracle database instance with a fixed size of 4GB and a variable (on-demand) requirement of another 2 GB plus a fixed size of 1GB for a smaller ASM instance to operate on each node.<br>➢ Ensure that the date/time are synchronized across all nodes. |
| **Network** | ➢ Ensure that the public IP addresses, VIP (virtual IP addresses) and the loop-back for the cluster interconnects are uniquely defined for the bladeFrame. |
| **Oracle Clusterware** | ➢ The Oracle clusterware VIPs should use the BladeFrame's internal back plane for interconnect communication. |
| **OCFS2** | ➢ Ensure that the OCFS2 console and tools packages are loaded along with their associated libraries.<br>➢ OCFS2 needs to start-up at boot-up time.<br>➢ Three partitions need to be created and managed by OCFS2. The first two are physical partitions meant for use by the OCR & voting disks. The third is an OCFS2 file-system formatted partition, which will be used for sharing the SPFILE across the nodes. Ensure that these partitions are mounted on all four nodes. Details on names and sizes of OCFS2 partitions have been included in the BRDB sizing & volume spreadsheet ([R11]). |
| **ASM** | ➢ ASM-lib version 2 or higher will be used as the storage management interface between Oracle and the disk storage. ASMlib provides a set of libraries, which allow for efficient access to shared raw devices used by ASM disk groups.<br>➢ The disks used to create ASM Disk Groups will be "oracleasm" partitioned. This allows for greater manageability and is Oracle's recommended approach and will be used for the Branch Database.<br>➢ Ensure that the ASM disks are configured to start on boot-up. |
| **Oracle Enterprise Edition RDBMS** | ➢ The Oracle software needs to be installed in Real Applications Cluster mode to allow the RAC-specific components to be installed.<br>➢ The starter database should not be created. |
| **ODBC** | ➢ Install the DataDirect ODBC driver version 5.3 on each Branch Database node. |

POL00115409



| | |
|---|---|
| **Driver** | |

**Table 2 – Guidelines for Installing & Configuring Oracle**

Detailed discussion of the installation e.g. order of software install and configuration of Oracle software components is beyond the scope of this document. PI refer to the Linux Host design document [R10] for the complete list of software components to be installed and the order of install. For installation details, refer to [R9].

## 6.2 Storage Management

### 6.2.1 Use of ASM

Oracle ASM will be used for managing the storage for the following database components:

- Control Files
- Online Redo-logs
- Persistent data-file tablespaces
- Temporary data-file tablespaces
- Archived Redo-Logs (in Flash Recovery area and NAS)
- Backups (in Backup Recovery Area)

There will be an instance of ASM created for each node of the branch database. Each instance will be named +ASM***n*** where ***n*** is the instance id and ranges from 1 to 4.



### 6.2.2 Creating the ASM instance

Common Linux platforms (four in total) will be used to host the BRDB database. Oracle RAC ASM instances (four in total) will be created on the BRDB-Host nodes to support the Branch database instances. Because BRDB database runs as four Oracle instances with each running on a different node, the ASM instance would also run on all four nodes.

The name of the instance will be "+ASM***n***" where ***n*** is the instance id and ranges from 1 to 4.

**BRDB Host – 1...4**

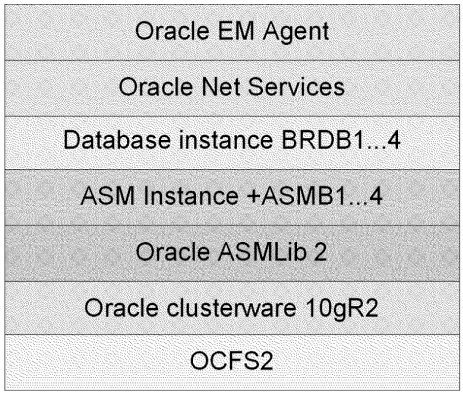



**Figure –** 12 **Oracle components on each BRDB-Host node**

POL00115409



The initialization parameters file for the ASM instances will be stored within a single SPFILE. This allows for maximum flexibility in maintaining the cluster. Each node will have a local PFILE (per instance) at the default location for initialization files (Oracle-Home/dbs) that simply points to the 'shared' SPFILE for the database. The 'shared' SPFILE can be shared across nodes using OCFS2.

### 6.2.3 Disk Groups

A number of very high read/write activity areas have been identified in the Branch database storage. In order to manage storage efficiently, heavily loaded database components/objects have been divided into five groups as shown below.

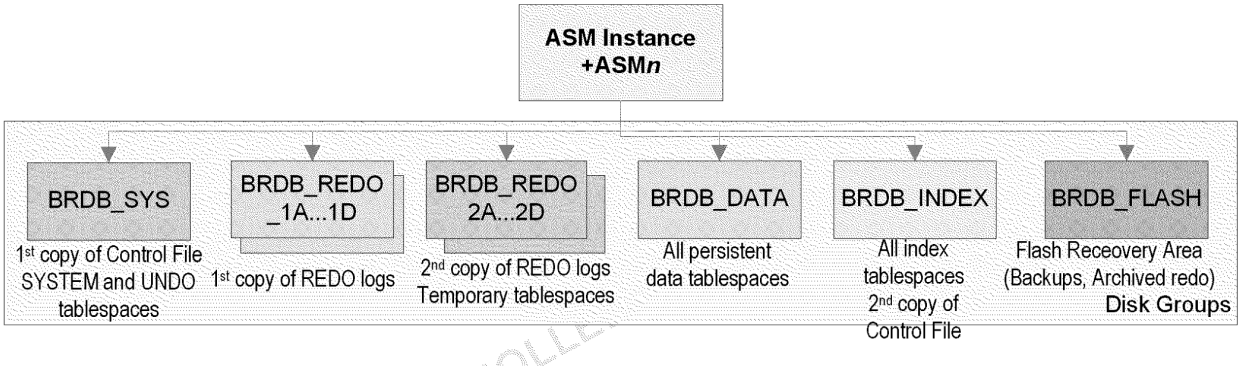



**Figure – 13 ASM*n* Instance Disk Group Layout**

The ASM disk groups to be created for the Branch Database are:

| Disk Group Name | Usage on a scale of 1 to 5 (1 low – 5 very high) | Description |
|---|---|---|
| BRDB_SYS | 3 | Used for the control file, system, sysaux and undo tablespaces |
| BRDB_REDO1A…1D | 5 | Used for maintaining the online redo-logs |
| BRDB_REDO_2A…2D | 5 | Used for temp tablespaces and for maintaining the local copies online redo logs. |
| BRDB_DATA | 5 | Used for storing the data |
| BRDB_INDEX | 3 | Used for storing the indexes and control file. |
| BRDB_FLASH | 3 | Used for the flash recovery area and archived redo-logs |
| BRDB_BRA_Pn/Sn (n-1,2 and 3) | 1 | Used for backups (not shown in the table above) |
| BRDB_STANDBY_REDO | 5 | Used for the Branch Standby redo logs (not shown in above table) |

**Table 3 – ASM Disk Group Usage Notes**

## 6.3 Database Components

### 6.3.1 Cluster Configuration Files

An Oracle10g release2 Real Application Cluster requires two cluster configuration files that must be created and maintained locally. Note that the installation of clusterware software and creation of the following files is covered in [R9].

**OCR:** The Oracle Cluster Registry stores cluster configuration details of all clustered Oracle

POL00115409



databases on the machine (node). Since the nodes hosting the Branch database will be used exclusively, the OCR will contain information about the Branch database cluster only.

The OCR is created when Oracle clusterware is installed but will be populated post database creation by issuing a number of "srvctl" commands.

The OCR is used by clusterware to start / stop databases, instances, listeners etc when the clusterware daemon starts up or stops respectively.

**Voting Disk:** The voting disk is used as means of synchronization and communication between RAC instances when one or more instances fail / start up.

Both files require a clustered file system or a cluster-aware logical volume manager to maintain them, as the files need to be accessible to all four nodes. Oracle OCFS will be used for storing these files.

Both of the files are vital to the functioning of the BRDB RAC cluster and need to be backed up. The backup approach is discussed in Section 12.5.

### 6.3.2 ASM Disk Groups

Already covered in Section 6.2.3.



### 6.3.3 Initialization Parameters

The initialization parameters file for both the ASM and the Branch Database instances will be stored as SPFILEs. This allows for maximum flexibility in maintaining the cluster. Each node will have a local PFILE (per instance) that simply points to the 'shared' SPFILE for the database. The 'shared' SPFILE can be shared across nodes using OCFS.

### 6.3.4 Character set

The Branch Database will use the WE8ISO8859P15 character set as per Oracle's recommendations on 8-bit character sets for databases that store data, which do not use windows encoding. The NLS Character set will be AL16UTF16.

[

### 6.3.5 Control Files

There will be three copies of the control file.

The first copy of the control file will be in the ASM disk group SYSTEM and the second in the ASM disk group BRDB_INDEX. The third copy of the control file will reside outside of ASM in the OCFS2 managed space.

### 6.3.6 Redo Logs

Four online redo log groups will be used per database node (or per thread). Size of the redo logs will be set to 400M[9] each.

Each online redo-log group will contain two members. The first member of every group will be created on synchronously replicated disks. The second member will reside on non-replicated fast disks.

[9] The size is chosen to allow the redo-logs to fill up every 1-1.5 minute at peak times. The final size may vary by 100MB on either side depending on the average size of each customer session message.

POL00115409



All online redo log files with reside within ASM. The first member of each redo-log group will be in the ASM disk group REDO_1A to REDO_2D and the second in the disk group REDO_2A to REDO_2D..

### 6.3.7 Flash Recovery Area

Flash recovery area is a location on disk where the database can create and manage a variety of backup and recovery-related files. As a minimum, BRDB will use the flash recovery area to store all the archived redo log files before they have been written to tape.

Flash recovery area can be enabled by setting the following parameter values in the spfile:

**DB_RECOVERY_FILE_DEST_SIZE** *(which specifies the disk quota, or maximum space to use for flash recovery area files for this database)* – This should be set to the same size as the archived redo log storage from the BRDB Sizing [R11].

**DB_RECOVERY_FILE_DEST** *(which specifies the location of the flash recovery area)* – This should point to the ASM disk group named FLASH.

Oracle's Flashback Query feature should not be enabled.



### 6.3.8 Archived Redo Logs

Two copies of the Archived redo log file should be created.

The first copy will reside inside the Flash Recovery Area and will be specified as mandatory.

The second copy will reside outside ASM on NAS storage and will be optional.

Refer to the BRDB Sizing [R11] for details on archived redo log sizes and locations.

### 6.3.9 Tablespace Definitions

The following tablespaces are in BRDB Application Representation in Designer10g. The implementation details are in BRDB Sizing [R11]. For performance considerations, the fact tables containing outlet-specific data will have their partitions distributed across multiple (128) tablespaces.

All tablespaces including SYSTEM will be 'locally' managed. All temporary tablespaces will be created using Oracle temporary files.

No tablespace should have its data-file set to "auto-extend" as is the standard database design practice. The auto-extend feature usually hides flaws in the system and usually lead to problems in the medium or long run that are difficult to overcome.

All tablespaces will reside within ASM. The disk group to be used is highlighted in the BRDB sizing spreadsheet.

- BRDB_CONTROL_DATA
- BRDB_CONTROL_INDEX
- BRDB_COMMON_DATA
- BRDB_COMMON_INDEX
- BRDB_COUNTER_REF_DATA
- BRDB_COUNTER_REF_INDEX
- BRDB_COUNTER_MISC_DATA
- BRDB_COUNTER_MISC_INDEX

POL00115409



- BRDB_FH_PART_*nnn*_DATA *where **nnn** ranges from 000 to 127*
- BRDB_FH_PART_*nnn*_INDEX *where **nnn** ranges from 000 to 127*
- BRDB_META_DATA
- BRDB_META_INDEX
- BRDB_REF_DATA
- BRDB_REF_INDEX
- BRDB_REPORT_DATA
- BRDB_REPORT_INDEX
- BRDB_TEMP*ppp* *where **ppp** ranges from 1 to 4*
- BRDB_UNDO*ppp* *where **ppp** ranges from 1 to 4*
- BRDB_SSC_DATA
- BRDB_STREAMS_DATA
- BRDB_TWS_DATA
- BRDB_WORKING
- BRDB_XDB_DATA
- SYSTEM
- USERS
- TOOLS



### 6.3.10 CASE Tool Schema Definitions

ERwin is being used a modelling tool for Branch Database. PDF files containing the schema definition can be found in [R36].

### 6.3.11 Table, Index and Meta data Definitions

Note that all table definitions detailed in this document are derived from the BRDB Application Representation in ERwin. ERwin is the primary source of these definitions therefore should be checked prior to using the definitions listed in this section.

For details of some Branch Database schema objects, please refer to the following appendices:

Appendix C – User, Role Sequence Definitions

## 6.4 BRDB Database Build

A set of database build scripts will be supplied as a part of the Branch database application delivery.

### 6.4.1 Building the Branch Database and Tablespaces

A set of build scripts will be delivered to create the branch database and required tablespaces.

The details of database build scripts will be present in related Low Level Design.

POL00115409



### 6.4.2 Security Measures for access control

On completion of database creation, the following security measures should be undertaken to prevent unauthorised access to the Branch Database. This is in accordance with the HADDIS standards ([R7]) and the HNG-X Security Architecture ([R23]).

Refer to Section 12.1.1 for some of the security changes required.

### 6.4.3 Schema Build

The definition of all branch database logical storage objects i.e. tables, indexes etc can be obtained from the ERwin data modeller repository. All objects have been defined as owned by the user OPS$BRDB.

A number of placeholders have been used in the schema definition in ERwin:

**#FH#** – The Fad-Hash value. All associated lines need to be replicated 128 times and the number #FH# replaced with a number ranging from 000 to 127 (along with the leading zeros).

**#AMP1#** – The input command-line parameter #1. This should be replaced with "&1" which will be later substituted in the build scripts.

**#AMP2#** – The input command-line parameter #2. This should be replaced with "&2" which will be later substituted in the build scripts.

**#UTYPE#** – Indicates if the user is live or training. Possible values are "LV" or "TR". For building the Live schema, use the value of "LV".

A set of build scripts will be delivered to create these two sets of the Branch Database application schema.

The details of schema build scripts will be present in related Low Level Design.



#### 6.4.3.1 Training Schema Build

The training schema will co-exist with the BRDB Live schema and all training schema objects will be owned by the schema owner OPS$BRDBTR.

When building the training schema, the placeholder "#UTYPE#" used in ERwin should be substituted with the value of "TR".

Refer to Section 5.6 for differences between the "live" and "training" schemas. Also refer to Section 18.1 for the list of tables that must be "shared" (read-only) by the "live" schema with the training schema.

## 6.5 Network Configuration

### 6.5.1 Services

The Branch Database will have the following services presented to the consumer applications (BAL, Batch Applications, Support, Monitoring etc). Services will be registered in the Cluster Registry using the "srvctl" utility.

| Service Name | Instance/s Name | Comments |
|---|---|---|
| BRDB1 | BRDB1 | Service will be available on the specific node and will be used to connect to specific instance. Failure to connect to that instance should result in an Oracle Error.<br>TAF will not be configured for this service. |
| BRDB2 | BRDB2 | |
| BRDB3 | BRDB3 | |
| BRDB4 | BRDB4 | |
| BRDB | BRDB1, BRDB2, BRDB3, BRDB4 | All four Branch Database instances are available for this service. |

POL00115409



| | | |
|---|---|---|
| | | TAF will not be configured for this service. |

### 6.5.2 Listener

A listener with the name of LISTENER_*hostname* will be configured to listen on Port #1529 for TCP/IP requests. The actual listener names will be **LISTENER_LPRPBDB00{1…4}**. The listener configuration will not refer to any Branch Database services. Instead the services will register themselves with the listener as per Oracle's recommendations for version 10gR2 instances. Refer to Section 22.1.2 for further details on setting any Net services specific initialisation parameters that facilitate this.

### 6.5.3 Local Naming

Local naming (TNSNAMES) will be configured for each of the services listed in 6.5.1.

In addition the following aliases will be configured in the TNSNAMES configuration file:

| Alias Name | Instance / Service Name | Comments |
|---|---|---|
| BRDB | BRDB | Load balance feature will be used. TNS addresses for all four Branch Database nodes will be specified. |
| BRSS1 | BRSS1 | Connect to the BRSS Instance |
| DRS | DRS | Connect to the DRS Instance |
| TES | TES | Connect to the TES Instance |
| APOP1 | APOP1 | Connect to the APOP1 Instance. |
| NPS1 | NPSSERVICE1_2 | Connect to the NPS1 Instance. No feature of TAF will be used. |
| NPS2 | NPSSERVICE2_1 | Connect to the NPS2 Instance. No feature of TAF will be used. |
| RDDS | RDDS | Connect to the RDDS Instance |
| RDMC | RDMC | Connect to the RDMC Instance |
| TPS | TPS | Connect to the TPS Instance |
| APS | APS | Connect to the APS Instance |
| LFS | LFS | Connect to the LFS Instance |



## 6.6 Replication to the SSC Support (History) Database

This section lists some of the changes required to the Branch Database to enable Streams replication to BRSS.

### 6.6.1 Initialization Parameters

The number of initialisation parameters need to be set to enable Streams to Capture and Propagate data to BRSS. The minimum values for the following parameters should be as follows:

JOB_QUEUE_PROCESSES: 500

STREAMS_POOL_SIZE: 100MB

Refer to Appendix E – Suggested Oracle Initialisation Parameters for the full list of suggested Oracle Initialisation Parameters.

### 6.6.2 Streams Administrator

The Oracle user STRADMIN will be used to administer Oracle Streams.

POL00115409



### 6.6.3 Supplementary Logging

To allow the log-mining feature of Oracle Streams to work, minimal supplementary logging must be enabled at database level. This can be done using the “Alter Database” SQL command.

### 6.6.4 Streams Capture

A new dedicated streams queue should be set up for the data capture. The streams queue table should be called BRDB_BRSS_REPL_QUEUE_TABLE and owned by the Streams Administrator.

The capture mechanism should make use of the archived redo-logs for capturing log changes in BRDB. A rule-set (handle for specifying capture rules) should be created and used to specify that the DML changes should be captured for all tables owned by the schema owner OPS$BRDB.

It is recommended that a degree of parallelism value of 4 should be used for capturing information however this parameter should be validated during performance testing.

### 6.6.5 Streams Propagation

The propagation should be set up to make use of the rule-set defined earlier. Using the same rule-set for capture and propagation allows for ease of changing rules by amending the single rule-set.

The source queue created earlier should be associated with a destination queue created in BRSS.

As required, source schema to be replicated should be instantiated.

Streams propagation should be disabled after creation and should be later enabled only after the Streams apply environment has been set up and enabled in the Branch Support System.



## 6.7 Setting up Branch Standby Database

Branch Standby Database is discussed in detail in Section 8.

## 6.8 Interfacing with non-Oracle Database Systems

The Branch Database needs to interface with non-Oracle based Estate Management system (EMDB). EMDB is Microsoft SQL Server based In this case EMDB will push data in to an interface table. This is being done to save on licensing costs for Oracle Heterogeneous Agent.

### 6.8.1 Listener and Local Names (TNSNAMES)

The existing listener LISTENER_{node_name} will be modified to include the following SID_DESC in the SID_LIST_*LISTENER_{node_name}* section for each node:

```
(SID_DESC =
        (SID_NAME = em)
        (ORACLE_HOME = /oracle/10g)
        (PROGRAM = hsodbc)
)
```

A new local names configuration will be added on each node of the Branch Database:

```
em =
        (DESCRIPTION =
                (ADDRESS = (PROTOCOL = TCP)(HOST = <BDB Node-name>)(PORT = 1521))
                (CONNECT_DATA =
                        (SID = em)
```

POL00115409



```
            )
            (HS = OK)
        )
```

POL00115409



# 7 BRDB Host Application

The BRDB host processes will written using PL/SQL and Pro*C. The PL/SQL compiler is built into the Oracle10g enterprise edition database. Pro*C pre-compiler and its libraries will be installed as a part of Oracle 10g (Release 2) database server installation.

As a part of platform build, all BRDB Linux users, their profiles, environment variables and the required directories will be created on the rig.

[The details of the Linux Users, Profiles and Environment Variables will be available in the delivery handover note.]

## 7.1 Application Development

### 7.1.1 Code reuse

In order to reduce the development costs and to maximise the benefits of pre-tested algorithms and code, it is recommended that code from existing host systems should be reused wherever possible. This is regardless of whether Pro*C or PL/SQL is used as the programming language.



The prime candidates for code re-use are:

- Process Control and Exception Handling common code
- Start-of-Day and Partition Management
- Audit, archiving and table housekeeping
- Optimizer Statistics
- File Housekeeping
- End-of-Day

### 7.1.2 Process Logging

All host processes will log significant steps on the standard output. Examples of significant steps are process started/finished, partition creation SQL executed etc. All messages displayed must be prefixed by a timestamp.

### 7.1.3 Debug facility

In any production environment a number of factors can affect the performance/execution of long running processes. It is usually difficult to determine the cause of failure without taking a data snapshot to a Development environment and re-creating the error scenario. It is particularly difficult to recreate performance-related issues, as these are usually rig-specific.

With the view of easing the faultfinding process, all modules will include a number of conditional statements that allow relevant debugging information to be displayed on standard output:

The output of debug information can be controlled by relevant System Parameters. The system parameter name should be “DEBUG_LEVEL_FOR_**xxx**” where **xxx** is the module name e.g. “BRDBC350”.

The following debug flag values should be used. Note that these are common guidelines and should not stop individual modules from displaying prominent information where necessary:

POL00115409



0 – No additional debug information displayed. The process-logging requirement described earlier could be merged with debugging and displayed for level – 0.

1 – First level of debugging. Message displayed for each milestone e.g. the action of fetching records from a table, opening / closing files etc. Messages must be prefixed by a procedure/function name identifier.

2 – Second level of debugging. Display all messages for previous levels plus messages for entry and exits from control structures. Messages must be prefixed by a procedure/function name identifier.

3 – Third level of debugging. Display all messages for previous levels plus prominent data items processed. . Messages must be prefixed by a procedure/function name identifier.

## 7.1.4 Process Control & Audit

### 7.1.4.1 Process Control Mechanism

Most BRDB host processes will be run as one or more stand-alone instances on the Host and will be part of a process schedule controlled via TWS. All such processes will have a built-in restart facility.

Instances that run once during a BRDB processing day will make use of the process control table as required by HADDIS [R7], to aid restarts. An entry will be made at the start of the process and the same will be updated to indicate completion before the process exits.

If an entry were already present for that instance for the BRDB processing day, this will indicate a restart and the instance would take actions accordingly.

If a process instance handles partitioned objects with each process being able to process multiple partitions, a facility will be provided for the process to log the start and end points of each partition. In case of a failure, this provides a facility to restart the process while skipping all partitions that had been successfully processed prior to the failure and starting from the partition being processed when the failure occurred. This provides the means to save on processing time for restarts after failures by going straight to the partition level and also provides a mechanism of breaking down the process into logical steps if so required.

Information about instances running more than once during a processing day will be obtained from the Process information table and will be used to increment the run number in process control for recording multiple executions. Checks will be made for previously failed runs for the instance and if found the same run number would be used by the instance for a restart if required.

### 7.1.4.2 Process Control Table

As described above, this table will store control information about all processes that are invoked as stand-alone instances. It will be non-partitioned with a primary key index. Entries will be stored in this table for the maximum duration of transactional data retention i.e. 60 days. For table definition refer to section Appendix A – Table and Index Definitions.

### 7.1.4.3 Process Control Routines

A set of common routines will be provided for the use of all BRDB application modules for performing operations on the Process Control table. These will be written in Pro*C for use of Pro*C modules and in PL/SQL for the use of database stored objects written in that language.

POL00115409



#### 7.1.4.4 Process Audit Mechanism

All BRDB processes will log an entry in the Process Audit Table as required by HADDIS [R7] at the start and end of the process instance to provide process audit information.

#### 7.1.4.5 Process Audit Table

As described above, this table will store audit information about all processes that are invoked as stand-alone instances. It will be non-partitioned with a unique system generated primary key. Entries will be stored in this table for the maximum duration of transactional data retention i.e. 60 days. For table definition refer to Appendix A – Table and Index Definitions.

#### 7.1.4.6 Process Audit Routines

A set of common routines will be provided for the use of all BRDB application modules for storing audit information in the Process Audit table. These will be written in Pro*C for use of Pro*C modules and in PL/SQL for the use of database stored objects written in that language.



### 7.1.5 Exception Handling

Exceptions will be grouped into two types, Data Exceptions and Operational Exceptions. Both types will be implemented using similar methods. Both types will be reported and hence resolved by different means.

All exceptions detected or generated will be logged by making a concurrent connection to the BRDB Oracle database in order to avoid affecting the processing. This will use the Oracle concurrent commit so as not to affect any transaction processing. An option will be provided to display the error details on standard or error output.

Data Exceptions will be reported using reports and remain on the reports until resolved by support at which point support will manually clear the exceptions.

Operational Exceptions will be reported using IBM Netcool®/OMNIbus™ software [local table, separate server, remote client] to the Operations desk and will remain visible until resolved by Operations.

#### 7.1.5.1 Data Exceptions

##### 7.1.5.1.1 Input Data Exceptions

Input data exceptions are exceptions that occur when the Branch Access Layer receives a request from the Counter that is either malformed (not well-formed XML structure) or fails validation e.g. missing mandatory attributes. These will be logged by the Branch Access Layer in the Input Exceptions Store.

If the BAL marks a request as ‘Bad’ or the primary key values cannot be derived, the request cannot be processed any further and will also be logged into Input Exceptions store.

These exceptions will be made accessible to support using an interface into the IBM Netcool®/OMNIbus ™ software.

##### 7.1.5.1.2 Host Interface Data Transfer Exceptions

Exceptions that occur while transferring messages to and from the existing host applications will be logged in Host Interface Feed Exceptions store (brdb_host_interface_feed_excp). These will include errors like Null value for mandatory columns or size larger than maximum allowed etc. The messages

POL00115409



will be logged in a single table in the Branch Database regardless of the direction of data flow across the interface.

These exceptions will be made accessible to support using an interface into the IBM Netcool®/OMNIbus ™ software.

#### 7.1.5.2 Operational Exceptions

Operational exceptions can occur during BRDB host processing due to a range of reasons, many of which may be outside control of the process or the BRDB system. When exceptions occur the relevant information needs to be recorded for fault analysis or correction. HADDIS [R7] requires that all operational exceptions be loaded into a table to be viewed and updated by support to indicate they have read and acted upon the exception. In this case the BRDB_OPERATIONAL_EXCEPTIONS table.

Operational Exception handling for BRDB will be modelled on existing host applications exception handling. Operational Exceptions can be divided into two categories:

- Anticipated operational exceptions raised by the application
- Unexpected operational exceptions raised by the application.

Operational Exceptions will be defined in a local reference data table using an eight-character identifier as defined by HADDIS [R7]. This table also contains the probable causes, action to be taken and exception severity.



##### 7.1.5.2.1 Application Anticipated Operational Exceptions

The mechanism will be written both in Pro*C and PL/SQL and callable from both languages. Any other programming language subsequently used in BRDB development for modules that cannot raise Operational Exceptions itself will have the mechanism implemented for it. Currently, only Pro*C and PL/SQL are to be used.

Information relevant to the type of Operational Exception such as Oracle or Linux generated will be passed. All such exceptions will be stored in a common Operational Exceptions table. An option will be provided to display the error details on standard or error output.

##### 7.1.5.2.2 Unexpected Operational Exceptions

PL/SQL provides in-built exception trapping mechanisms. These will be used to handle Operations Exceptions from models written in these languages.

For Pro*C, Operational Exceptions will be handled by a high-level exception handler. This handler will be modelled on the existing Host systems approach of ‘Try’, ‘Catch’ and ‘Finally’ macros, similar to that used in the C++ and Java programming languages.

The utility will be written in Pro*C and will provide ‘Try’, ‘Catch’ and ‘Finally’ macros for trapping calls made to a ‘Throw’ function. The information passed to the utility, that is the accessible diagnostic information, will be stored in a common Operational Exceptions table and will be identified by a system-generated sequence.

##### 7.1.5.2.3 Report / Alert

As with the other Host Systems at HNG-X, access will be provided to support via Netcool®/OMNIbus™ to view and respond to all Operational Exceptions as per HADDIS [R7].

One or more Netcool®/OMNIbus™ alerts will provide a conspicuous visual alarm when exceptions are raised. The same alarm will also indicate the number of outstanding exceptions.

POL00115409



A report written in Netcool®/OMNIbus™ scripting language will display a list of all exceptions that are outstanding or unresolved, along with their probable causes, actions required and priority. The report will be generated by support on 'as-required' basis.

All exceptions will need to be 'resolved'. A call will be raised and investigated by support in co-ordination with the support teams. A screen-based Netcool®/OMNIbus™ utility will be provided to mark exceptions as 'resolved'.

## 7.2 Host Processes

BRDB is mainly used by the Branch Access layer to store and process transactional, reporting, event and other forms of data as described earlier.

However, it also performs a number of activities that require host-based processes. Examples of such activities are being able to provide transactional and other feeds to various existing host systems and receive data from them, presenting an audit feed of the day's auditable messages received from the Counter to the Audit-System and aggregating the current day's reporting transactions.



### 7.2.1 Start of Day (Application BRDBC001)

The Start of Day process performs the startup activities for the Branch Database. The activities include incrementing the BRDB Host System Date, partition creation / deletion for all tables implemented as a date-based rolling window and any other startup tasks required by the Branch Database.

#### 7.2.1.1 Application Type

This is a PRO*C application that uses PL/SQL library functions developed as per HADDIS [R7] (along the lines of functions used by TES-Host).

#### 7.2.1.2 Inputs

No mandatory input parameters used.

The optional input parameters are:

**Table-Group**: Name of the table group to create partitions for

**Table-Name**: Name of the table within the Table-Group to create partitions for

**Partition-Date** (CCYYMMDD): The Partition-Date to create partitions for.

**System**-Date (CCYYMMDD): The System Date value to set in System Parameters.

#### 7.2.1.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

#### 7.2.1.4 Location

The module should reside in **$BRDB_PROC** directory on each Node.

POL00115409



#### 7.2.1.5 Scheduling

A single instance of the application runs every day on the Branch Database (Node-1) at a fixed time (typically after midnight) as defined in the Scheduling High Level Design [R12].

#### 7.2.1.6 Business Day

Business Day is read from "BRDB System Date" from table BRDB_SYSTEM_PARAMETERS and is incremented.

#### 7.2.1.7 Partitioned Tables

The partitioned tables whose partitions are created / dropped by the Start of Day process are defined in the partition management metadata tables in Appendix D – Metadata and Static Reference Data Definition.

#### 7.2.1.8 Processing

The process will create new partitions in the Audit Store, Settlement and some reporting tables 1 day[10] in advance. For example, if the process is run on day N (Oracle SYSDATE), it will create the partition for day N+1. This reduces the Branch Access Layer's dependency on the partition creation. Database access and usage will not be affected for at least a day if the partition creation for the day failed for any reason. Hence, the support will have enough time to resolve the problem before it becomes critical. Partition maintenance will be achieved through a generic set of partition management functions driven by partition management metadata as defined in HADDIS [R7].

The process will truncate and drop the partitions older than N (typically 3) calendar days for the tables listed in **Appendix D – Metadata and Static Reference Data Definition**. For example, if the process is run on day N (Oracle SYSDATE), it will truncate and drop the partitions older than 00:00:00 hours of the N-3rd day. Before deletion a check will be made to see if the partition to be dropped satisfies the criteria laid out in database column ***brdb_partitioned_tables.delete_check_criteria***, if specified.

If there is more than one eligible partition to be dropped, all will be dropped starting with the oldest. All create and drop operations will use the Metadata and common functions as defined in the HADDIS [R7]. While dropping partitions for more than a day, checks must be made to ensure that the number of remaining of partitions does not go below the (metadata driven) threshold, which is 3 calendar days for Audit Store, Settlement and Recovery tables and 60 calendar days for Reporting. This check is to prevent the housekeeping process inadvertently dropping valid partitions if the reference date (the retention period is applied to) is wrongly set to a future date.

Finally the values of the following system parameters (stored in the BRDB_SYSTEM_PARAMETERS table) will be updated:

BRDB HOST SYSTEM DATE – Date incremented by one day.

T&T Copy Complete flag – Value reset to 'N'

GREV Copy Complete flag – Value reset to 'N'

#### 7.2.1.9 Handling Failures and Rerun ability

In the event of an Instance failure, the failed job needs to be executed against another node of the Branch database. Details are available in Scheduling High Level Design [R12]. Other types of failures need to be handled as per the HADDIS guidelines.

[10] This, in effect, becomes 2 days as all create and drop operations will use midnight as the day boundary.

POL00115409



It is recommended that the Process Control functionality be used to control the Process for each Table-Group and table affected. Note that if the optional command-line parameters are passed in, Process-Control mechanism should not be used.

Since this process is expected to run a number of SQL DDL statements, care should be taken to ensure that the BRDB System Date is never re-incremented in case of a restart after failure. PI refer to the TES-Host Start of Day process for the additional functionality implemented to protect against such issues.

## 7.2.2 Message Journal Auditing (BRDBC002)

The message journal auditing process will generate the text files for a given day's auditable messages by reading records from the Message Journal table (BRDB_RX_MESSAGE_JOURNAL).

The data in the Message Journal will be divided among several audit files to:

- reduce the size of each audit file
- allow efficient use of the audit files later on

### 7.2.2.1 Application Type

This is a PRO*C application



### 7.2.2.2 Inputs

Two mandatory input parameters are passed on command-line:

- Business-Day is passed on command-line in CCYYMMDD format.
- The Branch Database Node Number (1…4)

A further optional parameter Fad-Hash can be passed on command-line. This is present to allow for regeneration of a subset of audit files if there is any reason for doing so e.g. output files got corrupted, file-system got full etc.

Note. The value of the BRDB_SYSTEM_PARAMETER for BRDB_BRDBC002_MAX_FILE_SIZE must be set to a value in the range 200 bytes to 4 Gigabytes. Setting a value below 200 will write a single audit record to each file and therefore will exceed the 999 files limit (detailed below).

### 7.2.2.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

### 7.2.2.4 Location

The module should reside in **$BRDB_PROC** directory on each Node.

### 7.2.2.5 Scheduling

The application runs every day in the form of four instances after 1 am local time.. Each instance will run on a separate node of the Branch Database.

POL00115409



### 7.2.2.6 Audit Source Tables

There is a single source table for all Auditable messages. The table name is BRDB_RX_MESSAGE_JOURNAL and all messages in it are meant to be audited.

### 7.2.2.7 Audit file record format

The data to be audited is present in the Oracle column BRDB_RX_MESSAGE_JOURNAL.JOURNAL_HEADER and BRDB_RX_MESSAGE_JOURNAL.JOURNAL_XML. The journal header column is of type varchar2 and the journal XML column is of type BLOB and optionally holds 'gzip' compressed data. The header column stores the 'http' header of the message sent by the Counter to the Branch Access Later and journal XML stores the payload.

Data needs to be sorted on Journal-Date, Fad-Hash, Branch-Accounting-Code, Node-Id and Journal-Sequence-Number when fetching from the table.

The file does not have any header or trailer records. There is no need for writing any begin or end tags around the file records. One record is written per message and comprises of 'http' header and payload separated by new line. The following additional headers are added to simplify the processing by Audit system.

- http-Header-Length: <nnn>
- Content-Length-Inflated: <nnn>

The payload will be inflated (uncompressed), where required, before it is written to the file. The entire file is then compressed using 'gzip'.

Sample uncompressed audit file is attached below.

AUDIT_20081218_039_001.aud



### 7.2.2.8 Audit-Store file generation

One or more files will be created for each Fad-Hash value resulting in minimum of 128 files. If there are no transactions for the Outlets belonging to a Fad-Hash, empty files will still be created. There is a further constraint that each audit file needs to be approximately 100MB) uncompressed. The limit size is held in the BRDB_SYSTEM_PARAMETERS value for BRDB_BRDBC002_MAX_FILE_SIZE. When the data being written to a file exceeds the limit, the file should be closed after completion of the record being written and a new file should be created with the file sequence number incremented by 1. Note that the file sequence number should always start with 1 for the first file for a Fad-Hash value. The program will return an error if more than 999 files are created for a single fad-hash.

As the data needs to be fetched in a sorted order, the most suitable index (if more than one) must be used for this purpose.

All the files for a Fad-Hash value (and there may be more than one file depending on size) will be made available to the Audit copy process once all the transactions for the Fad-Hash value have been successfully written to the audit files. The files need to be created in an interim location until then.

The audit store records need to be fetched from the correct Oracle instance (and Node) for a Fad-Hash value. Refer to Section 5.1 for details on the reasons and the mechanism for doing so. Fad-hashes will be fetched for the correct instance by using the view BRDB_FAD_HASH_CURRENT_INSTANCE.

As records are being written to the audit files, the process must optionally be able to monitor if the set of Journal-Sequence-Numbers for a node in a Branch is dense. The check should only be performed when

POL00115409



the value of mandatory System-Parameter 'JOURNAL_SEQ_DENSE_SET_CHECK_ENABLED' is "TRUE". When a missing journal entry is encountered, a message should be written on standard output along the lines of "…records between sequence numbers *M* and *N* are missing…". Once the list of auditable messages for a node is completed, an Operational exception should be raised to indicate the count of missing sequence numbers.

#### 7.2.2.9 Directories and Files

The final location of the Audit files created by this module is pointed to by the environment variable $BRDB_COUNTER_AUDIT_OUTPUT.

The actual directory locations will be covered in the Host delivery release notes.

The file names will be of the following format:

"AUDIT_" + Business Date in CCYYMMDD format + "_" + Three digit Fad-Hash '0' padded from left + "_" + Three digit File-Sequence-Number '0' padded from left + ".aud.gz"

#### 7.2.2.10 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun for the pending Fad-Hash values. If the failure reason is a Branch Database node/instance failure, recovery jobs will be run on the other nodes of the Branch Database for pending Fad-Hashes that belong to the failed node. Details are available in the Scheduling High Level Design [R12].

It is recommended that the Process Control functionality be used to control the Process at Fad-Hash level.



### 7.2.3 Heritage Host Systems Interface (BRDBX003)

The Legacy Host Interface application provides the process control and business logic required to implement any of the legacy host interfaces defined in 5.3.4 where the Branch Database owns the interface application logic.

#### 7.2.3.1 Application Type

The Interfaces will be implemented as PL/SQL stored packages/procedures/functions. The interface functionality may be handled by common PL/SQL objects as appropriate. There will be a single point for calling all host interfaces, which will be implemented in the form of an interface wrapper script developed as a Linux Shell script.

#### 7.2.3.2 Inputs

The interface wrapper will accept the following mandatory command-line input parameters:

- **Batch Interface Name:** This is the logical name of the Batch interface and is expected to correspond to a stored PL/SQL object that will be the access point to implement the interface.
- **Input Trading/Business Date:** This date is used to filter the records being passed across the batch interface. Format is CCYYMMDD.
- **Node/Instance Id:** This indicates the Branch Database Node that the batch interface job instance needs to execute against. Values range from 1…4.

In addition the interface wrapper will accept optional parameters that may be needed for specific interfaces.

POL00115409



All input parameters will be passed to the PL/SQL object that implements the interface.

#### 7.2.3.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures.

#### 7.2.3.4 Location

The wrapper script should reside in **$BRDB_SH** directory on each Node.

The interface will reside within the Branch Database.

#### 7.2.3.5 Scheduling

Multiple instances of the interface wrapper script run at different times during the day. With a few exceptions, each instance will run on a separate node of the Branch Database.

Refer to the Scheduling High Level Design [R12] for more details.



#### 7.2.3.6 Processing Details

The design principles behind the host interface and its logical implementation are discussed in Section 5.3.

The wrapper script validates the command-line input parameters and calls the relevant host systems interface PL/SQL object to pass on the parameters. The script traps / handles Node / Instance failures and returns appropriate return status back to the scheduler.

The interface PL/SQL object formulates the SQL statement required to implement each step of the host interface. The application logic required to implement various host interfaces is discussed in Section 5.3.4.

#### 7.2.3.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun from the point of failure. If the failure reason is a Branch Database node/instance failure, a number of recovery jobs will be fired on the other nodes/instances of the Branch Database. Details are available in the Scheduling High Level Design [R12].

The use of Process Control functionality is covered in the description of the Legacy Host Interfaces (see Section 5.3).

The level of Control / point of restart for an interface will depend on whether it has been implemented as a single instance or four instances.

Typically for interfaces that are implemented as a single instance, the restart point is entire interface run for the day. This covers all smaller interfaces where the amount of time lost in the event of a rerun is insignificant.

For interfaces implemented as four instances (one per database node), the restart point is at a Fad-Hash level. This covers all interfaces where finer control may be needed due to the large volumes of data being transferred.

There may be exceptions to this rule and these would be highlighted in Section 5.3.4.

POL00115409



## 7.2.4 Audit, Archive & Purge (Application BRDBC004)

The audit, archiving and purge process will run outside core business hours, typically after midnight. The objects covered by the audit function of this process are as per the HADDIS [R7] guidelines.

Note that the audit function of this process differs from the audit file generation functionality required for the counter generated auditable messages.

The process will:

- Generate an audit file from the data in the database access audit table (SYS.AUD$) where Record Insert Timestamp is between 00:00:00 and 23:59:59 for the previous calendar day.
- Audit the Process-audit information.
- Archive and delete Process Control and Audit information and any other tables as per the criteria defined by metadata.
- Purge data from non-partitioned tables as per rules defined in metadata.
- Mark data partitions as “ARCH” or “NARCH” depending on whether they have been archived or not, once the partitions have aged as per the retention policy in metadata. These will then be partition managed by BRDBC001.



### 7.2.4.1 Application Type

This is a PRO*C application

### 7.2.4.2 Inputs

- No parameters are accepted.

### 7.2.4.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

### 7.2.4.4 Location

The module should reside in **$BRDB_PROC** directory on each Node.

### 7.2.4.5 Scheduling

A single instance of the application runs every day on the Branch Database (Node-1) after the completion of the overnight bulk-copy jobs to existing host systems.

### 7.2.4.6 Audited / Archived Tables

The audit, archive and purge candidate tables are defined in BRDB_ARCHIVED_TABLES table in Appendix D – Metadata and Static Reference Data Definition.

POL00115409



### 7.2.4.7 Audit Record Formats

The audit data files will be text files containing the table name, date and time of archive, field delimiter used, table structure including the column names and type/length, and the variable length records. The file record formats will be as per the HADDIS [R7] guidelines.

The structure of each file will be as follows:

```
<map>
<table>BRDB_PROCESS_CONTROL
<date>12082003_011931
<trim>
<separator>|
<col>TRANSMISSION_DAY_NUMBER|N|NUMBER(3)
<col>TRANSMISSION_FILE_NUMBER|N|NUMBER(3)
<col>SUB_FILE_SEQUENCE_NUMBER|N|NUMBER(5)
<col>DATA_CENTRE|N|CHAR(1)
<col>ORG_UNIT_ID|N|NUMBER(10)
…
<col>TOTAL_BDC_TRANSACTIONS||NUMBER(6)
<end>
186|68|1|W|13403|17|N|N|N|05072003_233941|19|505038|05072003_000000|5|4|0|4|0||||||6||0|0
186|68|2|W|17924|17|N|N|N|05072003_233941|1|7261|05072003_000000|0|0|0|1|0||||||0||0|0
…
…
186|67|1|W|6864|9|N|N|N|05072003_233930|1|500470|05072003_000000|0|0|0|1|0||||||0||0|0
<trailer>
```

Audit files will be copied to a separate directory on the BRDB Host and then collected by the Audit Server on a daily basis. The details of the audit directory/file names etc. will be included in the low-level design of the module.

The audit, archive and deletion criteria are set in BRDB_ARCHIVED_TABLES table in Appendix D – Metadata and Static Reference Data Definition.

### 7.2.4.8 Archive File Format

The archive output files will be Oracle database dump files created on a per-table group basis.

### 7.2.4.9 Directories and Files

The location of the Audit and Archive files created by this module are pointed to by the environment variables $BRDB_HOST_AUDIT_OUTPUT and $BRDB_ARCHIVE_OUTPUT.

The actual directory locations will be covered in the Host delivery release notes.

The filenames will contain the Group Prefix, Table Prefix and timestamp of creation as per HADDIS [R7]. The file extension for Audit files will be

The file locations and names are as per the table below:

| File Type | File Location | File Name format | File Extension |
|---|---|---|---|
| Audit | As pointed to by environment variable $BRDB_HOST_AUDIT_OUTPUT | 3-character Group Prefix + 3-character Table Prefix + Business Date (in **CCYYMMDD** format) | ".aud" |
| Archive | As pointed to by environment variable $BRDB_ARCHIVE_OUTPUT | 3-character Group Prefix + 3-character Table Prefix + Business Date (in **CCYYMMDD** format) | ".dmp.Z" |

**Table 4 – Guidelines for Installing & Configuring Oracle**

POL00115409



#### 7.2.4.10 Handling Failures and Rerun ability

In the event of an Instance failure, the failed job needs to be executed against another node of the Branch database. Details are available in the Scheduling High Level Design [R12]. Other types of failures need to be handled as per the HADDIS guidelines [R7].

It is recommended that the Process Control functionality be used to sub-divide and control the Process for each type of action e.g. Audit, Archive and Purge.

### 7.2.5 Gathering Optimiser Statistics (BRDBX005)

On a daily basis, a process that gathers Optimiser statistics using Oracle utilities is run. It can either gather schema, table or system statistics. If called for table statistics it gets the details of the tables or table partitions to gather statistics on and any additional parameters to use from the table BRDB_ANALYZED_OBJECTS. The TWS scheduled job call to BRDBX005 will gather SCHEMA statistics.. The job call utilises parameter flags and there ParameterN below refers to the flag value, eg:

$BRDB_SH/BRDBX005.sh –i 1 –s SCHEMA



Additionally the Optimiser will also gather system level statistics and be able to update the data dictionary with the statistics gathered.

#### 7.2.5.1 Application Type

This is a Linux shell script based application that invokes PL/SQL package based functions.

#### 7.2.5.2 Inputs

Three input parameters are passed on command-line:

- Parameter1 – (Mandatory) The job Instance Name. This is a logical value used to allow multiple runs of the same Object Group during a processing day.
- Parameter2 – (Mandatory) Type of Statistics: Possible values are “SCHEMA” or “TABLE” or “SYSTEM”.
- Parameter3 – (Mandatory for –s TABLE or –s SYSTEM) varies based on whether Table or System statistics are gathered.
    - For Table statistics, Parameter – 3 consists of one or more Object-Groups that correspond to the database column BRDB_ANALYZED_OBJECTS.OBJECT_GROUP. Several object groups can be supplied in a comma separated list.
    - For System statistics, Parameter – 3 is either “GATHER” or “IMPORT”.

#### 7.2.5.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

#### 7.2.5.4 Location

The Optimizer script should reside in **$BRDB_SH** directory on each Node.

POL00115409



The PL/SQL packages will reside within the Branch Database.

#### 7.2.5.5 Scheduling

The application runs every day in the form of one or more jobs. Each instance can run on a separate node of the Branch Database. However they are usually scheduled as a single job to run on Node2. Details are available in the Scheduling High Level Design [R12].

#### 7.2.5.6 Target Tables

If invoked for TABLE statistics the complete list of objects to be analyzed i.e. entries in BRDB_ANALYZED_OBJECTS can be found in Appendix D – Metadata and Static Reference Data Definition. This is currently not invoked in the TWS schedule BRDB_ORA_STATS.

#### 7.2.5.7 Processing Details

The Oracle statistics gatherer uses the Oracle built-in DBMS_STATS package to gather statistics on Objects such as tables and indexes. Up to date statistics on large tables/indexes allow Oracle's Cost Based Optimizer to accurately determine the optimal execution path for DML statements involving those objects.

The Statistics gatherer should be able to gather system level statistics using the DBMS_STATS package. If the input parameter – 3 value is "GATHER", the statistics should be gathered and stored in the BRDB_OBJECT_STATS_ARC table using a suitable alias. For parameter value of "IMPORT", the gathered statistics should be imported into the data dictionary. Input parameter 3 is not used when input parameter 2 is 'SCHEMA'.

The System level statistics will not be gathered on a daily basis. Instead jobs will be manually submitted as and when appropriate by support. Refer to the BRDB Support Guide for details.

#### 7.2.5.8 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun for all pending/partially processed object groups. If the failure reason is a Branch Database node/instance failure, the job will be rerun on another node/instance. Details are available in the Scheduling High Level Design [R12].

It is recommended that the Process Control functionality be used to control the Process at Instance-Name and Object-Group level.

### 7.2.6 File Housekeeping (BRDBX006)

The file housekeeping process deletes 'old' file-system files using standard Linux utilities. The file template of files to be deleted is defined in the metadata table BRDB_FILES_TO_HOUSEKEEP.

#### 7.2.6.1 Application Type

This is a Linux shell script based application.

#### 7.2.6.2 Inputs

None

POL00115409



#### 7.2.6.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

#### 7.2.6.4 Location

The script should reside in **$BRDB_SH** directory on each Node.

#### 7.2.6.5 Scheduling

The application runs every day in the form of a single instance after the completion of the jobs BRDBC002 – BRDBC005. The instance will run on the first node of the Branch Database. Refer to the Scheduling High Level Design [R12] for more details.

#### 7.2.6.6 Target Directories

The directories containing these files are detailed in the Scheduling High Level Design [R12].

The main target directories for file housekeeping are:

- /app/brdb/trans/audit
- /app/brdb/trans/support/archive



#### 7.2.6.7 Processing Details

The file housekeeping process selects all rows from the BRDB_FILES_TO_HOUSEKEEP table, evaluates the age of the files referenced and deletes all files older than the stored retention period value. Files to housekeep include:

- Audit Files
- Support Archive Files

Where hard links have been created to Audit directories, the Audit System will be responsible housekeeping the resulting files.

In order to minimise development effort, it is highly recommended to use existing functionality used by APOP-Host system. However re-using functionality is not a requirement.

#### 7.2.6.8 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun for all pending files. If the failure reason is a Branch Database node/instance failure, the job will be rerun on another node/instance. Details are available in the Scheduling High Level Design [R12].

It is recommended that the Process Control functionality be used to control the execution at Process level i.e. just log one entry in the Process-Control table for each run of this process.

### 7.2.7 Data Aggregation Tools (BRDBX007)

The Data Aggregation provides the process control and business logic required to perform the data aggregations described in Section 5.4.

POL00115409



### 7.2.7.1 Application Type

The aggregation mechanism will be implemented in the form of PL/SQL stored packages/procedures/functions. The functionality may be handled by common PL/SQL objects as appropriate.

There will be a single point for calling all aggregation objects, which will be implemented in the form of a wrapper script developed as a Linux Shell script.

### 7.2.7.2 Inputs

The interface wrapper will accept the following mandatory command-line input parameters:

- **Aggregation Logical Name:** This is the logical name of the SQL Aggregation to be performed and is expected to correspond to a stored PL/SQL object that will be the access point to implement the aggregation.
- **Input Trading/Business Date:** This date is used to filter the records used for aggregation. Format is CCYYMMDD.
- **Node/Instance Id:** This indicates the Branch Database Node that the aggregation job instance needs to execute against. Values range from 1…4.

In addition the aggregation wrapper will accept the following optional parameters that may be needed for specific aggregation jobs.

- **Optional Parameter #1:** This is an alphanumeric parameter that can be up to 100 characters long.
- **Optional Parameter #2:** This is an alphanumeric parameter that can be up to 100 characters long.
- **Optional Parameter #3:** This is an alphanumeric parameter that can be up to 100 characters long.



All input parameters will be passed to the PL/SQL object that performs the aggregation.

### 7.2.7.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures.

### 7.2.7.4 Location

The wrapper script should reside in **$BRDB_SH** directory on each Node.

The aggregation business logic will reside within the Branch Database.

### 7.2.7.5 Scheduling

Multiple instances of the aggregation wrapper script run at different times during the day. With a few exceptions, each instance will run on a separate node of the Branch Database.

Refer to the Scheduling High Level Design [R12] for more details.

POL00115409



#### 7.2.7.6 Processing Details

The design principles behind the data aggregation tool and its logical implementation are discussed in Section 5.4.

The wrapper script validates the command-line input parameters and calls the relevant data aggregation PL/SQL object to pass on the parameters. The script traps / handles Node / Instance failures and returns appropriate return status back to the scheduler.

The aggregation PL/SQL object formulates the SQL statement required to implement each step of the aggregation required. The application logic required to implement various data aggregations to be performed is discussed in Section 5.4.

#### 7.2.7.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun from the point of failure. If the failure reason is a Branch Database node/instance failure, a number of recovery jobs will be fired on the other nodes/instances of the Branch Database. Details are available in the Scheduling High Level Design [R12].

The level of Process Control / point of restart for a data aggregation job will depend on whether it has been implemented as a single instance or four instances.

Typically for aggregation jobs that are implemented as a single instance, the restart point is entire run for the day. This covers all smaller aggregations where the amount of time lost in the event of a rerun is insignificant.

For aggregations implemented as four instances (one per database node), the restart point is at a Fad-Hash level. This covers all aggregation jobs where finer control may be needed due to the large volumes of data being processed.



There may be exceptions to this rule and these would be highlighted in Section 5.4.

### 7.2.8 Check Job Completion (BRDBC008)

The check job completion tool is an important component of the overnight schedule and is used by the scheduler to determine if a batch schedule that consists of multiple (Fad-Hash specific) restart points has completed successfully.

#### 7.2.8.1 Application Type

This is a Pro*C application.

#### 7.2.8.2 Inputs

The check job tool accepts the following mandatory command-line input parameters:

- **Process Name:** This is the name of the process that runs the batch job in question e.g. BRDBC002.
- **Input Business/Selection Date:** This indicates the business day for the processing. Format is CCYYMMDD.

The check job tool can also accept the following optional command-line input parameters:

- **Input Parameter 1:** First component used to form the “Job Instance Name”.
- **Input Parameter 2:** Second component used to form the “Job Instance Name”.

POL00115409



- **Input Parameter 3:** Second component used to form the “Job Instance Name”.

#### 7.2.8.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

#### 7.2.8.4 Location

Script should reside in **$BRDB_PROC** directory on each Node.

#### 7.2.8.5 Scheduling

The process runs a number of times every day at the completion of each batch job schedule involving running jobs that are controllable at Fad-Hash level. The job will run on Node-1 of the Branch Database.

Refer to the Scheduling High Level Design [R12] for more details.



#### 7.2.8.6 Processing Details

The process forms the “Job Instance Name” template using Input Parameter 1(if present) + “ - “ + Input Parameter 2 (if present) + “ - “ + Input Parameter 3 (if present). If none of the optional parameters are present, the Instance-Name template remains as NULL.

For each Fad-Hash value (0…127), the Instance Name is completed by appending “ - “ + Fad-Hash to the “Job Instance Name” template and is checked against the Process-Control using input Process Name and Business Date. Run-Number must be left as 1. If the query returns that the instance was never run or did not finish successfully, the check tool exits with failure.

If all instances finished successfully, the check tool exits with success.

The check tool must indicate success/failure status of each Process-Control entry on standard output to aid support.

#### 7.2.8.7 Handling Failures and Rerun ability

The check tool is designed to fail if the jobs instances that it is checking for have not finished successfully. Recovery actions have been defined in the Scheduling High Level Design [R12] in the event of failures.

It is recommended that the Process Control functionality should not be used to control the execution.

### 7.2.9 End Of Day (BRDBC009)

The BRDB End of Day process will run as the last activity prior to the database backup activities.

#### 7.2.9.1 Application Type

This is a Pro*C application.

#### 7.2.9.2 Inputs

None.

POL00115409



### 7.2.9.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

### 7.2.9.4 Location

Script should reside in **$BRDB_PROC** directory on each Node.

### 7.2.9.5 Scheduling

The application runs every day in the form of a single instance after the completion of all other host application instances (except the interactive batch jobs). The job will run on Node-1 of the Branch Database.

### 7.2.9.6 Processing Details

It is highly recommended to use existing functionality used by TES-Host system.

End of Day will set the values of the system flags that control the execution of the Track & Trace and the Guaranteed Reversals feed to NPS, to allow such processes to exit with Success. This will allow the interactive Batch Interfaces to exit cleanly.

Additionally the process will check whether instances listed in BRDB_OPERATIONAL_INSTANCES table have restarted, in which case that instance's "Is-Available" flag will be reset to "Y". The check should be in the form making a TCP/IP (not bequeath) connection to the instance to run a dummy SQL. This will implicitly check if (a) the listener is up and (b) the instance (and hence the ASM) is up.

The possible outcomes of the check and action to be taken are summarised below:

| BRDB Instance / ASM / node / Service | BRDB-Operational-Instances Is-Available flag value | Action to be taken |
|---|---|---|
| Up (i.e. dummy SQL returns expected response) | Y | None. This is the expected result. |
| Up (i.e. dummy SQL returns expected response) | N | Reset the Flag value to Y. Write a diagnostic message to STDOUT. |
| Down (dummy SQL fails due to any reason) | N | Raise a B (medium) priority exception to highlight that the node is down. |
| Down (dummy SQL fails due to any reason) | Y | Raise an A (high) priority exception to highlight that the node is down but metadata indicates that the node is up.<br>This will require an immediate response from first line to investigate and reset the flag in **BRDB-Operational-Instances** table. |

### 7.2.9.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance should rerun from the beginning of the job. If the failure reason is a Branch Database node/instance failure, the job will be rerun on another node/instance. Details are available in the Scheduling High Level Design [R12].

It is recommended that the Process Control functionality be used to control the execution at Process level.

POL00115409



## 7.2.10 Detect and Report Node Failures using FAN (BRDBX010 build directory)

The failure detection script uses Oracle's Fast Application Notification (FAN) facility. FAN is a feature that filters and publishes only high-level failure events such as the unavailability of a Node / Branch Database Instance or unavailability of the underlying cluster drivers.

The failure detection script will be one of the scripts called by the Oracle supplied failure notification wrapper script. **This script will be called 'fan_event_handler.ksh'**. This script will then call a second script

This second script .'brdb_inst_unavail.ksh' uses the Failure Notification events passed in on command-line to update the BRDB_OPERATIONAL_INSTANCES table. This will allow the Branch Access Layer to determine the correct instance to access for an incoming Fad-Hash request.

Please note this script is **not** called by the TWS schedule.

### 7.2.10.1 Application Type

This is a Linux Shell Scripts.



### 7.2.10.2 Inputs

The first script is called by Oracle background processes and accepts the following command line parameter pairs in the format PARAMETER=VALUE :-

- Version
- Service
- Database Name
- Instance Name
- Host Name
- Status
- Reason
- Card
- Timestamp

The second script must accept the following Oracle FAN input parameters :-

- Event type
- Status
- Database Name
- Host Name
- Event String

Ideally it should accept all the input parameters that Oracle's FAN is capable of passing to the script.

### 7.2.10.3 Outputs

Return Code 0 for Successful Execution

POL00115409



Return 1 for all other types of Failures

#### 7.2.10.4 Location

The second script should reside in **$BRDB_SH** directory on each node of the Branch Database cluster.

It should be called by a first script, which resides in the **$ORA_CRS_HOME/racg/usrco** directory on each Branch Database Node. Note that the wrapper script might be calling other alerting/monitoring scripts hence any changes to the wrapper script must be made with care.

#### 7.2.10.5 Scheduling

There is no scheduling involved. The script resides on the Branch Database nodes and will be invoked by Oracle's FAN in the event of a high priority event.

#### 7.2.10.6 Processing Details

The wrapper script should perform two functions:

- Propagate the notification event as a 'Fatal' event onto the SYSLOG for onward routing to Event Monitoring. This can be achieved by using the OS level command "logger".
- Call the application component of the failure detection script in the $BRDB_SH directory.

The failure detection script that resides in the $BRDB_SH directory performs the following functions:

- Check for all failures at Node and ASM level as well as Instance level where the database name is "BRDB" (to protect against failure of any other database sharing the node with BRDB).
- If the failure **Status** is "**down**" or "**nodedown**", script should use the BRDB service to connect to the database and update the "IS_AVAILABLE" status for the failed node in the BRDB_OPERATIONAL_INSTANCES table to "N".



#### 7.2.10.7 Handling Failures and Rerun ability

Writing the event to SYSLOG should not cause the script to fail but if in the rare event of this occurring, the failure cannot be detected and will pass un-noticed.

If the function of updating the BRDB_OPERATIONAL_INSTANCES table cannot be performed, the script should log another event on the SYSLOG to that effect and also log an Operational Exception.

The script is not controlled by Process Control.

### 7.2.11 Update System Parameters (BRDBX011)

The Update-Parameters script will be executed at various times during the Business Day to set / reset the values of the System Parameter that is passed on the command-line.

#### 7.2.11.1 Application Type

This is a Linux Shell Script with optional PL/SQL elements.

#### 7.2.11.2 Inputs

The script accepts the following mandatory command-line input parameters:

- **Parameter Name:** This is the name of an existing Branch Database System Parameter.

POL00115409



- **Parameter Type:** Indicates the data-type of the System Parameter. Possible values are:
  - **N –** Number
  - **D –** Date (format YYYYMMDD)
  - **T –** Text (Alphanumeric value)
- **Parameter Value:** Value to set the parameter to.

#### 7.2.11.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

#### 7.2.11.4 Location

Script should reside in **$BRDB_SH** directory on each Node.

#### 7.2.11.5 Scheduling

The script will be run at multiple times during the business day in the form of a single instance The script will run on Node-1 of the Branch Database.



#### 7.2.11.6 Processing Details

The script needs to use the three input parameters to formulate the SQL Statement to execute against the BRDB_SYSTEM_PARAMETERS table. The SQL statement should match the following template:

```
UPDATE  brdb_system_parameters
SET
        parameter_date = TO_DATE (input-parameter-value, 'YYYYMMDD') - If input-parameter-type is 'D'
…or…
        parameter_number = input-parameter-value - If input-parameter-type is 'N'
…or…
        parameter_text = input-parameter-value - If input-parameter-type is 'T'

WHERE   parameter_name = input-parameter-name
```

#### 7.2.11.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance should rerun from the beginning of the job. If the failure reason is a Branch Database node/instance failure, the job will be rerun on another node/instance and this will be handled by TWS. Details are available in the Scheduling High Level Design [R12].

It is recommended that the Process Control functionality should not be used to control the execution at Process level. However every run must involve creating Process-Audit entries as per HADDIS guidelines [R7].

### 7.2.12 SSC Transaction Correction Tool (BRDBX015)

This utility will allow SSC to correct transactions by adding compensating correction records to transactional / accounting / stock tables in the BRDB database. The utility will also audit the changes made and ensure that the changes + audit are performed as a single commit unit. There will be no updating / deleting of records in the Branch database. This is done by reading and executing the SQL in transaction files that are stored in the file system.

POL00115409



### 7.2.12.1 Application Type

This is a Linux shell script based application. The core functionality will be implemented in an Oracle PL/SQL package, which will be called by the Linux shell script.

The shell script will be owned by Linux user "supporttooluser" and it is deliberately kept separate from the standard $BRDB_SH directory so that access to the script and the associated components can be restricted to authorised users.. The PL/SQL package PKG_BRDB_TXN_CORRECTION will be owned by Oracle user "OPS$SUPPORTTOOLUSER". The PL/SQL package PKG_BRDB_TXN_CORRECTION will execute with the permissions of the OPS$SUPPORTTOOLUSER account and can only insert rows into the transaction tables as controlled by an entry in BRDB_SYSTEM_PARAMETERS. The account will not have update or delete privileges.

The BRDBX015.sh script logs into Oracle as '/' (i.e. OPS$<SSCusername>), therefore in order to run, all of the Oracle OPS$ users for the SSC users require database privileges on the package as follows:

| Object Name | Object Type | Ins | Sel | Upd | Del | Exe |
|---|---|---|---|---|---|---|
| PKG_BRDB_TXN_CORRECTION | Package | | | | | X |

Note that the create_db_user.sh script has been changed to grant the above privilege to new SSC Oracle users when they are created.



### 7.2.12.2 Inputs

The utility accepts the following mandatory command-line input parameter:

- **Transaction-File-Name:** Name of the file (including path) that contains the compensating transaction correction record.
- **Branch Code:** The number of the branch specific to this correction transaction.

### 7.2.12.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

### 7.2.12.4 Location

The script should reside in /app/brdb/trans/support/brdbx015 directory on each Node.

Each of the transaction tables that are allowed to have balancing transactions inserted on them has an associated template file. Each file contains a template of an INSERT statement for that table, in the required format, and listing all of the columns on the table. Users should create their own transaction file based upon the relevant template file, substituting the values they require into the SQL. Note that some of the column values specified in the template should not be changed – these are annotated with comments as appropriate.

POL00115409



The collection of template files are available from source control along with the other components. They are also included in the Branch Support Guide.

The completed transaction file, based on the template file, contains a SQL INSERT statement that creates the required balancing transaction. The file must be placed in this directory:

/app/brdb/trans/support/brdbx015/input

When execution is complete the file is then moved to directory '/app/brdb/trans/support/brdbx015/output' and the log file is created in directory '/app/brdb/trans/support/brdbx015/log'. Log file will be named using the following convention:

<transaction_file_name>_<CCYYMMDDHHMISS>.log

### 7.2.12.5 Scheduling

The utility will not run on a regular basis. It will be used manually by SSC (third-line) support, please refer to the Branch Support Guide [R43] for usage..



### 7.2.12.6 Processing Details

Warning: The use of this powerful tool has inherent risks. If the SQL statement is incorrect or badly written, it is possible to cause unintended consequences, some of which may cause serious problems to the Branch Database. It is expected that only a small number of skilled staff will run this tool and that they will have detailed guidance as to when and how to use the tool.

Having logged into their own Unix user, the SSC team members will change directory to the /app/brdb/trans/support/brdbx015 directory and place their transaction file in the /app/brdb/trans/support/brdbx015/input sub-directory. They will then invoke BRDBX015 manually. The shell script module will be owned by the Unix user "supporttooluser".

- From the Unix command prompt, execute the following

  ./BRDBX015.sh MyTransactionFile.sql 2001

  where the first parameter is the transaction file name and the second parameter is the branch code where the balancing transaction is going to be applied. Note that the branch code must exist in the database, and must not be for a closed branch. If this is not the case, then an error message will be shown and the run aborted.

The tool will read the contents of the input transaction file, which will be in the form of single SQL-insert statement.

The SQL also includes a number of bind variables (e.g :bind_branch_code). Actual values for these fields will be substituted into the SQL before it is executed. The available bind variables are listed below, together details of the values that will be substituted for them if they appear in a transaction file:

- :bind_branch_code : substituted with the values of the branch code argument of BRDBX015

POL00115409



- :bind_SSC_user : substituted with the Oracle user that is carrying out the actual insert i.e. SUPPORTTOOLUSER
- :bind_instance_name : substituted with the name of the Oracle instance upon which the tool is run

The correction tool places a number of constraints on the contents of the transaction file. These are necessary in order to provide a defined baseline upon which it can base its operation. If any of the constraints are violated then validation will detect it and abort the run with a meaningful error message. The constraints are as follows:

- The transaction file must be less than 32K in size
- The transaction file must only contain Unix-style end of line markers (EOL), not DOS format end of line markers (CR/EOL)
- The transaction file can only contain a single SQL statement. If more than one balancing transaction is required then more than one transaction file must be created, each of which is executed with a separate run of the tool
- If the transaction file contains an introductory comment, then it must be a '/* …… */' style comment, not a '-- ……' style comment
- The closing '*/' of the introductory comment must have a trailing space (i.e. '….. */ ')
- The run symbol at the end of the SQL must be a ';' , not '/', and must have a trailing space (i.e. '…..; ')
- The SQL must be a valid SQL statement according to the normal Oracle SQL parsing rules (e.g. valid syntax, objects accessible etc)
- The SQL must begin with 'INSERT INTO OPS$BRDB.' and be of the form 'INSERT INTO ….. SELECT ….. FROM dual, (SELECT ….. FROM …. WHERE …..)'.
- The table name must be one of the tables named in the BRDB_TXN_CORRECTION_ALLOWED_TABLES1 or BRDB_TXN_CORRECTION_ALLOWED_TABLES2 configuration parameters
- All of the columns that exist on the table in question must be explicitly named. It is not necessary for every listed column to be on a separate line, but this is advisable for readability.
- The values to be inserted must be provided by the 'SELECT … FROM dual …'. Each value must be on a separate line. Trailing comments are allowed, but must be a '-- …..' style comment. Any such comment must not include any commas. All columns must have values provided for them (even if that value is NULL).
- Certain columns are common between a subset of the transaction tables. In some cases, these columns should be set to the same value no matter what table is in use. With the exception of the bind variables listed earlier, the value that the SQL will try to insert is under the control of the user (i.e. it is determined by the value specified in the SQL). However, the tool can be configured to validate that the value specified in the SQL matches that expected. In order to do this, set the BRDB_TXN_CORRECTION_ENFORCED_VALUES configuration parameter to include the field and the required value.

  The parameter is populated as a comma-delimited list of name/value pairs, where the name is the name of the column name, and the value is the value to be enforced. As released, this configuration parameter is set to:

POL00115409



NODE_ID=99,APP_SERVER_NODE_NAME=999,BRANCH_USER=:bind_SSC_user,BRDB_INSTANCE_NAME=:bind_instance_name

which, for example. ensures that if a 'node_id' column exists on the transaction table, it's value is specified as 99. If there is no 'node_id' on the transaction table, then no value is enforced for that field. Note that if the parameter does not exist, then no values are enforced in the SQL.

Since all transaction tables are partition and sub-partitioned by fad hash. It is vital that the insert statement set the value of the fad hash to the correct value corresponding to the branch code. Development have created within the templates SQL code that derives the fad hash correctly, this should be used as the basis of the SQL preparation and not amended as per the comments within the template file.

If valid, the SQL statement is modified by substituting actual values for bind variables, and the modified SQL statement is executed and logged in the journal as an XML string.



The SQL statement being executed will be logged in the table BRDB_TXN_CORR_JOURNAL. The format of the data to be written to the column JOURNAL_XML is:

"<?xml version="1.0" encoding="UTF-8"?>

<Support_Insert>

<Unix_User>Unix User Name</Unix_User>

<Oracle_User>Oracle User Name</Oracle_User>

<Sql>SQL Statement</Sql>

</Support_Insert>"

where :

- Unix User Name is the Unix user name under which the user logged in
- Oracle User Name is Oracle user that is carrying out the actual insert i.e. SUPPORTTOOLUSER
- SQL Statement is the final (i.e. after substituting actual values for bind variables) SQL that is executed to insert the balancing transaction

More details on the process and the risks will be defined in the Branch Database Support Guide [R43].

### 7.2.12.7 Handling Failures and Rerun ability

In the event of a node / instance failure, if the failure reason could be interpreted from the Oracle Error, the utility needs to fail with error code 99. For any other failures, the exit code should be 1 and an operational exception should be logged.

The Process Control functionality should not be used to control the execution at Process level. However the Process Audit log (brdb_process_audit) must be populated once at the beginning and then end of execution.

POL00115409



## 7.2.13 BRDB transfer to DWh (BRDBX020)

This utility is the file transfer for BRDB Branch Migration Status data feed to DWh.

### 7.2.13.1 Application Type

This is a Linux shell script based application.

### 7.2.13.2 Inputs

The utility accepts the following mandatory command-line input parameter:

- **Input Business Date:** This date is used to source files. Format is CCYYMMDD.

### 7.2.13.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures



### 7.2.13.4 Location

The script should reside in **$BRDB_SH** directory on each Node.

### 7.2.13.5 Scheduling

The utility will be run on a daily basis via the BRDB TWS schedule.

### 7.2.13.6 Target Directories

See REPOSITORY env variable.

### 7.2.13.7 Processing Details

The utility will perform the following functions:

1. Accept command line input for the effective date.

2. Deduce the source file and directory based on the BRDB_MSU_OUTPUT env variable and the effective date.

3. Form the destination directory based and file name based on the REPOSITORY env variable, system name and effective date.

4. Check the availability of the source and destination directories and the source file.

5. Moves the source file to the destination directory. NOTE: This means that the source directory will not have the file, once it has been successfully moved to the destination directory.

POL00115409



#### 7.2.13.8 Handling Failures and Rerun ability

In the event of a node / instance failure, if the failure reason could be interpreted from the Oracle Error, the utility needs to fail with error code 99. For any other failures, the exit code should be 1 and an operational exception should be logged.

## 7.2.14 Pause / Start Streams Propagation (BRDBX021)

This utility will pause or restart the propagation of Streams messages to the BRSS system.

### 7.2.14.1 Application Type

This is a Linux shell script based application.

### 7.2.14.2 Inputs

The utility accepts the following mandatory command-line input parameter:

- **Action-Type:** The type of action to perform on Streams Propagation. Allowable values are "PAUSE" and "RESTART".



### 7.2.14.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures

### 7.2.14.4 Location

The script should reside in **$BRDB_SH** directory on each Node.

### 7.2.14.5 Scheduling

The utility will not run on a regular basis. It will be used either manually or called from another script as required.

### 7.2.14.6 Target Directories

No files or directories are written to by this utility.

### 7.2.14.7 Processing Details

The utility will perform two actions depending on the value of input parameter "Action-Type". For the "PAUSE" action, the utility will pause the propagation of LCRs to the remote queue in the BRSS system.

For the "RESTART" action, a previously paused propagation of LCRs to BRSS will be restarted.

There are a number of ways of pausing / restarting queue propagation such as unscheduling / scheduling propagation or by disabling and enabling a propagation schedule. The exact method used will be determined in the Low Level design.

POL00115409



The utility can 'hard-code' the names of Oracle objects associated with streams replication to BRSS such as rule, propagation schedule, queue name etc.

#### 7.2.14.8 Handling Failures and Rerun ability

In the event of a node / instance failure, if the failure reason could be interpreted from the Oracle Error, the utility needs to fail with error code 99. For any other failures, the exit code should be 1 and an operational exception should be logged.

The Process Control functionality should not be used to control the execution at Process level. However the Process Audit log (brdb_process_audit) must be populated once at the beginning and then end of execution.

## 7.2.15 Hydra-specific XML Data Processor (BRDBX030)

The XML data processor parses XML data present in an Oracle database column and loads it in relational form into one or more target tables.

This is a hydra-specific tool and is expected to be put out of use once the hydra phase is complete. The source XML tag names and the target table and column names are hard-coded with the view of simple implementation.



### 7.2.15.1 Application Type

The XML data processor will be implemented in the form of a PL/SQL stored package.

There will be a single wrapper script developed as a Linux Shell script to provide access to the PL/SQL object and to pass command-line input parameters.

The script will call the end-of-day processing first then call the in-day processing second. This is a logical requirement because the stock units and stock positions are required before the in-day process 'flips' the branch status from Horizon to HNG-X. In-day processing will only be performed for 'NORMAL' runs.

### 7.2.15.2 Inputs

The XML processor will accept the following mandatory command-line input parameters:

- **Input Trading/Business Date:** This date is used to filter the records used for parsing. Format is CCYYMMDD.
- **Node/Instance Id:** This indicates the Branch Database Node that the aggregation job instance needs to execute against. Values range from 0…4.

  [DN: Node-Id is not strictly needed but provides with the flexibility of being able to run the parser on multiple nodes if performance issues arise.] NOTE. When the script is run on a single node then the command line parameter for Node/Instance Id should be 0 (zero). This will then iterate through each fad_hash / instance goup.
- **Run Mode:** 'NORMAL' (default) or 'CATCHUP' - This determines whether or not to apply TPS filtering.

All input parameters will be passed to the PL/SQL object that performs the XML processing.

### 7.2.15.3 Outputs

Return Code 0 for Successful Execution

POL00115409



Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures.

#### 7.2.15.4 Location

The wrapper script should reside in **$BRDB_SH** directory on each Node.

The XML processing business logic will reside within the Branch Database.

#### 7.2.15.5 Scheduling

A single instance of the XML processor will be run in 'NORMAL' mode once during the processing day following the completion of the Aggregation and pre-processing (BRDBX033) once every Sunday in 'CATCHUP' mode. NOTE. The module has been designed to be able to run on all four nodes concurrently. This feature is required if the rollout of branches become compressed and a large number of branches are migrating concurrently.

Refer to the Scheduling High Level Design [R12] for more details on scheduling of the XML processor jobs.



#### 7.2.15.6 Processing Details

The XML processing step forms the second part of the processing involved for the hydra-specific ongoing and 'in-day' feeds from TPS. The logical implementation of the XML processing required including the XML tag-names – database column mappings are discussed in [R40] and [R28].

The process must be able to logically partition the branches by fad_hash code so that concurrent runs of the job on different nodes do not interfere with each other. The split of branches across instances will be controlled by the mapping of branch codes to fad hash to available instances.

The processing will use a series of temporary / working tables to decompose the XML statements into their constituent parts. These tables are 'list' partitioned by either fad_hash or instance node id.

The wrapper script validates the command-line input parameters and calls the relevant PL/SQL objects in order. The script traps/handles Node/Instance failures and returns appropriate return status back to the scheduler.

The XML data processor PL/SQL object reads the XML data that is stored in a CLOB in the source tables, parses and converts the XML into relational form and loads the target tables.

#### 7.2.15.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun from the point of failure. If the failure reason is a Branch Database node/instance failure, the failed job instance will be fired on one of the other nodes/instances of the Branch Database. Details are available in the Scheduling High Level Design [R12].

The level of Process Control / point of restart for the XML data processor is at the level of a single XML statement, which means that the processing must be repeated in the event of a restart after failure. However given the program will only select un-processed rows, the time of night the process is expected to run and the scope of the process (hydra-specific), a selective rerun is acceptable.

POL00115409



## 7.2.16 BRDB Reset JSN, SSN, USN (BRDBX031)

Counters do not always logoff tidily in the HNG-X environment, hence the information maintained in table BRDB_BRANCH_NODE_INFO for JSN/USN/SSN can be out of date. A tidy counter logoff will update the values in BRDB_BRANCH_NODE_INFO correctly.

To combat this the job searchs the underlying BRDB tables for the maximum value for each JSN, USN and SSN used that day. If the value is greater in the base table then the BRDB_BRANCH_NODE_INFO (BBNI) value is out of date and can be updated with the value from the base table. For JSN & USN we don't have to worry about rollover but as SSN is a NUMBER(6) in the base table this will rollover every @7 years for a counter. As such for SSN the script caters for SSN rollover from 999,999 to 0+.

### 7.2.16.1 Application Type

The BBNI maintenance script will be implemented in the form of a Linux Shell script.

### 7.2.16.2 Inputs

The Linux Shell script takes a single parameter of business day, whose format is CCYYMMDD.



### 7.2.16.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures.

### 7.2.16.4 Location

The wrapper script should reside in **$BRDB_SH** directory on each Node.

### 7.2.16.5 Scheduling

A single instance of the script will be run once during the processing day following the completion of aggregation. The module is able to run on any available node.

Refer to the Scheduling High Level Design [R12] for more details on scheduling of the BBNI maintenance job.

### 7.2.16.6 Processing Details

a) Truncate the table WKG_BRDB_JSN_USN_SSN
b) Populate the WKG_BRDB_JSN_USN_SSN fields from BRDB_BRANCH_NODE_INFO.
c) For each counter (node-id) that exists in BRDB_RX_MESSAGE_JOURNAL find the current maximum JSN (CURRENT_MAX_JSN) in table BRDB_RX_MESSAGE_JOURNAL and update it in WKG_BRDB_JSN_USN_SSN.
d) If the value for the CURRENT_MAX_JSN now in WKG_BRDB_JSN_USN_SSN is greater than that currently in BRDB_BRANCH_NODE_INFO then update BRDB_BRANCH_NODE_INFO. CURRENT_MAX_JSN.

POL00115409



e) For each counter (node) that exists in BRDB_RX_RECOVERY_TRANSACTIONS find the current maximum USN (CURRENT_MAX_USN) in table BRDB_RX_RECOVERY_TRANSACTIONS and update it in WKG_BRDB_JSN_USN_SSN.
f) If the value for the CURRENT_MAX_USN now in WKG_BRDB_JSN_USN_SSN is greater than that currently in BRDB_BRANCH_NODE_INFO then update BRDB_BRANCH_NODE_INFO. CURRENT_MAX_USN.
g) For each counter (node) that exists in BRDB_RX_REP_SESSION_DATA find the current maximum SSN (SESSION_ID) in table BRDB_RX_REP_SESSION_DATA and update CURRENT_MAX_SSN in WKG_BRDB_JSN_USN_SSN.
h) If the value for the CURRENT_MAX_SSN now in WKG_BRDB_JSN_USN_SSN is greater than that currently in BRDB_BRANCH_NODE_INFO then update BRDB_BRANCH_NODE_INFO. CURRENT_MAX_SSN. If the existing value of CURRENT_MAX_SSN now in BRDB_BRANCH_NODE_INFO is > 990,000 and the value in WKG_BRDB_JSN_USN_SSN is between 0 and 009,000 then update BRDB_BRANCH_NODE_INFO. CURRENT_MAX_SSN with the value in WKG_BRDB_JSN_USN_SSN.CURRENT_MAX_SSN. Also the field CURRENT_MAX_SSN_JOURNAL_DATE will be updated.

#### 7.2.16.7 Handling Failures and Rerun ability

In the event of a failure, the failed job needs to be able to rerun from the start. If the failure reason is a Branch Database node/instance failure, the failed job instance will be fired on one of the other nodes/instances of the Branch Database. Details are available in the Scheduling High Level Design [R12]



## 7.2.17 Hydra-specific XML Data Pre-Processor (BRDBX033)

The XML data processor parses XML data present in an Oracle database column and loads it in relational form into one or more target tables.

This is a hydra-specific tool and is expected to be put out of use once the hydra phase is complete. The source XML tag names and the target table and column names are hard-coded with the view of simple implementation.

### 7.2.17.1 Application Type

The XML data processor will be implemented in the form of a PL/SQL stored package.

There will be a single wrapper script developed as a Linux Shell script to provide access to the PL/SQL object and to pass command-line input parameters.

The script will call the end-of-day pre-processing.

### 7.2.17.2 Inputs

The XML processor will accept the following mandatory command-line input parameters:

- **Input Trading/Business Date:** This date is used to filter the records used for parsing. Format is CCYYMMDD.
- **Node/Instance Id:** This indicates the Branch Database Node that the aggregation job instance needs to execute against. Values range from 0…4.

POL00115409



[DN: Node-Id is not strictly needed but provides with the flexibility of being able to run the parser on multiple nodes if performance issues arise.] NOTE. When the script is run on a single node then the command line parameter for Node/Instance Id should be 0 (zero). This will then iterate through each fad_hash / instance goup.

- **Run Mode:** 'NORMAL' (default) or 'CATCHUP' - This determines whether or not to apply TPS filtering.

All input parameters will be passed to the PL/SQL object that performs the XML processing.

### 7.2.17.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures.

### 7.2.17.4 Location

The wrapper script should reside in **$BRDB_SH** directory on each Node.

The XML processing business logic will reside within the Branch Database.



### 7.2.17.5 Scheduling

A single instance of the XML processor will be run in 'NORMAL' mode once during the processing day following the completion of the TPS filter branches (BRDBX034) and TPS-BRDB hydra-specific interface (see Section **Error! Reference source not found.**) once every Sunday in 'CATCHUP' mode. NOTE. The module has been designed to be able to run on all four nodes concurrently. This feature is required if the rollout of branches become compressed and a large number of branches are migrating concurrently.

Refer to the Scheduling High Level Design [R12] for more details on scheduling of the XML processor jobs.

### 7.2.17.6 Processing Details

The XML processing step forms the first part of the processing involved for the hydra-specific ongoing feeds from TPS. The logical implementation of the XML processing required including the XML tag-names – database column mappings are discussed in [R40] and [R28].

The process must be able to logically partition the branches by fad_hash code so that concurrent runs of the job on different nodes do not interfere with each other. The split of branches across instances will be controlled by the mapping of branch codes to fad hash to available instances.

The processing will use a series of temporary / working tables to decompose the XML statements into their constituent parts. These tables are 'list' partitioned by either fad_hash or instance node id.

The wrapper script validates the command-line input parameters and calls the relevant PL/SQL objects in order. The script traps/handles Node/Instance failures and returns appropriate return status back to the scheduler.

The XML data processor PL/SQL object reads the XML data that is stored in a CLOB in the source tables, parses and converts the XML into relational form and loads the target tables.

POL00115409



#### 7.2.17.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance needs to be able to rerun from the point of failure. If the failure reason is a Branch Database node/instance failure, the failed job instance will be fired on one of the other nodes/instances of the Branch Database. Details are available in the Scheduling High Level Design [R12]

The level of Process Control / point of restart for the XML data processor is at the level of a single XML statement, which means that the processing must be repeated in the event of a restart after failure. However given the program will only select un-processed rows, the time of night the process is expected to run and the scope of the process (hydra-specific), a selective rerun is acceptable.

## 7.2.18 Hydra-specific TPS filter loader (BRDBX034)

This loader will pull branch_codes from rdds for those branches pending migration within the next 14 days.



### 7.2.18.1 Application Type

The loaded will be implemented in the form of a PL/SQL stored package.

There will be a single wrapper script developed as a Linux Shell script to provide access to the PL/SQL object and to pass command-line input parameters.

### 7.2.18.2 Inputs

The processor will accept the following mandatory command-line input parameters:

- **Input Trading/Business Date:** This date is used to filter the records used for parsing. Format is CCYYMMDD.

  All input parameters will be passed to the PL/SQL object that performs the processing.

### 7.2.18.3 Outputs

Return Code 0 for Successful Execution

Returns 99 for a failure due to Node/Instance failure and 1 for all other types of Failures.

### 7.2.18.4 Location

The wrapper script should reside in **$BRDB_SH** directory on each Node.

The processing business logic will reside within the Branch Database.

### 7.2.18.5 Scheduling

A single instance of the loader will be run once during the processing day following the completion of 'start of day'. NOTE. The module has been designed to be able to run on one node only.

### 7.2.18.6 Processing Details

Merge into brdb_hydra_tps_filter all branch_codes in the rdds view rd_migrating_branches where target_migrationj_day is less than 14 days ahead.

POL00115409



### 7.1.1.7 Handling Failures and Rerun ability

In the event of a failure, the failed job instance should be re-run

POL00115409



# 8 Branch Standby Database Solution

The Branch Database acts as the sole persistent store for Counter trading data. Unavailability of the Branch Database would mean loss of all trading capabilities at the Counter. The Branch Database solution needs to provide an alternate capability to prevent it from being the single point of failure that results in a site DR.

The Branch Database will use SRDF replicated Service Level-1 storage) for its data store to protect the data against a storage device failure. However SRDF does not provide any protection against table / index level block corruptions that may lead to either loss of data or prevent the database from functioning. As discussed later in Section 12.6.4, Oracle block corruptions can potentially result in the Branch Database being unavailable for a length of time that may result in violation of availability targets (see 12.2.1).

In this scenario, alternate means of trading will be provided via a standby database, which will be an asynchronously updated clone of the BRDB. The standby database will be implemented using Oracle Data Guard technology, which provides a highly efficient means of maintaining a continually updated database copy.

The standby database data-store will reside in a separate EMC frame (cabinet) from the Branch Database. This will be done to mitigate against any software problems that affect the entire storage frame (cabinet).

In order to provide speedy reporting on corruptions, Data Guard will be used to check for corruptions in source database blocks before they are applied to the BRDB-Standby.



## 8.1 Standby Database Configuration

The BRDB-Standby will be implemented as a “physical standby” database i.e. the database will provide an identical copy of BRDB in terms of database block structures and will be maintained by applying changes made to the standby and archived redo-log files of BRDB.

The BRDB will be configured to propagate changes to standby database in “Maximum Performance” mode, which provides the highest level of data protection that is possible without affecting the performance of the primary database. This is accomplished by allowing a transaction to commit as soon as the redo data needed to recover that transaction is written to the local online redo log. The primary database’s redo data stream is also written to the standby database, but that redo stream is written asynchronously with respect to the transactions that create the redo data.

Commits in BRDB will not be synchronous with propagation of changes to BRDB-Standby. Thus in the event of a switch to standby, the standby will contain changes up to the last transaction in the most recent redo log file. Transactions in the online redo-logs will need to be recovered by using the online redo logs from primary.

The physical standby database is maintained by applying redo data from the standby redo log files. When LGWR writes to the redo logs on the primary, a network server process (LNS) will read these logs and communicate with an RFS (Remote File Service) process, which will write to the standby logs. The write to the standby logs will be asynchronous so there is no pressure for a "Commit Complete" record to be returned to the primary to allow processing to continue. This configuration overall means that the LGWR processes on the primary RAC instances are under no extra workload to keep the standby log-files up-to-date.

The archiver process(es) on the standby will then write the logs to an archive destination at which stage the Managed Recovery Process (MRP) will apply the archive logs to the standby. In effect the redo logs will be up-to-date on both sites, the only delay being the archiving of the logs on standby before they are

POL00115409



applied to the database. In the event of a failover, the standby should in theory be up-to-date by the time the business decision is taken to fail the primary over.

With Oracle10g, there is the possibility of Real-Time Apply i.e. the MRP process reads the standby redo logs (as opposed to the archive logs) and applies them immediately to the database. Bearing in mind the overhead that such a configuration would present on the (already overloaded) single-node Standby, this option has ***not*** been chosen. A further problem with Real-Time Apply is that there can only be 1 MRP process per server (yet multiple RFS processes) and there is a strong possibility that the MRP process would not be able to keep up with the workload on the 1-node standby especially during peak times.

Use of the physical standby option means that the standby database cannot be open in read-write or even read-only modes while log-apply continues. This is acceptable as there is no requirement for the database to be used while the primary database (BRDB) is operational.

The main advantage of using a "physical standby" over the alternate "logical standby" is performance.

The Redo Apply technology used by the physical standby database applies changes using low-level recovery mechanisms, which bypass all SQL level code layers; therefore, it is the most efficient mechanism for applying high volumes of redo data and will present a reduced load on the BRDB-Standby server.



The standby solution will use a Data Guard Broker to manage and monitor the Data Guard replication. Using the broker will result in more meaningful alerts to appear in Grid Control if the replication were to fail.

## 8.2 Setting up Branch Standby Database

As outlined in Section 8.1, the Branch Standby database will be implemented as a physical standby database to ensure maximum performance of the primary and efficient use of standby resources.

This section outlines some of the important configuration changes required to the Branch Database and Branch Standby Database.

### 8.2.1 Initial Setup

The BRDB-standby database will be setup as a clustered database with four nodes (instances). The platform and the product component stack used will be the same as for the Branch Database (see Section 6.1).

BRDB-Standby will use a separate cluster from the BRDB RAC cluster.

Storage will be managed by ASM as in case of the Branch Database (see Section 6.2). Branch Standby Database will use non-SRDF replicated disks. Refer to [R11] for further details on the standby storage configuration.

Note: The Standby database does not use the default Branch Standby Database disk group names, but uses unique names, enabling a more flexible approach, e.g. should the need arise, the standby database could use the same ASM instance as the Primary.

The initial setup of the database will be facilitated using an RMAN backup copy of the Branch Database from the production environment and restoring it as a standby using RMAN.

Note: After the recovery of the backup onto the standby, it will be necessary to change the DBID inline with Oracle recommendations.

Once the standby environment has been set up, three of the four nodes will be switched off and added to the p-blade pool for use by other applications.. The BRDB-Standby database will operate as a single-node Oracle RAC cluster.

POL00115409



### 8.2.2 Configuration

#### 8.2.1.1 Branch Database Configuration Changes

Enable database level forced logging

Create a database password file to store the password for the SYS user

Standby redo log files will be used as per Oracle's recommendation. The recommended number of standby redo files is:
*(Maximum number of log files for each thread* ***(4)*** *+ 1) * Maximum number of threads* ***(4)***

This equates to 20 standby redo log files.

In addition, log and archived-redo log file name conversion must be set up to conform to the file naming conventions.

Refer to Appendix E – Suggested Oracle Initialisation Parameters for the full list of suggested Oracle Initialisation Parameters for the Branch Database and the Branch Standby including the changes required to support the Data Guard implementation.



## 8.3 Branch Standby Database Backups

The BRDB-Standby will not be backed up and will be able to use BRDB backups for recovery of any data that had originated from the primary (BRDB).

Although the Branch Standby Database will not be backed up daily, the Clusterware setup and configuration as well as the database configuration files will be backed up so that the most recent known cluster image can be restored from if needed.

The approach outlined for Branch Database backups in Sections 12.5.1 and 12.5.2 applies to Branch Standby Database as well. OCR and Voting Disk backups need to be taken daily and retained for 7 days.

## 8.4 Switching to BRDB-Standby

Once a decision has been made to switch the live service to using BRDB-Standby, there are two options available for the role transition (primary to standby):

- BRDB could 'switchover' with BRDB-Standby.

  A switchover is used when the primary is operational but a role reversal is required for a maintenance operation such as applying an OS patch.

  There should be no data loss during a switchover except in the case of a logical corruption in the BRDB online redo-logs (see Section 12.6.4).

- BRDB could 'failover' to BRDB-Standby.

  Failover is used if the primary becomes unavailable for any reason such as a fatal block corruption (see Section 12.6.4) and there is no possibility of recovering from the problem within operational service levels for failover, given agreement to proceed by CS Management..

There are a number of options available for managing the transitions to standby and role reversions. These are (a) Manually executed SQL scripts, (b) using the Data Guard Command-line utility and (c) using the Data Guard Broker GUI tool.

POL00115409



There are security implications that rule out using the Data Guard Broker GUI tool. At the time of writing this document, the Data Guard Command-line utility was considered error prone as it was failing to complete the role reversal intermittently.

Hence all scripting for transition to standby and for managing role reversals will be done using manually executed SQL scripts.

*Note: The Data Guard Broker command-line utility (dgmgrl) is in fact being used and will help support teams better diagnose and control the Data Guard configuration (specifically when using OEM Grid Control)*

### 8.4.1 Switchover to Standby

The Branch Database design does not support a Switchover to Standby.

### 8.4.2 Failover to Standby

The automatic failover feature of Oracle should be disabled. A decision to failover to standby must drive a pre-scripted but manually initiated process. The process will perform the following checks and functions:



- Check for missing archive logs on standby. This can be achieved using SQL, similar to the following:
  ```
  SQL> select thread#, low_sequence#, high_sequence# from v$archive_gap
  ```

- Ensure that all archive logs have been applied
  ```
  SQL> select unique thread# as thread, max(sequence#)
  SQL> over (partition by thread#) as last from v$archived_log
  ```

- Copy to standby any available primary archive logs that contain sequence numbers higher than the highest sequence number on the target standby database. If there are then register those files with the standby

  ```
  SQL> alter database register physical logfile 'log-file-name'
  ```

- Repeat all of the above three steps until no more gaps are highlighted. There is a possibility that additional gaps may be introduced / highlighted as archive log files are processed. This is especially true if archive logs have to be manually registered with the database.

- Disconnect all NAS storage that has been connected to the primary and NFS-mount them to the standby.

- Start the now defunct primary in mount mode.

- Switch the log files for all four threads and send them across to the standby. This will allow all the transactions present in the online redo logs at the time of failure to be recovered.

- Execute the following SQL or similar on the standby to initiate the failover:
  ```
  SQL> alter database recover managed standby database finish force
  ```

- Once the standby database has caught up with the pending redo information, transition the physical standby database to the primary database role as follows. Not that at this stage the database can no longer be used as a standby.

  ```
  SQL> alter database commit to switchover to primary
  ```

POL00115409



```
SQL> alter database open
```

- Perform cold backup of the new primary database using a BCV-split.
- Initiate Streams replication to BRSS from the new primary database.
- Change the DNS alias configuration so that all external access to “pbdb001…4” and “pbdb001…4-vip” is now redirected to the new primary -“pbds001…4” “pbds001…4-vip” respectively.
- Rebuild the old primary as a physical standby database and reconfigure Data Guard from new primary to new standby database.

[DN: This ‘role reversion’ i.e. old primary taking over the role of the standby can be achieved using other means such as using the ‘flashback’ feature. See assumptions 1.4 on flashback]

### 8.1.3 Resuming Streams feed to Branch History

No changes required to the new primary to resume the Streams replication to Branch Support System. Re-start of Streams will be required.

### 8.1.4 Role re-reversal

Once a role reversal has successfully completed, the (old standby) new primary will continue to operate at the Branch Database and the old primary will function as the standby even after the problems that lead to a failover have been resolved. Role re-reversal will not be initiated unless forced due to a failure on the new primary.

The reasons for not re-reversing roles is that both the Primary and Standby use same or similar CPUs & Memory and Tier-1 SAN storage and there would be no performance benefit in switching them back as they were. A role re-reversal will lead to an outage, which may be counted as a service unavailability SLA violation (see Section 12.2).

POL00115409



# 9 Platforms

This section provides a view of the platform and storage required to support the application systems ported to IRE11 / IRE19.

## 9.1 Branch Database

### 9.1.1 Hardware Requirements

#### 9.1.1.1 Platform

The Branch Database (BRDB) will use Fujitsu-Siemens BladeFrame environment. The BRDB environment will be operational on four p-blades (slim-line servers) with identical hardware configurations.

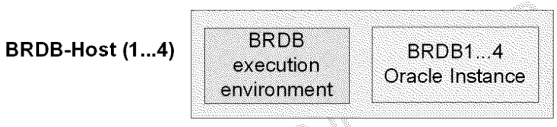



**Figure – 14 Branch Database Execution Environment**



#### 9.1.1.2 Processor

The minimum processor specification for each p-blade is: 4 X x86-based single-core processors with clock speeds of 2.0 GHz.

#### 9.1.1.3 Memory

The minimum memory required for each p-blade is 32GB.

### 9.1.2 Operating System

The Operating system required is Red Hat Enterprise Edition Linux version 4.5 at a suitably stable patchset.

### 9.1.3 Coexistence

The Branch Database Oracle instances and execution environments have exclusive access to resources on each node of the BRDB-Host. Hence co-existence is not relevant.

### 9.1.4 Storage

The storage requirements are defined in the relevant sizing and volumes document [R11].

All storage used by the Branch Database requires tier-1 storage, which must be synchronously replicated to the DR site.

POL00115409



## 9.2 BRDB-Standby Database

### 9.1.1 Hardware Requirements

#### 9.1.1.1 Platform

The Branch Standby Database (BRDB-standby) will use Fujitsu-Siemens BladeFrame environment. The BRDB environment will be operational on a single p-blade (slim-line server) but will require access to four p-blades at the time of setup to create a 4-node cluster.


**Figure – 15 BRDB-Standby Platform Execution Environment**



#### 9.1.1.2 Processor

The minimum processor specification for the p-blade is: 4 X x86-based single-core processors with clock speeds of 2.0 GHz.

#### 9.1.1.3 Memory

The minimum memory required for the p-blade is 32GB.

### 9.1.2 Operating System

The Operating system required is Red Hat Enterprise Edition Linux version 4.5 at a suitably stable patchset.

### 9.1.3 Coexistence

The BRDB-standby database and associated application processes/components will have exclusive access to resources on the BRDB-Standby-Host.

### 9.1.4 Storage

The storage requirements are defined in the relevant sizing and volumes document [R11].

All storage used by the BRDB-standby requires tier-1 storage, which must be synchronously replicated to the DR site.

POL00115409



# 10 Networks

Network aliases named “lprpbdb001…4” (public) & “lprpbdb001…4-vip” (vip) for the Branch Database nodes will be defined in the VPN configuration against the logical names of the clustered platform nodes as per DES/PPS/HLD/0006.

Additionally aliases named “lprpbds001…4” & “lprpbds001…4-vip” for the Branch Standby Database nodes will also be defined.

This network alias will be referred to in all of the Oracle and application specific configuration files.

POL00115409



# 11 Manageability

Standard POA data-centre management tools will be used to monitor the data-centre based subsystems.

Section 5.7 describes the support interface in place to facilitate the Branch Database support management.

POL00115409



# 12 System Qualities

## 12.1 Security

The following sections address potential security issues as highlighted by [R23].

### 12.1.1 Overview

BRDB security is designed in line with **HADDIS [R7]** and the HNG-X Security architecture [R23].

All applications will be run under the batch scheduler through defined Linux users with only the necessary access to run the application. These users will, where necessary, be accessible only locally on the Host, that is they cannot be accessed remotely.

Oracle database connectivity and access control will be through defined Oracle Users and the Roles granted to those Users would only be those that are necessary to run the application.

Data security will be maintained by constraining the database object by type and range and by restricting access through Role to only those applications requiring such access. That is applications that require only read access to tables will only be provided with a role granting 'Select' access and not 'Update' access.

An Oracle user's system and object privileges will be assigned via roles on a least permissions required basis. Note that this precludes running batch processes using the schema owner (i.e. OPS$BRDB).

User names and passwords will be protected across the network by using Oracle encryption functionality, which is the default.

System and DBA access to the Branch Database is by authorised operations staff only.

The Branch Database application creates files containing the day's auditable messages in the directory pointed to by the environment variable $BRDB_COUNTER_AUDIT_OUTPUT. These files contain security sensitive data and adequate operational measures must be taken to prevent unauthorised access to these files and their data.

### 12.1.2 Oracle Level Security

The following security changes will need to be made as a part of BRDB database creation:

#### 12.1.2.1 Lock and Expire Default User Accounts

All non-business Oracle users created as a part of Oracle installation should be reviewed and the ones that are not required for the day-to-day running of the database should be locked.

#### 12.1.2.2 Revoke unnecessary privileges on SYS objects

Privileges on powerful database packages need to be revoked from server user group PUBLIC. Oracle recommends that privileges on UTL_SMTP, UTL_TCP and UTL_HTTP objects should be revoked from the PUBLIC user group.

#### 12.1.2.3 Roles and User Accounts

The Branch Database complies with HADDIS standards ([R7]) by providing named roles with recommended maximum privileges. Assigning unnecessary privileges must be avoided.

POL00115409



Roles will be granted to users only if a user needs all of the privileges that have been assigned to the role.

#### 12.1.2.4 Password Policy Settings

Any Oracle / OS level users that are created as a part of the database build will be locked up on creation. These will be manually unlocked and passwords will be assigned as per the security guidelines set out in [R23].

#### 12.1.2.5 Enabling Data Dictionary Protection

Data dictionary protection will not be enabled as doing so will impact the process of gathering statistics which are a part of the Process Auditing feature referred to in the HADDIS standards.

#### 12.1.2.6 Network Access

##### 12.1.2.1.1 Listener Administration Control

Prevent online administration by requiring the administrator to have write privileges on the LISTENER.ORA file and the listener password:

Add or alter the following line to the LISTENER.ORA file

```
ADMIN_RESTRICTIONS_LISTENER_{node_name}=ON
```

Then RELOAD the configuration.



### 12.1.3 Application Security

Only batch users and support users can have a presence on the Branch Database host.

Users must not have read-write access to the application executables. The support users must not have execute privileges on the application executables and batch users must not have read privileges.

As per HADDIS, all application access to the database must use OS-level authentication with the exception of error logging, where a named user ORAEXCP will be used. This user must have minimum possible privileges.

### 12.1.4 Network Security

The Branch Database will reside in its own DMZ and hence be protected by the firewall from any unauthorized access from the network.

The Branch Access Layer's access to the BRDB nodes will be through Key Management (KMNG), Agents are through Oracle Wallets, which provide password encapsulation and encryption features. Refer to [R21] for further details.

## 12.2 Availability

### 12.2.1 Targets

| Service Level | Max downtime core hours per year | Percentage availability |
|---|---|---|

POL00115409



| | | |
|---|---|---|
| Outages in Core Hours where the Core Solution (Central & Branch Network, Core Infrastructure and Branch Database) is unavailable at > 10% of Branches per SLT year | 1[11] | 99.89568% |
| Outages in Core Hours where the Banking Solution (CAPO, A&L, Link) is unavailable at > 10% of Branches per SLT year. This includes time when the Banking Solution is unavailable because the Core Solution is unavailable. | 1[12] | 99.72181% |
| Outages in Core Hours where Other Services (ETU, DVLA, PAF, APOP, DCS) are unavailable at > 10% of Branches per SLT year. This includes time when the Other Services are unavailable because the Core Solution is unavailable. | 1[13] | 99.51316% |

**Table 5 – Branch Database Availability Targets**

### 12.2.2 Restart/Recoverability

Process Control supports process restart and recoverability. For more details see Section 7.1.4.

The Restart/Recoverability at database/instance level are discussed in Section 12.6.

### 12.2.3 Fail-over

The Branch Database runs on the BladeFrame Linux environment. Resilience and fail-over are implemented using an Oracle Dataguard based standby Branch database and remotely mirrored EMC file store.

For a detailed discussion of Failure scenarios and failover, refer to Sections 12.6 and 12.7.



## 12.3 Usability

There is no direct human interaction with the Branch Database other that what may be required for unplanned maintenance activity.

Any direct user access is strongly discouraged as inappropriate access to data on the branch database will cause the performance to drop significantly that will most likely result in timeouts at the PO Branches thus affecting the Branch trading and potentially causing SLT violations.

## 12.4 Performance

### 12.4.1 Service-specific Volumes & Throughput

This section lists the service-specific volume and throughput figures. The volumes have been used to size the database storage objects and the throughput figures have been considered when designing the structure of the tables and host processes.

[11] SLT value is 3 hours but 2 of those 3 hours have been allocated to Branch Network (1) and Core Infrastructure (1)

[12] Downtime for Core Solution + NPS Database (1) + Firewall Farm (CAPO, A&L, LINK: 1.5) + Service (CAPO, A&L, LINK: 1.5) + N/W Connection (A&L: 1)

[13] Downtime for Core Solution & Banking + Firewall Farm (ETU, DCS, DVLA: 1.5) + Service (ETU, DCS, DVLA, PAF, APOP: 2.5) + N/W Connection (ETU, DCS, DVLA: 3) + APOP (2) + PAF (1)

POL00115409



Only the most performance intensive services have been listed. All figures have been obtained from the HNG-X System Qualities Manual [R4].

### 12.4.1.1 Overall Targets

#### 12.4.1.1.1Throughput[14]

| Design limit (Peak 5 minute TPS) | Allowance for actual per second peak |
|---|---|
| 1068 | 1388 |

**Table 6 – Branch Database Throughput Target**

#### 12.4.1.1.2Volumes

| Design limit (Peak 5 minute TPS) | Allowance for actual per second peak |
|---|---|
| 77,986,779 | 269,932,941 |

**Table 7 – Branch Database Overall Volumes**

### 12.4.1.2 Settlements, EPOSS Events and Audit

| Service | Peak month | Peak week | Peak 2 days | Peak day | Peak hour | Peak 5 min avg TPS |
|---|---|---|---|---|---|---|
| EPOSS | 126,246,451 | 42,383,331 | 18,744,387 | 10,323,021 | 1,476,192 | 413 |
| APS | 48,770,033 | 13,814,266 | 6,751,466 | 3,637,887 | 673,009 | 215 |
| NBS | 41,040,766 | 11,081,007 | 5,289,984 | 3,064,320 | 741,888 | 245 |
| DCS | 5,052,000 | 1,517,721 | 679,139 | 346,110 | 95,242 | 26 |
| ETU | 2,751,844 | 673,699 | 256,540 | 145,440 | 20,705 | 6 |
| DVLA | 4,757,116 | 2,745,745 | 1,354,513 | 885,329 | 122,396 | 36 |
| PAF | 14,363,767 | 4,802,916 | 2,260,397 | 1,320,945 | 194,695 | 55 |
| Settlement | 143,557,723 | 43,335,142 | 20,626,183 | 11,479,010 | 2,076,311 | 638 |
| APOP | 8,200,000 | 2,050,000 | 1,045,000 | 660,000 | 120,000 | 39 |

**Table 8 – Settlement Transaction Throughput & Volumes**

### 12.4.1.3 Reporting Transactions

| Service | Contracted peak 5 min avg TPS | Design limit peak 5 min avg TPS | Live 5 min avg TPS at overall peak period | Live 5 min avg TPS |
|---|---|---|---|---|
| Reports | 100 | 120 | 5 | 40 |

**Table 9 –Reporting Throughput & Volumes**

### 12.4.1.4 Session Management

| Service | Contracted peak 5 min avg TPS | Design limit peak 5 min avg TPS | Live 5 min avg TPS at overall peak period | Live 5 min avg TPS |
|---|---|---|---|---|
| Log On/Off | 100 | 120 | 5 | 48 |

[14] Although the throughput is expressed as number of transactions, since there will be one commit per transaction in the Branch Database (with the exception of recoverable transactions such as Banking or Postal Services, where an exra commit is counted as a separate service). Thus the number of services match the number of commits.

POL00115409



Table 10 – Session Management Throughput & Volumes

#### 12.4.1.5 Recovery Management

| Service | Contracted peak 5 min avg TPS | Design limit peak 5 min avg TPS | Live 5 min avg TPS at overall peak period | Live 5 min avg TPS |
|---|---|---|---|---|
| Banking, Debit Card, ETU & Postal Services? | ? | ? | ? | ? |

Table 11 – Recovery Management Throughput & Volumes

#### 12.4.1.6 Help

| Service | Contracted peak 5 min avg TPS | Design limit peak 5 min avg TPS | Live 5 min avg TPS at overall peak period | Live 5 min avg TPS |
|---|---|---|---|---|
| Help Text | 50 | 60 | N/A | N/A |

Table 12 – Help Service Throughput & Volumes



### 12.4.2 Response Time Targets

As per [R4], the following response time targets exist. All of the following targets are defined as round trip times from the counter:

- Network Banking transactions will take on average 2.5 seconds or less within the total of the HNG-X systems and infrastructure. This is the total time to and from the counter, excluding the time in the banks' infrastructure and systems.
- Settlements will take on average 2 seconds or less.
- No Settlement within the 95th Percentile will take 7 seconds or more.

### 12.4.3 Batch Job Execution Targets

To ensure that the BRDB batch schedule completes within an expected timeframe, the following targets on execution time have been defined for a few BRDB Host processes.

#### 12.4.3.1 Bulk copy to / from Legacy

The following targets are set for processing days when no transactions fail to be copied across the interfaces due to data-type / size / format mismatch errors.

| Interface Heading | Peak Day / 2-Day (Design Limits) | Interface Transfer Time target (elapsed minutes – including all overheads[15]) |
|---|---|---|
| EPOS, AP, NWB, DCS, ETU & Bureau Transactions and EPOSS Events to TPS | Peak day | 60 |
| | Peak 2 days | 90 |

[15] The elapsed time should measure from the time the job/s are invoked by the scheduler to the time the control

POL00115409



| | | |
|---|---|---|
| AP Transactions to APS | Peak day | 20 |
| | Peak 2 days | 30 |
| NWB, DCS & ETU Transactions to DRS | Peak day | 30 |
| | Peak 2 days | 45 |
| Cash, PCOL & PDEL transactions to LFS | Peak day | 15 |
| | Peak 2 days | 20 |
| Track & Trace to NPS | Peak day | Within 5 minutes of each transaction being logged in BRDB during core hours |
| | Peak 2 days | N/A |
| Guaranteed Reversals to NPS | Peak day | Within 3 minutes of each reversal being logged in BRDB during core hours |
| | Peak 2 days | N/A |
| **(Hydra)** EPOS, AP, NWB, DCS, ETU & Bureau Transactions from TPS to BRDB | Peak day | 120 |
| | Peak 2 days | 180 |
| **(Hydra)** 'Ongoing' Reconciliation Totals from TPS to BRDB | Peak day | 30 |
| | Peak 2 days | 60 |
| **(Hydra)** 'In-day' feed from TPS to BRDB | Peak day | 30 |
| | Peak 2 days | NA |



**Table 13 – Bulk Copy Throughputs**

The performance targets being set for batch bulk copy processes is that in the worst peak 2 days scenario, the copy processes should not take more than four hours (240 minutes) to complete.

### 12.4.3.2 Counter Message Audit File Generation

On a daily basis the BRDB Host applications are required to generate files containing auditable messages to be copied across to the Audit Server.

| | Peak Day (design limit) | Peak 2 Days (design limit) |
|---|---|---|
| File Generation Time target (elapsed minutes – including all overheads) | 120 | 180 |

**Table 14 – Audit File Throughputs**

The performance targets being set for audit file generation is that in the worst peak 2 days scenario, the process should not take more than three hours (180 minutes) to complete.

## 12.4.4 System Performance and Tuning

### 12.4.4.1 Instance Tuning

POL00115409



Refer to Appendix E – Suggested Oracle Initialisation Parameters for the full list of suggested Oracle Initialisation Parameters for the Branch Database and the Branch Standby including those required for instance tuning.

Shared pool for instances should not need to be overly big. Insert statements are relatively simple and use host variables so once they are read into memory for the first time there should be no further parses for these statements needed. This leaves memory free for fetches and executions.

#### 12.4.4.2 Database Tuning

Some of the points listed here may have also been covered in relevant sections elsewhere.

Again refer to Appendix E – Suggested Oracle Initialisation Parameters for the full list of suggested Oracle Initialisation Parameters for the Branch Database and the Branch Standby including those required for database tuning.

Size of the redo logs is set to 400M each so that at full load the logs fill every 1-1.5 minutes, which ultimately results in a database checkpoint.

All tablespaces in the database will be locally managed and reside within ASM.



#### 12.4.4.3 Oracle Cost Based Optimisation

As recommended by Oracle, the Oracle Cost Based Optimiser will be used to optimise query performance. This will require statistics to be generated on a daily basis by a scheduler controlled script. Both Schema and Dictionary level statistics will be gathered every day. System statistics will be collected once a week.

Where a BRDB process needs to possibly override the optimiser's execution of a query, hints will be used to direct the optimiser. These hints will be defined and refined during the development process and will be stored in the BRDB_SQL_HINTS table.

Please note that Oracle's implementation of Hints has changed from Oracle9i to Oracle10g. With Oracle10g, the weight associated by the optimiser to a user-supplied hint is much less resulting in a lesser likelihood of the optimiser actually using the hint.

#### 12.4.4.4 Other considerations

Oracle Database Resource Management (DRM) provides the means for managing and controlling database resources so that available resources are appropriately allocated to the business critical tasks. DRM provides the two key features of resource plans and consumer groups.

The Branch Database will not use either of resource plans or consumer groups due to the operational performance overhead that they present.

## 12.5 Backups

This section provides a brief overview of the backup requirements for various components of the Branch Database system. It is essential for the Branch Database to be available for operational use while the backups take place.

Each of the sub-topics in the following sections are discussed in the context of the Branch Database only. The Branch Standby Database backups are discussed in Section 8.3.

POL00115409



### 12.5.1 OCR and Voting Disk

The OCR (Oracle Cluster Registry) and the Voting Disk files are essential for the running of the Branch database cluster. Refer to Section 6.3.1 for a brief description of these files.

The OCR file is fairly static and changes only when amended using Server Control (srvctl) commands. The file resides as a partition in a directory managed by OCFS2 and the size is expected to be less than 100MB. It is sufficient for the OCR file to be backed up on a daily basis alongside the overnight database backup.

The Voting Disk gets written to and read from very frequently however changes are recorded only in the event of a cluster-managed service becomes (un) available. The file resides as a partition in a directory managed by OCFS2 and the size is expected to be around 400MB.

As Oracle only requires a valid Voting Disk file in the correct location regardless of the age of the file for the RAC to work, it is sufficient for the Voting Disk file to be backed up on a daily basis alongside the overnight database backup.

The backups of the OCR & Voting Disk must be retained for 7 calendar days.

### 12.5.2 Configuration Files



#### 12.5.2.1 Initialisation Parameters File

The SPFILE is a binary file containing Oracle initialisation parameters for all four nodes of the cluster. The file contents rarely change and size is size is expected to be less than 10MB. The file resides in a directory managed by OCFS2.

It is sufficient for the SPFILE file to be backed up on a daily basis alongside the overnight database backup. Retention period for the purpose of recovery must be a minimum of 7 calendar days.

#### 12.5.2.2 Local Naming (TNSNAMES)

The TNSNAMES file is a text file containing aliases for all services / instances that the Branch Database node connects to. File contents rarely change and size is a few KB. The file resides in the RHEL filesystem formatted directory $ORACLE_HOME/network/admin.

It is sufficient for the TNSNAMES file to be backed up on a daily basis alongside the overnight database backup. Retention period for the purpose of recovery must be a minimum of 7 calendar days.

#### 12.5.2.3 Listener Configuration

File type, location and backup retention period are same as for TNSNAMES (above).

### 12.5.3 Database Backup

This section covers the backup of database control files, online redo log files, database data files and archived redo logs.

The use of ASM to manage all data files mandates the use RMAN for performing backups and recovery.

The overall approach is to perform Level-n (full & incremental) hot backups onto ASM diskgroups at both data centres for resilience.

The reason incremental backups have been chosen is that the size of the database is large (~400GB) and a full hot backup might take too much time. As the retention period of the largest tables in Branch database is set to 3 days, it can be said that the database receives roughly a third of its data volumes

POL00115409



every day. A Level-1 incremental backup will reduce the size of the backup set by a third but will also mean that recovery will take a proportionally longer time as for recovery, the most recent Level-0 backup will need to be restored and the incremental Level-1 backups applied on top.

Hence the backup approach will take a middle position of taking a full Level-0 backup twice a week (Sunday and Wednesday) and Level-1 (incremental) backup on remaining days.

The following figure presents the daily backup schedule for the Branch Database RMAN backups. The term L0 is used synonymous with Level-0 and L1 is used alongside Level-1.

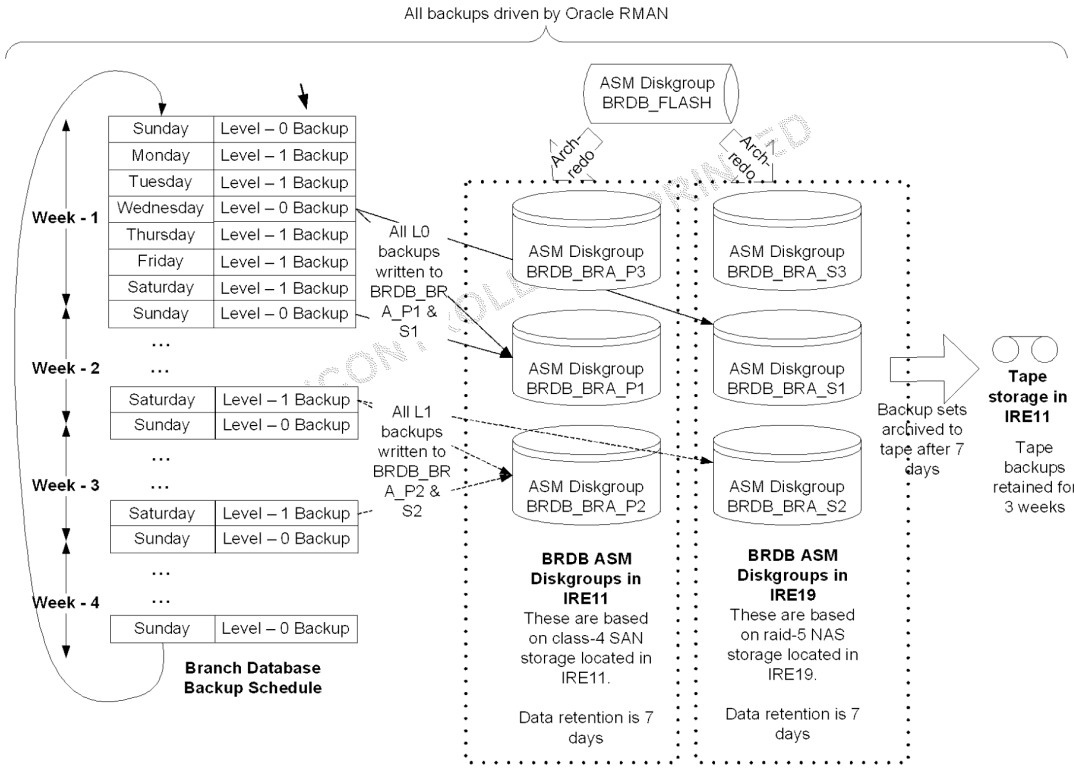



**Figure – 16 BRDB Backup Approach**

The backup approach can be summarised as follows:

- Level-0 backups will be taken on all Sundays and Wednesdays. Backups will include control files and all non-temporary data files. The target locations are ASM diskgroup BRDB_BRA_P1, which is based on SAN storage and located in the DMX at IRE11 and diskgroup BRDB_BRA_S1, which is based on NAS storage that is physically located in the DMX array at IRE19 but accessible the Branch Database nodes in IRE11.
- Level-1 incremental backups will be taken on all other days of the week. Backups will include control files and all non-temporary data files. The target locations are ASM diskgroup BRDB_BRA_P2, which is based on SAN storage and located in the DMX at IRE11 and diskgroup BRDB_BRA_S2, which is based on NAS storage that is physically located in the DMX array at IRE19 but accessible the Branch Database nodes in IRE11.

POL00115409



- All archived redo log files created since the last L0/L1 backup will be backed up. The source is ASM diskgroup BRDB_FLASH and the target locations are ASM diskgroup BRDB_BRA_P3, which is based on SAN storage and located in the DMX at IRE11 and diskgroup BRDB_BRA_S3, which is based on NAS storage that is physically located in the DMX array at IRE19 but accessible the Branch Database nodes in IRE11.
- In all of the above cases, the backup sets written to IRE19 can be initiated using a separate RMAN backup session if it presents a more efficient approach. The Backup High Level Design [R35] should specify the approach to be taken in this regard.
- Backups need to be taken as 'backup-sets' (as opposed to image copies), should be compressed as directed by the Backup design [R35].
- An RMAN catalog should be used to manage the backups. Creation of the catalog is outside the scope of this design document. Refer to the Backup design [R35] for further details on the RMAN catalog.
- Naming conventions for backup sets should be as per standard Oracle naming conventions unless specified by the Backup design [R35].
- RMAN will check for physical block corruptions by default. RMAN should be used to check for logical block corruptions as a part of the backup operation.
- The Level-1 backups should take no more than 2 hours under normal operational circumstances. Level-0 backups should take no more than 3 hours. Means for improving efficiency such as parallelising by use of multiple backup channels should be explored and used if applicable.
- Backup sets should be retained on disk for a minimum of 7 calendar days. After that they should be archived off to tape where they will be retained for a further 3 weeks period. Either of the backup diskgroup sets could be used to archive off to tape and the most network efficient means should be chosen. The tape infrastructure and the procedures used should be as per standard RMGA practice for HNG-X.

POL00115409



## 12.6 Failure & Restart/Recovery

This section lists the various Oracle failure scenarios in detail and discusses the action to be taken in response to each different type of failure.

POL00115409



## 12.6.1 Problem Scenarios & Recovery Options – Summary

| Problem Type | Problem Detail | Recovery Action Details |
|---|---|---|
| Instance / Node Failure | A single Branch Database instance crashes | CRS ability of automatically restarting the failed instance will be disabled. The instance will be 'logically' removed from the list of Branch Database instances that are available for handling Branch requests. This happens automatically by Oracle RAC-FAN utility (BRDBX010).<br>The failed instance will be automatically added into the list of 'available' instances at end of business day. There is an option of manually making the instance 'available' before that and the steps involved in doing that need to be discussed in the support guide.<br>Refer to Section 12.6.2 for further discussion on this topic. |
| | A single Branch Database Node crashes & restarts | The failure notification will occur via ITM Tivoli agent and via Grid Control (from Oracle perspective).<br>Bladeframe technology will attempt to automatically restart the failed 'pserver' on a different physical blade in the LPAN. Once RHEL4 initialises in the new blade, the BRDB database instance will automatically start via oratab.<br>As with single database instance failure, the instance failed node will not be used until end of day. |
| | A single Branch Database instance crashes but fails to restart | This is not a valid failure scenario, as the Branch Database instances are not restarted on failure. This problem option has been included for completeness. |
| | A single Branch Database Node crashes but fails to restart | The failure notification will occur via ITM Tivoli agent and via Grid Control (from Oracle perspective).<br>If Bladeframe cannot automatically restart the failed 'pserver' on a different physical blade in the LPAN, it will flag an error via the PAN manager and support can follow it up.<br>As with single database instance failure, the BAL / Branch Database host will not attempt to use the database instance that was running before the node failed until end of day or when the node starts, whichever occurs later. |
| | Two or more instances of the Branch Database crash | Automatic restart by CRS has been disabled.<br>The Oracle RAC-FAN utility (BRDBX010) will have automatically 'voted-out' the failed instances logically.<br>Instances need to be manually restarted. Once the instances restart and if no further problems are encountered, support should manually make the instances logically 'available' to accept counter requests, as per instructions in the Branch Database support guide.<br>However if the instances that have come back up present further problems e.g. network issues, then the instances should be treated as non-restartable and the recovery action for that failure scenario should be followed. |

POL00115409



| | | |
|---|---|---|
| | Two or more Branch Database Nodes crash & restart | The failure notification will occur via ITM Tivoli agent and via Grid Control (from Oracle perspective).<br>Bladeframe technology will attempt to automatically restart the failed pservers on the same / different physical blade in the LPAN as defined by the LPAN configuration.<br>If the pserver can be restarted and RHEL4 can start without any issues, the recovery actions for Branch Database instance crash & successful restart (see above) should be followed.<br>If the pserver / RHEL4 cannot be restarted, the recovery actions for node failure of two or more nodes with no subsequent restart (see below) need to be followed. |
| | Two or more instances of the Branch Database crash but fail to restart | This scenario is a continuation from the previous problem.<br>A manual restart of the failed instances does not resolve the issue, the recommended resolution option should be failover to BRDB-Standby. If the failure occurs during core business hours, it is more advantageous to attempt a failover. Whereas if the problem occurs outside of core business hours and at least two branch database nodes are available, these may be sufficient to support the reduced business traffic. In such cases it may be beneficial to carry on using BRDB to support the reduced business while attempting to fix the problem and restart the failed nodes before start of core business hours.<br>Any failover to BRDB-Standby should always be manually initiated and after consultation with Service-delivery as will be outlined in the Support Guide.<br>Refer to Section 8.4.2 for detailed discussion on the failover to BRDB-Standby. |
| | Two or mode Branch Database nodes crash but fail to restart | If the Bladeframe PAN manager cannot automatically restart the failed blades, either on the same or alternate pblades, an error will be flagged and support will need to investigate.<br>The recommended resolution action is for the Branch Database to switch to BRDB-Standby platform. Failover will be manually initiated and will need to follow the steps outlined in Section 8.4.2. |
| Media Failure | Alert-logs, RMAN backup (check logical), dbv, Grid Control detects a corrupt block | If the block corruption results in a node / database failure, as may be expected if the corruption affects the data dictionary in the system tablespace or the online-redo logs, the resolution options discussed as a part of Instance failure (above) should be followed.<br>If the block corruption does not result in a node / instance failure, block recovery using RMAN "blockrecover" command should be attempted. Block-recovery will make use of the backup sets to recover the last known image of the block and any changes made using the archived and online redo logs. |
| | The monitoring / verification tools detect a data file corruption | As discussed for block corruption above, if the file corruption results in database instance/s failure, the resolution options discussed as a part of Instance failure (above) should be followed.<br>If the file corruption does not result in a node / instance failure, the corrupt file and its contents can be recovered by standard RMAN recovery options. |
| | The monitoring / verification tools detect a data file corruption along with corruption of online / archived redo log files | The Branch Database writes archived redo log files to two different locations and each location resides in a separate EMC DMX. This makes the possibility of redo-log corruptions due to hardware issues unlikely. There is relatively a greater chance of redo logs being corrupt due to a logical corruption in the Oracle / EMC buffers. However such corruptions will be detected quickly by the Data Guard physical standby replication. Hence the potential of data loss in the event of such a corruption is low.<br>As discussed before, if such a corruption results in database instance/s failure, failover to BRDB-Standby should be considered. If database instances do not fail, RMAN-based database point-in-time recovery should be performed.<br>Refer to Section 8.4.2 for detailed discussion on the failover to BRDB-Standby. |

POL00115409



| | | |
|---|---|---|
| | The monitoring / verification tools detect a corruption of the database control file | This will most likely result in a database failure, as the corruption will be replicated across all copies of the control file.<br>Although recovery is a straightforward[16] process, it is recommended that the Branch Database service switch to BRDB-Standby. |
| | The Voting / OCR disks get corrupted | This will most likely result in a database failure along with all other Oracle cluster services. These files are small in size and change infrequently hence are backed up daily. They should be restored and the cluster should be restarted. If the cluster fails to start, a switch to standby should be considered. |
| Network Problems | Network slows down which results in Counter requests timing out | There are two options available: Stay on the BRDB cluster while the problem is resolved or switch to BRDB-Standby. The decision to do either should be made in consultation[17] with Linux Support & CS. |
| User Errors | User inadvertently drops a table | The dropped table can be recovered using standard RMAN backup recovery procedures. RMAN will need access to the most recent Level-0 backup and all subsequent Level-1 backups along with copies of archived redo log files since the Level-0 backup.<br>Refer to section 12.5 for details on the location and retention period of the various backup components. |

**Problem Scenarios & Recovery Options – Summary**



[16] Recovery steps are: restore a previous backup of the control file and apply any file-level changes made since then and manually add any temporary files i.e. temporary tablespaces. Perform full RMAN database recovery to bring the restored control file in line with the change numbers in the data files.

[17] As there are a number of variables such as extent of the network issue, time when the problem occurs etc, design cannot provide any meaningful input into the decision-making.

POL00115409



## 12.6.2 Instance Failure

An *Instance failure* occurs when software or hardware problems disable an instance. After instance failure, the CRS daemon automatically attempts to restart the failed instance. This facility needs to be disabled.

The reason for doing so is that if an instance fails in a RAC environment, the SMON process in available instances will automatically perform instance recovery on behalf of the failed instance. The combination of Oracle FAN (see Section 7.2.10) and a connection re-attempt by the Branch Access Layer will quickly redirect traffic to the available instances through the use of the ***brdb_operational_instances*** mapping table. If the failed instance was restarted soon afterwards, that instance would not get any traffic going to it at all. In such cases Oracle attempts a very basic form of load balancing such as transferring ownership of 1/4$^{th}$ of the blocks in use in the cluster to the restarted instance which will result in performance degradation as has already been proven in prototyping.

The Branch DB architecture [R3] suggests that a BRDB cluster running 3 nodes will be able to handle the Branch traffic without any drop in performance. Hence the failed node will remain down until a restart is manually initiated by support at non-busy time in the processing day.



## 12.6.3 Media Failure

A *media failure* is a physical problem that arises when Oracle tries to write or read a file that is required to operate the database. An example is disk head crash causing loss of all data on a disk.

All Branch database components essential for the operation of the database will reside on SRDF replicated Raid-1 disks. SRDF will ensure that every byte saved to the disk on the local (primary) site is synchronously replicated and saved to the secondary (DR) site before a positive response is returned back to the OS or disk management software such as ASM.

RAID-1 technology ensures that every byte of data written is mirrored on an erstwhile location in the disk array so even if a disk in the disk array were to fail, there will be no data loss.

The combination of SRDF and RAID-1 ensure that the possibility of media failure is greatly minimised.

The Oracle database will be running in archivelog mode and the archived redo logs are copied across to multiple locations so even if a media failure were to occur, provided the archived redo logs are accessible, a straightforward database point in time recovery will ensure that there is no data loss.

## 12.6.4 Block Corruptions

A block corruption is said to occur when Oracle detects that the block wrapper (block header & footer) of one or more blocks is corrupt/invalid.

Once Oracle detects a corrupt block, it writes the following error message to the alert log for the instance that detects the corruption:

> ORA-01578: ORACLE data block corrupted (file # %s, block # %s)
>
> Where file# is the file ID of the Oracle datafile and block# is the block number, in Oracle blocks, within that file.

Any potential downtime is dependent on how quickly block corruptions are detected.

POL00115409



### 12.6.4.1 Detecting Block Corruptions

An Oracle error #1578 does not always mean that the block on disk is truly physically corrupt as the error message might just be indicating that there is a corruption in the cache. A number of means are available to verify that the block on disk is corrupted (either physically or logically):

**Block Corruption Detection mechanisms to be used**

- DB_BLOCK_CHECKSUM parameter

  DB_BLOCK_CHECKSUM determines whether DBW*n* and the direct loader will calculate a checksum (a number calculated from all the bytes stored in the block) and store it in the cache header of every data block when writing it to disk. This allows Oracle to detect block corruptions whenever the block is read. If the block is corrupted, error messages are written to the alert log but processes may not necessary fail.

  This parameter will be set to TRUE (default) for the Branch Database.

- RMAN

  RMAN can detect both logical and physical corruptions when performing backups by using the "BACKUP VALIDATE" command:

  ```
  BACKUP VALIDATE CHECK LOGICAL DATABASE ARCHIVELOG ALL;
  ```

  Each new corrupt block is recorded in the control file and in the alert-log. The Oracle view V$DATABASE_BLOCK_CORRUPTION provides a list of all corrupt blocks and indicates whether the corruption is logical or physical.

  RMAN will be used to detect logical and physical block corruptions through the entire Branch Database.

- Oracle Alert-Log

  Oracle records any block corruption errors that it detects in the alert log. Since corruptions may not necessarily cause the Instance or processes to fail, these errors may go un-noticed. However the HNG-X Solution provides for a centralised Enterprise Manager Grid Control based Monitoring setup, which will be used to monitor errors in the Alert log. Refer to

- DB_BLOCK_CHECKING parameter

  Setting this initialisation parameter to TRUE causes Oracle to proactively check for corruptions and hence aid early detection. There is an overhead of between 1% and 10% associated with this, which is unacceptable for the Branch Database. If block corruptions occur more frequently this parameter can be reviewed.

  The Branch Database will be delivered with this parameter value set as FALSE.

- Corruption Detection SQL & DB-Verify

  Corruptions can be detected at object-level by using the SQL "ANALYZE TABLE…VALIDATE STRUCTURE…" and the db-verify utility to detect corruptions in data files.

  Since the Branch Database will use RMAN to detect logical corruptions, the corruption detection SQL and DB-Verify need not be used for Branch Database.

The Branch Database will make use of Oracle Dataguard to maintain the BRDB-standby database. Dataguard transparently detects corruptions and ensures that the corrupted blocks are not replicated to the standby database. Corruptions detected can be flagged to support via EM Grid Control.

POL00115409



#### 12.6.1.2 Recovering from Block Corruptions

Once block corruption has been detected, the key task is to recover lost data. There are four options available for recovering data in the Branch Database:

- Use application-specific means of restoring data.
- Switch to using the BRDB-standby database
- Restore from backups and recover using the archived redo-logs
- Restore from Oracle export files

Oracle block corruption will occur on all mirrored copies of the disk, both locally and at the remote DR site, so failing over to DR is not an appropriate recovery method.

Index block corruptions are best resolved by rebuilding the index. There may be a performance impact of rebuilding and recommendations about time of rebuilding have been made in the later sections.

The appropriate recovery option to choose will depend on the type and criticality of the object being recovered and also whether the corrupted information has been consumed by all consumer systems. The following section categorises the Branch Database objects and discusses recovery options for each category:



## 12.7 Switch to BRDB-Standby

For detailed discussion on this topic, refer to Section 8.4.

## 12.8 Potential for Change

See the Changes Expected section 0.7.

POL00115409



# 13 Implementation

## 13.1 Summary

The Branch Database environment will consist of a number of components. Some of these will need to be in place well in advance of the beginning of pilot. This will allow enough time for configuring the storage and network components and testing of the same.

## 13.2 Delivery Units

Branch Database system delivery will be divided into a number of logically related delivery units. Each delivery unit may be further sub divided into one or more work packages. Work packages for subsequent delivery units should follow work packages for preceding delivery units.

**Delivery Unit 1** will contain the scripts/instructions to create ASM instances named "+ASM1…4" on the new BRDB-Host (x4). It will also create the ASM disk groups (as defined in section 6.2.3). It will also include the configuration changes to allow the ASM instance to accept connections across TCP/IP.

Work packages associated with this delivery unit will typically be applied to the live estate well in advance of the pilot.

**Delivery Unit 2** will contain the scripts/instructions to create all relevant Linux users, profiles, environment variables and the required OS directories. This delivery unit will also contain the scripts to create a 4-node clustered database for BRDB (non-clustered for test) along with tablespace definitions. Additionally the delivery unit will contain the Oracle network configuration changes to allow the BRDB instances to accept connections across TCP/IP.



Refer to sections 6.3.3 through to 6.3.8, 6.4 and 6.5 for more details.

Work packages associated with this delivery unit will typically be applied to the live estate well in advance of the pilot but will follow **Delivery Unit 1**.

**Delivery Unit 3** will provide the scripts to build the BRDB application schema objects such as tables, views etc and PL/SQL based application deliverables. It will also deliver metadata, static reference data and other related deliverables.

Additionally this delivery unit will define the rules for data capture and propagation to the BRSS using Streams and the configuration changes for setting up a database level data capture and propagation to BRDB-standby via DataGuard.

This delivery unit will also provide the application executables, shell scripts and execution environment for the Branch Database host processes. Refer to Section 6.3.10, 6.3.11 and 6.4.3 for more details.

Work packages associated with this delivery unit will typically be applied to the live estate in advance of the pilot but will follow **Delivery Unit 2**.

**Delivery Unit 4** will provide the TWS execution environment and the BRDB-Host schedule definition for running the Branch Database batch schedule.

Refer to [R12] for more details on BRDB job scheduling.

Work packages associated with this delivery unit should be applied to the live estate in advance of the pilot but the schedule should remain deactivated.

**Delivery Unit 5** will create the BRDB-Standby environment including the ASM instance, the clustered single-node database along with the necessary configuration changes for running the BRDB-standby database in physical standby mode.

POL00115409



**Delivery Unit 6** will prime the database objects and metadata for the first day of the BRDB-Host batch schedule. This includes creating partitions for days D and D + 1, prepping the database with up to date reference data and applying any changes to the Branches – Fad-Hash mappings.

Work packages associated with this delivery unit will be applied to the live estate on the day prior to running the TWS schedule (see Delivery Unit 4).

Note that the enabling of data capture and propagation via Streams to the BRSS and via DataGuard to BRDB-Standby should be done via additional work packages.

POL00115409



# 14 Application Development

All application development must follow the guidelines set out in [R7].

As a minimum, the following low-level design documents will be produced. It is acceptable to produce more low-level designs documents if needed for improving clarity and for a more detailed coverage.

- Database and Schema Build (including setting up the ASM disk groups)
- Start of Day (Application BRDBC001)
- Audit-Store Auditing (BRDBC002)
- Common Legacy Host Interface (BRDBX003)
- Audit, Archive & DB-copy (Application BRDBC004)
- Gathering Optimiser Statistics (BRDBX005)
- File Housekeeping (BRDBX006)
- Data Aggregation Tool (BRDBX007)
- Check Job Completion (BRDBC008)
- End Of Day (BRDBC009)
- Detect and Report Node Failures using FAN (BRDBX010 Fan event)
- Update System Parameters (BRDBX011)
- SSC Transaction Correction Tool (BRDBX015)
- BRDB File Transfer to DWh (BRDBX020)
- Pause / Start Streams Replication (BRDBX021)
- Maintain JSN SSN USN (BRDBX031)
- Hydra specific XML Data Processor (BRDBX030 & BRDBX033 & BRDBX034)

POL00115409



# 15 Testing and Validation

This section lists some of the important areas to focus on in the Branch Database system from a test perspective. There is no distinction made in the areas to focus on for different types of testing. It is left to the test designer to decide upon which test environment is best suited for each focus area.

**Application Components**

- Ensure that the BRDB Start-of-Day process creates new partitions one day in advance for all transactional and event tables. Also ensure that old partitions are being housekept in order to conserve space.
- The BRDB feed for Audit Server must contain all auditable messages logged by the BAL in the BRDB during a calendar day on behalf of the Counter.

**Integration**

- Ensure that the Branch Access Layer (BAL) can access all four nodes of the Branch Database (BRDB).
- The BAL access of BRDB for outlet specific data should always be a 'Fad-Hash' aware access. Refer to section 5.1 for accessing data in a Fad-Hash aware manner.
- If a Branch Database node fails, the BAL should use an alternate (metadata-defined) node for accessing the database. BRDB node failure should not result in a transaction failure at the Counter.
- Ensure that all BRDB – Batch application batch data feeds can restart from logical start point prior to the failure such that they minimise the re-work required on a restart. With the exception of erroneous records, which should be accounted for as exceptions, there should be no data loss in the dataset that reaches the target database.



**Performance & Volume**

- Peak transaction throughputs must be tested using an adequately simulated test environment.
- BRDB must be able to notify of node failures to all consumer applications by changing the Fad-Hash mapping metadata.
- The audit-store audit file generation needs to complete within the time limits specified.

**Business Continuity Testing**

- Data replication to the Branch Support System must not lose data.
- Data replication to the Branch Standby Database must not lose data.
- Database backups (both level – 0 and level – 1) must be adequately tested for data integrity and backup performance.
- Recovery of BRDB using database backups must be tested.
- Block corruptions must be simulated where possible and the actions to be taken in the event of block corruptions such as switchover to the BRDB-standby database must be tested. Refer to Section 12.6.4 for discussions around the different types of data affected by block corruptions and the recovery actions to be taken for each type.

POL00115409



# 16 Risks and Assumptions

Risks and assumptions have been covered in section 1.4.

POL00115409



# 17 Requirements Traceability

The SRS for this topic architecture captures incoming requirements. The requirements system has a skeleton copy of the document (section numbers only). The skeleton document identifies the section where the requirement is being addressed.

Section will be updated to reference the document once it is published.

POL00115409



# 18 Appendix A – Table and Index Definitions

Table and index definitions for Branch Database will be in [R36]:

## 18.1 Live Schema objects shared with Training

The following tables will reside in the “Live” schema but will be shared with the “Training” schema by the use of public synonyms:

| |
|---|
| BRDB_ANALYZED_OBJECTS |
| BRDB_ARCHIVED_TABLES |
| BRDB_BRANCH_ROLES |
| BRDB_BRANCH_ROLE_SERVICES |
| BRDB_EXCEPTION_CODES |
| BRDB_FAD_HASH_INSTANCE_MAPPING |
| BRDB_FAD_HASH_OUTLET_MAPPING |
| BRDB_FAD_HASH_VALUES |
| BRDB_FILES_TO_HOUSEKEEP |
| BRDB_HOST_INTERFACES |
| BRDB_LOCAL_APP_INTERESTS |
| BRDB_ORACLE_ERROR_CODES |
| BRDB_PARTITIONED_INDEXES |
| BRDB_PARTITIONED_TABLES |
| BRDB_PARTITION_CREATES |
| BRDB_PARTITION_STATUS_HISTORY |
| BRDB_PROCESSES |
| BRDB_PROCESS_AUDIT |
| BRDB_PROCESS_CONTROL |
| BRDB_SQL_HINTS |
| BRDB_SUBPARTITION_RANGES |
| BRDB_SYSTEM_PARAMETERS |
| BRDB_TABLE_GROUPS |
| BRDB_TABLE_PARTITIONS |
| RDDS_ACCOUNTING_NODES |
| RDDS_BRANCH_OPENING_PERIODS |
| RDDS_DELIVERY |
| RDDS_DELIVERY_TYPE |
| RDDS_DESKTOP_MEMO |
| RDDS_DESKTOP_MEMO_DISTR |
| RDDS_PACKAGE |
| RDDS_PACKAGE_CONTENT |
| RDDS_PACKAGE_TYPE |
| RDDS_PRODUCTS |

POL00115409



# 19 Appendix B – View Definitions

View definitions for Branch Database will be in [R36]

POL00115409



# 20 Appendix C – User, Role Sequence Definitions and Database Links

## 20.1 Linux User Definitions

| User Name | Groups | Description |
|---|---|---|
| brdb | pathway | User owns the Branch Database "Live" schema, application executables and the business data on file-system |
| brdbtr | pathway | User owns the Branch Database training schema, application executables and the business data on file-system |
| supporttooluser | pathway | Use for the BRDBX015 transaction support tool |
| oracle | dba, oinstall | Owner of the Oracle installation on the Branch Database Nodes 1…4 |
| brdbblv1 | pathway | Users used to run batch jobs on the BRDB-Hosts. Wherever BRDB is accessed, the users connect to the **Live** BRDB schema. |
| brdbblv2 | | |
| brdbblv3 | | |
| brdbblv4 | | |
| rep_gen | pathway | Generic reports user |



## 20.2 Oracle User Definitions

### 20.2.1 Schema Owner

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| OPS$BRDB | USERS | BRDB_TEMP1 | CONNECT, RESOURCE<br>*Role DBA granted only for the duration of execution of the schema build scripts.*<br>EMDB_USERS<br>OMDB_USERS<br>TWS_USERS<br>*Above 3 with grant option* | SELECT ANY DICTIONARY,<br>CREATE_TABLE,<br>CREATE_VIEW | User owns all Branch Database live schema objects. |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| OPS$BRDBTR | USERS | BRDB_TEMP2 | CONNECT,<br>RESOURCE*Role DBA granted only for the duration of execution of the schema build scripts.* | SELECT ANY DICTIONARY,<br>SELECT , DELETE On ops$brdb.brdb_branch_users, brdb_branch_user_roles, brdb_branch_user_sessions, brdb_branch_user_last_logon, brdb_password_history.<br>Update on ops$brdb.brdb_branch_users.<br>Select on ops$brdb.brdb_branch_info. | User owns all Branch Database training schema objects. Due to SQL92_SECURITY being set then any object owned by a different user, which this is user has DELETE privilege also needs SELECT privileges. |

## 20.2.2 Branch Access Layer



| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| LVBALUSER | USERS | BRDB_TEMP1 | BALLVROLE | | Used by the Branch Access Layer to connect to the live schema in the branch database |
| LVBALUSER1 | USERS | BRDB_TEMP1 | | | |
| LVBALUSER2 | USERS | BRDB_TEMP2 | | | |
| LVBALUSER3 | USERS | BRDB_TEMP3 | | | |
| LVBALUSER4 | USERS | BRDB_TEMP4 | | | |
| TRBALUSER | USERS | BRDB_TEMP1 | BALTRROLE, DELTRROLE | | Used by the Branch Access Layer to connect to the training schema in the branch database |
| TRBALUSER1 | USERS | BRDB_TEMP1 | | | |
| TRBALUSER2 | USERS | BRDB_TEMP2 | | | |
| TRBALUSER3 | USERS | BRDB_TEMP3 | | | |
| TRBALUSER4 | USERS | BRDB_TEMP4 | | | |
| DELTRUSER | USERS | BRDB_TEMP | DELTRROLE | | Used by the Branch Access Layer to delete training in the branch database |

## 20.2.3 Batch Processes

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| OPS$BRDBBLV1 | USERS | BRDB_TEMP1 | BATCHLVROLE | | Used by BRDB host-based batch processes to connect to the **Live** schema |
| OPS$BRDBBLV2 | USERS | BRDB_TEMP2 | | | |
| OPS$BRDBBLV3 | USERS | BRDB_TEMP3 | | | |
| OPS$BRDBBLV4 | USERS | BRDB_TEMP4 | | | |
| | | | | | |
| | | | | | |
| | | | | | |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| | | | | | |

## 20.2.4 Performance Monitoring

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| OPS$METRON | USERS | BRDB_TEMP1 | CONNECT | SELECT ANY DICTIONARY | Used by the Metron performance monitoring tool to gather database statistics on the **Live** schema only. |
| | | | | | |
| | | | | | |
| | | | | | |

## 20.2.5 Application Exceptions

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| ORAEXCPLV | USERS | BRDB_TEMP4 | BRDB_EXCEPTIONSLV | | Used to make concurrent / asynchronous connections to the Oracle database to log operational exceptions from within batch processes.<br>Connect to the **Live** schema. |
| | | | | | |
| APP_MONITOR_USER | USERS | BRDB_TEMP1 | APP_MONITOR<br>CONNECT | Select on ops$brdb.brdb_exception_codes, ops$brdb.brdb_operational_exceptions | User is used by the ITM and Oracle Application Express to connect to the BRDB database and fetch details about operational exceptions in the **Live** schema. |



## 20.2.6 First Line Support

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| User-specific. | USERS | BRDB_TEMP2 | DB_MONITOR<br>Optionally UNXADM but always done manually. | | Used by the Helpdesk/ISD (1st line) users |

## 20.2.7 Second Line Support

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| User-specific. | USERS | BRDB_TEMP2 | SMC | | Used by the SMC (2nd line) users |

## 20.2.8 Third Line Support

POL00115409



| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| User-specific. | BRDB_SSC_DATA | BRDB_TEMP2 | SSC<br>Optionally APPSUP but always done manually. | EXECUTE privilege on<br>pkg_brdb_txn_correction<br>SELECT on ops$brdb.brdb_branch_info | Used by the SSC (3rd line) users |

## 20.2.9 Audit Access

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| AUDITUSER | USERS | BRDB_TEMP3 | AUDITOR | | Used by the Audit users to query BRDB for Fad-Hash mappings and running relevant audit-specific queries on the **Live** Schema. |



## 20.2.10 Support Correction Tool Owner

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| | | | | BRDB_TXN_CORR_TOOL_CTL<br>BRDB_TXN_CORR_TOOL_JOURNAL<br>BRDB_PROCESS_AUDIT<br>BRDB_OPERATIONAL_EXCEPTIONS<br><br>BRDB_RX_APS_TRANSACTIONS<br><br>BRDB_RX_BUREAU_TRANSACTIONS<br>BRDB_RX_EPOSS_EVENTS<br>BRDB_RX_EPOSS_TRANSACTIONS<br>BRDB_RX_NWB_TRANSACTIONS<br>BRDB_RX_REP_SESSION_DATA<br>BRDB_RX_DCS_TRANSACTIONS<br><br>BRDB_RX_REP_EVENT_DATA<br><br>BRDB_RX_CUT_OFF_SUMMARIES | |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| | | | | BRDB_SYSTEM_PARAMETERS<br>BRDB_FAD_HASH_OUTLET_MAPPING<br>BRDB_FAD_HASH_CURRENT_INSTANCE<br>BRDB_BRANCH_INFO<br>BRDB_PROC_AUD_SEQ<br>BRDB_OPER_EXCP_SEQ<br><br>Execute privilege on<br>PKG_BRDB_COMMON<br>PKG_BRDB_EXCEPTION<br>PKG_BRDB_FEED_COMMON<br>SYS.UTL_FILE<br><br>Read privilege on<br>BRDBX015_DIR | |
| | | | | | |



## 20.2.11 Software Administrator Accounts

POL00115409



| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| STRADMIN | BRDB_STREAMS_DATA | BRDB_TEMP4 | CONNECT<br>RESOURCE<br>DBA<br>SELECT_CATALOG_ROLE | ALTER ANY RULE<br>ALTER ANY RULE SET<br>CREATE ANY RULE<br>CREATE ANY RULE SET<br>CREATE EVALUATION CONTEXT<br>CREATE RULE<br>CREATE RULE SET<br>DEQUEUE ANY QUEUE<br>ENQUEUE ANY QUEUE<br>EXECUTE ANY RULE<br>EXECUTE ANY RULE SET<br>MANAGE ANY QUEUE<br>RESTRICTED SESSION<br>SELECT ANY DICTIONARY<br>UNLIMITED TABLESPACE | Administrator account for managing the streams capture and propagate functions. |
| | | | | | |



## 20.2.12 Batch Applications Interface Users

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| BRDBRDDS | USERS | BRDB_TEMP1 | RDDS_USERS | | User used by the RDDS interface to copy data out of the Branch Database |
| BRDBRDMC | USERS | BRDB_TEMP2 | RDMC_USERS | | User used by the RDMC interface to copy data out of the Branch Database |
| EMDB_SUP | USERS | BRDB_TEMP2 | EMDB_USERS | SELECT on ops$brdb.brdb_branch_info.<br>ops$brdb.brdb_branch_node_nfo.<br>ops$brdb.rdds_branch_opening_periods<br>CREATE VIEW | User used by EMDB to maintain EMDB tables for BRDB Estate Management |

POL00115409



| | | | | | |
|---|---|---|---|---|---|
| OMDBUSER | USERS | BRDB_TEMP4 | OMDB_USERS | | User is used by the OMDB (SYSMAN) to read PONR for HNG-X Branch Migration |

## 20.2.13Tivoli Workload Scheduler

| User Name | Default Tablespace | Temp Tablespace | Roles Granted | Privs Granted | Description |
|---|---|---|---|---|---|
| TWS | USERS | BRDB_TEMP1 | TWS_USERS | | User is used by the TWS Scheduling solution |

## 20.3 Oracle Role Definitions

| Role Name | Roles Granted | Object & System Privileges Granted | Description |
|---|---|---|---|
| | | | |
| BRDB_EXCEPTIONSLV | CONNECT | SELECT, UPDATE, INSERT ON **ops$brdb.brdb_operational_exceptions**<br>SELECT ON **ops$brdb.brdb_exception_codes**<br>**SELECT ON**<br>**ops$brdb. brdb_oper_excp_seq**<br>**SELECT, UPDATE ON ops$brdb. brdb_operational_instances** | Branch Database Exception Recording Role (for **Live** schema) |
| BALLVROLE | CONNECT<br>SELECT_CATALOG_ROLE | SELECT on all OPS$BRDB sequences.<br>SELECT on all non-working OPS$BRDB tables and views<br>INSERT, UPDATE on all non-working OPS$BRDB tables<br>DELETE on brdb_branch_decl_item, brdb_ps_barcodes | Role assigned to all users used by the Branch Access Layer to connect to the **live** schema |
| BALTRROLE | CONNECT<br>SELECT_CATALOG_ROLE | SELECT, INSERT, UPDATE ON ALL TABLES OWNED BY OPS$BRDBTR<br>SELECT ON ALL SEQUENCES OWNED BY OPS$BRDBTR<br>DELETE on ops$brdbtr.brdb_branch_decl_item,<br>ops$brdbtr.brdb_ps_barcodes<br>SELECT ANY SEQUENCE<br>**INSERT,UPDATE on**<br>**ops$brdb.brdb_branch_node_info**<br>**ops$brdb.brdb_branch_users,**<br>**ops$brdb.brdb_branch_user_sessions**<br>**ops$brdb.brdb_branch_user_roles**<br>**ops$brdb.brdb_branch_user_last_logon**<br>**ops$brdb.brdb_password_history**<br>**ops$brdb.rd_package_delivery**<br>**ops$brdb.rdds_checksum** | Role assigned to all users used by the Branch Access Layer to connect to the **training** schema |

POL00115409



| | | | |
|---|---|---|---|
| BATCHLVROLE | CONNECT<br>EXP_FULL_DATABASE<br>AUDITOR | **SELECT,INSERT,UPDATE,DELETE on WKG_BRDB_JSN_USN_SSN**<br>**Select on all non-working OPS$BRDB tables**<br>**Insert on all non-working OPS$BRDB tables**<br>**DELETE on tables which are purged by the archive routine.**<br>UPDATE on brdb_bal_sql_definition_list,**brdb_partitioned_tables, brdb_sub_partition_ranges, brdb_system_parameters, brdb_process_control, brdb_partition_creates, brdb_tt_transactions, brdb_c0_reversals**<br>SELECT on OPS$BRDB views.<br>SELECT on OPS$BRDB sequences<br>EXECUTE on fn_compress_4_to_3, pkg_brdb_common, pkg_brdb_data_aggregation, pkg_brdb_exception, pkg_brdb_feed_common, pkg_brdb_hydra_inday, pkg_brdb_hydra_recon, pkg_brdb_optimiser, pkg_partition_utils, wkg_truncate_for_x031 | Role assigned to all users that are used to execute BRDB batch jobs against the **Live** schema |
| | | | |
| APPSUP | CONNECT<br>RESOURCE | All System privileges that belong to the DBA role. | Role has been defined for use by ISD Support which will act as first line support team for the Branch Database |
| AUDITOR | MONITOR<br>CONNECT | SELECT ON<br>sys.aud$<br>**ops$brdb.brdb_fad_hash_outlet_mapping**<br>**ops$brdb.brdb_branch_info** | Role defined for use by Internal/External auditors of the system. The role has access to the live schema only |
| APP_MONITOR | CONNECT<br>RESOURCE | | Role has been granted to "APP_MONITOR_USER" in order to support the application exception monitoring using ITM and Oracle Application Express products.<br>Monitoring is provided for the **Live** schema only. |
| DB_MONITOR | CONNECT<br>EXP_FULL_DATABASE | SELECT ANY TABLE<br>SELECT ANY DICTIONARY | Role has been defined for use by ISD Support which will act as first line support team for the Branch Database |
| EMDB_USERS | CONNECT | SELECT,INSERT,UPDATE,DELETE on ops$brdb.emdb_managed_node,<br>ops$brdb.emdb_post_office | Role used by the EMDB – BRDB Interface which provides the BRDB with the EM view of branch and counter status |
| MONITOR | CONNECT | SELECT ANY TABLE | Role made available for all users that require query-only access to the system |
| OMDB_USERS | CONNECT | SELECT, INSERT ON **ops$brdb.brdb_branch_info,** | Role used by the OMDB – BRDB interface which provides the SYSMAN with PONR for migrating Branches.. |
| RDDS_USERS | CONNECT | SELECT ON **ops$brdb.brdb_branch_info,** | Role used by RDDS host applications to read data from the Branch Database. |
| RDMC_USERS | CONNECT | SELECT ON **ops$brdb.brdb_branch_info,** | Role used by RDMC host applications to read data from the Branch Database. |

POL00115409



| | | | |
|---|---|---|---|
| GEN_REPORTS | RESOURCE | SELECT on ops$brdb tables gen_rep_report_definitions, gen_rep_report_sections, gen_rep_report_views.<br>SELECT/INSERT/DELETE on gen_rep_report_parameters | Generic reporting tool role |
| SECURITY_MANAGER | CONNECT | CREATE USER<br>GRANT ANY ROLE<br>SELECT ANY TABLE<br>DROP USER<br>ALTER USER | Role has been defined for use by support staff who are authorised to administer support users and to investigate security breaches. |
| SMC | MONITOR | SELECT ANY DICTIONARY | Role has been defined for use by SMC Support which will act as second line support team for the Branch Database |
| SSC | MONITOR<br>RESOURCE | SELECT ANY DICTIONARY | Role has been defined for use by SSC (EDSC) Support which will act as third line support team for the Branch Database |
| DELTRROLE | CONNECT<br>SELECT_CATALOG_ROLE | EXECUTE on ops$brdbtr.pkg_bal_training_data<br>SELECT ANY SEQUENCE | Role is assigned to user responsible for the deleting training data |
| TWS_USERS | CREATE SESSION | SELECT on ops$brdb.brdb_operational_instances | Role is assigned to TWS_USER to determine the status of BRDB instances for batch job scheduling |
| UNXADM | CONNECT<br>RESOURCE<br>DBA | | Role has been defined for use by ISD Support which will act as third line (DBA) support team for the Branch Database |



## 20.4 Sequence Definitions

BRDB-sequences.rtf

## 20.5 Database Links

| Link Name | Is Public? | Username & Initial Password | Connect String |
|---|---|---|---|
| RDDS | Y | RDDSBRDB / BRDBRDDS | RDDS |
| APS | Y | APSBRDB / BRDBAPS | APS |
| DRSNWB | Y | DRSNWBBRDB / BRDBDRSNWB | DRS |
| DRSEFT | Y | DRSEFTBRDB / BRDBDRSEFT | DRS |
| TPS | Y | TPSBRDB / BRDBTPS | TPS |
| LFS | Y | LFSBRDB / BRDBLFS | LFS |
| NPSTT1 | Y | NBX_TT_HARVESTER_AGENT_5 / NBX_TT_HARVESTER_AGENT_5 | NPSSERVICE1_2 |
| NPSTT2 | Y | NBX_TT_HARVESTER_AGENT_6 / NBX_TT_HARVESTER_AGENT_6 | NPSSERVICE1_2 |
| NPSTT3 | Y | NBX_TT_HARVESTER_AGENT_7 / | NPSSERVICE1_2 |

POL00115409



| | | | |
|---|---|---|---|
| | | NBX_TT_HARVESTER_AGENT_7 | |
| NPSTT4 | Y | NBX_TT_HARVESTER_AGENT_8 / NBX_TT_HARVESTER_AGENT_8 | NPSSERVICE1_2 |
| NPSGREV1 | Y | NBX_GREV_AGENT_5 / NBX_GREV_AGENT_5 | NPSSERVICE2_1 |
| NPSGREV2 | Y | NBX_GREV_AGENT_6 / NBX_GREV_AGENT_6 | NPSSERVICE2_1 |
| NPSGREV3 | Y | NBX_GREV_AGENT_7 / NBX_GREV_AGENT_7 | NPSSERVICE2_1 |
| NPSGREV4 | Y | NBX_GREV_AGENT_8 / NBX_GREV_AGENT_8 | NPSSERVICE2_1 |
| BRSS | Y | STRADMIN / ADMINSTR | BRSS |
| | | | |

POL00115409



# 21 Appendix D – Metadata and Static Reference Data Definition

## 21.1 Partition Management metadata

### 21.1.1 Table Groups

| Table Group | Description |
|---|---|
| Message Journal | Holds all journalised messages sent by the counter |
| Settlement Transactions | Holds Settlement transactional details |
| Event Store | Holds details of all events generated by the Counter (other than EPOSS events) |
| Report Session Data | Used to load input data for all transactions to be used for reporting |
| Recovery Transactions | Holds details of all transactions needing reversing in the event of failure |
| Guaranteed Reversals | Holds the Guaranteed Reversals table |
| Aggregations | Contains all aggregation tables |
| Hydra | Groups together Hydra-specific tables |

**Table 15 – BRDB_TABLE_GROUPS Metadata**

POL00115409



## 21.1.2 Partitioned Tables

| Table Group | Table Name | Primary Partition Type | Next Partition | Primary Tablespace | Next Tablespace | Tablespace Root Name | Partition Root Name | Partition Mechanism | Partitions Per Day | Old Partition Other SQL | New Partition Other SQL | Deletion Check Criteria |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Aggregations | BRDB_RX_CUTOFF_SUMMARIES | D | | *1* | | | TRAD_DT | R | | | | |
| | BRDB_TX_TRANSACTION_TOTALS | D | | *1* | | | TRAD_DT | R | | | | |
| | BRDB_DAILY_CUMMULATIVE_SUMMARY | D | | *1* | | | JRNL_DATE | R | | | | |
| | BRDB_DAILY_SUMMARY | D | | *1* | | | JRNL_DATE | R | | | | |
| | BRDB_TX_APS_TRANSACTION_TOTALS | D | | *1* | | | TRAD_DT | R | | | | |
| Hydra | BRDB_HYDRA_CT_MODE_TOTALS | D | | *1* | | | TRAD_DATE | R | | | | |
| | BRDB_HYDRA_CT_TOTALS | D | | *1* | | | TRAD_DATE | R | | | | |
| Message Journal | BRDB_RX_MESSAGE_JOURNAL | D | | *1* | | | JRNL_DATE | R | | | | |
| Settlement Transactions | BRDB_RX_APS_TRANSACTIONS | D | | *1* | | | JRNL_DATE | R | | | | |
| | BRDB_RX_BUREAU_TRANSACTIONS | D | | *1* | | | JRNL_DATE | R | | | | |

POL00115409



| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRDB_RX_EPOSS_TRANSACTIONS | D | | *1* | | | JRNL_DATE | R | | | | |
| | BRDB_RX_NWB_TRANSACTIONS | D | | *1* | | | JRNL_DATE | R | | | | |
| | BRDB_RX_DCS_TRANSACTIONS | D | | *1* | | | JRNL_DATE | R | | | | |
| | BRDB_RX_EPOSS_EVENTS | D | | *1* | | | EVENT_DATE | R | | | | |
| Event Store | BRDB_RX_REP_EVENTS | D | | *1* | | | REP_EVENT_ED | R | | | | |
| Report Session Data | BRDB_RX_REP_SESSION_DATA | D | | *1* | | | JRNL_DATE | R | | | | |
| Recovery Transactions | BRDB_RX_RECOVERY_TRANSACTIONS | D | | *1* | | | TXN_STRT_DATE | R | | | | settlement_complete_timestamp IS NULL |
| Guaranteed Reversals | BRDB_RX_GUARANTEED_REVERSALS | D | | *1* | | | RECEIPT_DATE | R | | | | nps_delivered_timestamp IS NULL |

**Table 16 – BRDB_PARTITIONED_TABLES Metadata**

POL00115409



## 21.1.3 Table Partitions

| Table Name | Column Name | Parent Column | Own Tablespace |
|---|---|---|---|
| BRDB_RX_REP_EVENT_DATA | EVENT_DATE | | |
| BRDB_RX_EPOSS_EVENTS | EVENT_DATE | | |
| BRDB_RX_NWB_TRANSACTIONS | JOURNAL_DATE | | |
| BRDB_DAILY_CUMULATIVE_SUMMARY | JOURNAL_DATE | | |
| BRDB_DAILY_SUMMARY | JOURNAL_DATE | | |
| BRDB_RX_EPOSS_TRANSACTIONS | JOURNAL_DATE | | |
| BRDB_RX_DCS_TRANSACTIONS | JOURNAL_DATE | | |
| BRDB_RX_BUREAU_TRANSACTIONS | JOURNAL_DATE | | |
| BRDB_RX_APS_TRANSACTIONS | JOURNAL_DATE | | |
| BRDB_RX_REP_SESSION_DATA | JOURNAL_DATE | | |
| BRDB_RX_MESSAGE_JOURNAL | JOURNAL_DATE | | |
| BRDB_RX_GUARANTEED_REVERSALS | RECEIPT_DATE | | |
| BRDB_RX_CUT_OFF_SUMMARIES | TRADING_DATE | | |
| BRDB_HYDRA_CT_TOTALS | TRADING_DATE | | |
| BRDB_HYDRA_CT_MODE_TOTALS | TRADING_DATE | | |
| BRDB_TX_APS_TRANSACTION_TOTALS | TRADING_DATE | | |
| BRDB_TX_TRANSACTION_TOTALS | TRADING_DATE | | |
| BRDB_RX_RECOVERY_TRANSACTIONS | TRANSACTION_START_DATE | | |



**Table 17 – BRDB_TABLE_PARTITIONS Meta Data**

## 21.1.4 Sub-partition Ranges

| | | | | | | |
|---|---|---|---|---|---|---|
| BRDB_RX_REP_EVENT_DATA | EVENT_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_EPOSS_EVENTS | EVENT_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_NWB_TRANSACTIONS | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_DAILY_CUMULATIVE_SUMMARY | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_DAILY_SUMMARY | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_EPOSS_TRANSACTIONS | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_DCS_TRANSACTIONS | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_BUREAU_TRANSACTIONS | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_APS_TRANSACTIONS | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_REP_SESSION_DATA | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_MESSAGE_JOURNAL | JOURNAL_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |

POL00115409



| | | | | | | |
|---|---|---|---|---|---|---|
| BRDB_RX_GUARANTEED_REVERSALS | RECEIPT_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_CUT_OFF_SUMMARIES | TRADING_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_HYDRA_CT_TOTALS | TRADING_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_HYDRA_CT_MODE_TOTALS | TRADING_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_TX_APS_TRANSACTION_TOTALS | TRADING_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_TX_TRANSACTION_TOTALS | TRADING_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |
| BRDB_RX_RECOVERY_TRANSACTIONS | TRANSACTION_START_DATE | 1 | | D | &CURRDT1 | YYYYMMDD |

**Table 18 – BRDB_SUBPARTITION_RANGES Meta Data**

POL00115409



# 21.2 Housekeeping & Maintenance Metadata

## 21.2.1 Archived Tables

| Archive Group Alias | Archive Group Description | Table Name | Table Alias | Timestamp Column Name | Archive Type | Purge After Processing | Archive Directory | Audit Directory | Retention Period | Archive Threshold | Additional Criteria |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AST | Archived Statistics Table | BRDB_OBJECT_STATS_ARC | AST | D1 | RP | Y | BRDB_ARCHIVE_OUTPUT | | 2 | | And ROWNUM < 10001 |
| AUD | System audit table | SYS.AUD$ | AUD | TIMESTAMP# | RP | Y | BRDB_ARCHIVE_OUTPUT | BRDB_HOST_AUDIT_OUTPUT | 5 | | |
| BCH | Branch Details | BRDB_BRANCH_DECL | BD2 | INSERT_TIMESTAMP | RP | Y | BRDB_ARCHIVE_OUTPUT | | 0 | | AND ( zero_decl_yn = 'Y' OR (fad_hash,branch_accounting_code,stock_unit) IN (SELECT fad_hash,branch_accounting_code, stock_unit FROM brdb_branch_decl MINUS SELECT fad_hash,branch_accounting_code, stock_unit FROM brdb_branch_stock_units WHERE NOT (is_deleted = 'Y' AND deleted_date < (TO_DATE(pkg_brdb_common.fn_get_short_date,'YYYYMMDD')-28)))) |
| BCH | Branch Details | BRDB_SU_PENDING_TRANSFER_DET | BT1 | INSERT_TIMESTAMP | RP | Y | | | 0 | | AND rowid IN (SELECT t2.rowid FROM brdb_su_pending_transfer t1, brdb_su_pending_transfer_det t2 WHERE t1.status = 'C' AND TO_DATE(pkg_brdb_common.fn_get_short_date,'YYYYMMDD') - TRUNC(t1.update_timestamp) > 19 AND t2.fad_hash = t1.fad_hash |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | AND t2.branch_accounting_code = t1.branch_accounting_code AND t2.node_id = t1.node_id AND t2.source_session_id = t1.source_session_id) |
| BCH | Branch Details | BRDB_SU_PENDING_TRANSFER | BT2 | UPDATE_TIMESTAMP | RP | Y | | | 20 | | AND STATUS = 'C' |
| BCH | Branch Details | BRDB_SUSPENDED_CUSTOMER_SESS | BSS | UPDATE_TIMESTAMP | RP | Y | | | 20 | | AND UPPER(session_status) = 'COMPLETE' |
| BCH | Branch Details | BRDB_SU_OPENING_BALANCE | BD9 | INSERT_TIMESTAMP | RP | Y | | | 62 | | AND rowid IN (SELECT t2.rowid FROM brdb_branch_info t1, brdb_su_opening_balance t2 WHERE t2.fad_hash = t1.fad_hash AND t2.branch_accounting_code = t1.branch_accounting_code AND (LEAST(t1.current_trading_period, 12) + DECODE(SIGN(LEAST(t1.current_trading_period, 12)- LEAST(t2.trading_period, 12)), -1, 12, 0)) - (LEAST(t2.trading_period, 12)) > 3) |
| BCH | Branch Details | BRDB_RX_BTS_DATA | BD8 | INSERT_TIMESTAMP | RP | Y | | | 62 | | AND rowid IN (SELECT t2.rowid FROM brdb_branch_info t1, brdb_rx_bts_data t2 WHERE t2.fad_hash = t1.fad_hash AND t2.branch_accounting_code = t1.branch_accounting_code AND (LEAST(t1.current_trading_period, 12) + DECODE(SIGN(LEAST(t1.current_trading_period, 12)- LEAST(t2.trading_period, 12)), -1, 12, 0)) - (LEAST(t2.trading_period, 12)) > 3) |
| BCH | Branch Details | BRDB_PS_BARCO | PSB | UPDATE_TIM | RP | Y | | | 20 | | AND UPPER(collection_state) IN |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | DES | | ESTAMP | | | | | | | ('OUTOFOFFICE','EXCEPTION') |
| BCH | Branch Details | BRDB_CUTOFF_TOTALS | BD7 | INSERT_TIMESTAMP | RP | Y | | | 62 | | AND rowid IN (SELECT t2.rowid FROM brdb_branch_info t1, brdb_cutoff_totals t2 WHERE t2.fad_hash = t1.fad_hash AND t2.branch_accounting_code = t1.branch_accounting_code AND (LEAST(t1.current_trading_period, 12) + DECODE(SIGN(LEAST(t1.current_trading_period, 12)- LEAST(t2.trading_period, 12)), -1, 12, 0)) - (LEAST(t2.trading_period, 12)) > 3) |
| BCH | Branch Details | BRDB_CUTOFF_MARKERS | BD5 | INSERT_TIMESTAMP | RP | Y | | | 62 | | AND rowid IN (SELECT t3.rowid FROM brdb_branch_info t1,brdb_cutoff_details t2, brdb_cutoff_markers t3 WHERE t2.fad_hash = t1.fad_hash AND t2.branch_accounting_code = t1.branch_accounting_code AND t3.fad_hash = t2.fad_hash AND t3.branch_accounting_code = t2.branch_accounting_code AND t3.stock_unit = t2.stock_unit AND t3.report_id = t2.report_id AND t3.cut_off_timestamp = t2.cut_off_timestamp AND (LEAST(t1.current_trading_period, 12) + DECODE(SIGN(LEAST(t1.current_trading_period, 12)- LEAST(t2.trading_period, 12)), -1, 12, 0)) - (LEAST(t2.trading_period, 12)) > 3) |
| BCH | Branch Details | BRDB_CUTOFF_DETAILS | BD6 | INSERT_TIMESTAMP | RP | Y | | | 62 | | AND rowid IN (SELECT t2.rowid FROM brdb_branch_info t1, brdb_cutoff_details t2 WHERE t2.fad_hash = t1.fad_hash AND t2.branch_accounting_code = |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | t1.branch_accounting_code AND (LEAST(t1.current_trading_period, 12) + DECODE(SIGN(LEAST(t1.current_trading_period, 12)- LEAST(t2.trading_period, 12)), -1, 12, 0)) - (LEAST(t2.trading_period, 12)) > 3) |
| BCH | Branch Details | BRDB_BRANCH_USER_SESSIONS | BD4 | SESSION_START_TIMESTAMP | RP | Y | | | 7 | | AND UPPER(session_status) IN ('LOGOUT','INVALIDATE') |
| BCH | Branch Details | BRDB_BRANCH_STOCK_UNITS | BD1 | DELETED_DATE | RP | Y | | | 205 | | AND is_deleted = 'Y' |
| BCH | Branch Details | BRDB_BRANCH_DECL_ITEM | BD3 | INSERT_TIMESTAMP | RP | Y | | | 0 | | AND (fad_hash,branch_accounting_code,stock_unit) IN (SELECT fad_hash,branch_accounting_code, stock_unit FROM brdb_branch_decl_item MINUS SELECT fad_hash,branch_accounting_code, stock_unit FROM brdb_branch_decl) |
| CMJ | Counter Message Journal | BRDB_RX_MESSAGE_JOURNAL | BMJ | | RP | | | | 4 | | |
| DMO | Desktop Memos | RDDS_DESKTOP_MEMO | RDM | BRDB_RECEIVED_TIMESTAMP | RP | | BRDB_ARCHIVE_OUTPUT | | 62 | | |
| DMO | Desktop Memos | RDDS_DESKTOP_MEMO_DISTR | RMD | BRDB_RECEIVED_TIMESTAMP | RP | | BRDB_ARCHIVE_OUTPUT | | 62 | | |
| DMO | Desktop Memos | BRDB_DESKTOP_MEMO_USER_DISTR | BMD | BRDB_RECEIVED_TIMESTAMP | RP | | BRDB_ARCHIVE_OUTPUT | | 62 | | AND counter_read_timestamp IS NOT NULL |
| DSM | Daily Summary | BRDB_DAILY_SUMMARY | BDS | | RP | | | | 85 | | |
| DSM | Daily Summary | BRDB_DAILY_CUM | BCS | | RP | | | | 5 | | |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ULATIVE_SUMMARY | | | | | | | | | |
| DSM | Daily Summary | BRDB_TX_TRANSACTION_TOTALS | BTO | | RP | | | | 6 | | |
| DSM | Daily Summary | BRDB_TX_APS_TRANSACTION_TOTALS | BAO | | RP | | | | 6 | | |
| EVT | Events Table | BRDB_RX_EPOSS_EVENTS | BEE | | RP | | | | 4 | | |
| EVT | Events Table | BRDB_RX_REP_EVENT_DATA | BRE | | RP | | | | 63 | | |
| HYD | Hydra | BRDB_HYDRA_CT_TOTALS | BHT | | RP | | | | 6 | | |
| HYD | Hydra | BRDB_HYDRA_CT_MODE_TOTALS | BMT | | RP | | | | 6 | | |
| LFS | Logistics, Stock and Cash | LFS_PLO_HEADER | LPH | BRDB_RECEIVED_TIMESTAMP | RP | Y | | | 14 | | |
| LFS | Logistics, Stock and Cash | LFS_PLO_DETAILS | LPD | BRDB_RECEIVED_TIMESTAMP | RP | Y | | | 14 | | |
| LFS | Logistics, Stock and Cash | LFS_RDC_HEADER | LRH | BRDB_RECEIVED_TIMESTAMP | RP | Y | | | 14 | | |
| LFS | Logistics, Stock and Cash | LFS_RDC_DETAILS | LRD | BRDB_RECEIVED_TIMESTAMP | RP | Y | | | 14 | | |
| LFS | Logistics, Stock and Cash | BRDB_POUCH_DEL_HEADER | BPH | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND lfs_delivered_timestamp IS NOT NULL |
| LFS | Logistics, Stock and Cash | BRDB_POUCH_DEL_DETAILS | BPX | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND (fad_hash,branch_accounting_code,delivery_id) IN (SELECT fad_hash,branch_accounting_code,delivery_id FROM brdb_pouch_del_details MINUS |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | SELECT fad_hash,branch_accounting_code, delivery_id FROM brdb_pouch_del_header) |
| LFS | Logistics, Stock and Cash | BRDB_POUCH_COLL_HEADER | PCH | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND lfs_delivered_timestamp IS NOT NULL |
| LFS | Logistics, Stock and Cash | BRDB_POUCH_COLL_DETAILS | PCD | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND (fad_hash,branch_accounting_code ,collection_id) IN (SELECT fad_hash,branch_accounting_code, collection_id FROM brdb_pouch_coll_details MINUS SELECT fad_hash, branch_accounting_code,collection_id FROM brdb_pouch_coll_header) |
| LFS | Logistics, Stock and Cash | BRDB_CASH_HEADER | BCH | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND lfs_delivered_timestamp IS NOT NULL |
| LFS | Logistics, Stock and Cash | BRDB_CASH_DETAILS | BCD | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND (fad_hash,branch_accounting_code ,trading_date) IN (SELECT fad_hash,branch_accounting_code, trading_date FROM brdb_cash_details MINUS SELECT fad_hash,branch_accounting_code, trading_date FROM brdb_cash_header) |
| NXN | NPS Transactional Table | BRDB_RX_GUARANTEED_REVERSALS | BGR | | RP | | | | 6 | | |
| NXN | NPS Transactional Table | BRDB_RX_TT_TRANSACTIONS | BTT | | RP | Y | | | 5 | | |
| OPR | Operational Table | BRDB_OPERATIONAL_EXCEPTIONS | OPE | EXCEPTION_TIMESTAMP | RP | Y | BRDB_ARCHIVE_OUTPUT | | 62 | | |
| OPR | Operational Table | BRDB_HYDRA_EXCEPTIONS | HEX | INSERT_TIMESTAMP | RP | Y | BRDB_ARCHIVE_OUTPUT | | 31 | | |
| OPR | Operational Table | BRDB_PARTITION | BPN | STATUS_DAT | RP | Y | BRDB_ARCHI | | 62 | | AND status = 'DEL' |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CREATES | | E | | | VE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_PARTITION_STATUS_HISTORY | BPH | CREATE_DATE | RP | Y | BRDB_ARCHIVE_OUTPUT | | 62 | | AND rownum < 10001<br>AND rowid IN<br>(SELECT bpsh.rowid<br>FROM<br>brdb_partition_status_history bpsh<br>WHERE NOT EXISTS<br>(SELECT 1<br>FROM<br>brdb_partitioned_tables bpt,<br>brdb_partition_creates bpc<br>WHERE<br>bpt.table_name = bpsh.table_name<br>AND bpc.table_name = bpt.table_name<br>AND bpc.partition_range_value = LTRIM(bpsh.partition_name,bpt.partition_root_name||'_')<br>AND bpc.status <> 'DEL')) |
| OPR | Operational Table | BRDB_PROCESS_AUDIT | BPA | SYSTEM_DATE | RP | Y | BRDB_ARCHIVE_OUTPUT | BRDB_HOST_AUDIT_OUTPUT | 62 | | |
| OPR | Operational Table | BRDB_PROCESS_CONTROL | BPC | SYSTEM_DATE | RP | Y | BRDB_ARCHIVE_OUTPUT | | 62 | | |
| OPR | Operational Table | BRDB_RX_REPORT_REPRINTS | BRR | INSERT_TIMESTAMP + KEEP_DAYS | RP | Y | | | 0 | | |
| OPR | Operational Table | BRDB_HOST_INTE | BIE | PROCESSING | RP | Y | | | 5 | | |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RFACE_FEED_EXCP | | _DATE | | | | | | | |
| OPR | Operational Table | BRDB_OPERATIONAL_INSTANCES_HST | OIH | INSERT_TIMESTAMP | RP | Y | BRDB_ARCHIVE_OUTPUT | BRDB_HOST_AUDIT_OUTPUT | 62 | | |
| OPR | Operational Table | BRDB_PASSWORD_HISTORY | PWH | INSERT_TIMESTAMP | RP | Y | | | 0 | | AND rowid IN (SELECT rowid FROM (SELECT rowid, ROW_NUMBER() OVER (PARTITION BY fad_hash,branch_accounting_code, branch_user ORDER BY insert_timestamp DESC) pos FROM brdb_password_history) WHERE pos > 12) |
| OPR | Operational Table | BRDB_ANALYZED_OBJECTS | BAO | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_ARCHIVED_TABLES | ART | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_BAL_CONFIG_PARAMETERS | BCP | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_BAL_SQL_DEFINITIONS | BSD | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_EXCEPTION_CODES | BEX | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_FAD_HASH_INSTANCE_MAPPING | FHI | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_FAD_HASH_OUTLET_MAPPING | FHO | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_FAD_HASH_VALUES | FHV | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_FILES_TO_HOUSEKEEP | FHK | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_HOST_AGGREGATIONS | BHA | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OPR | Operational Table | BRDB_HOST_INTERFACE_FEEDS | BIF | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_OPERATIONAL_INSTANCES | BOI | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_ORACLE_ERROR_CODES | OEC | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_PARTITIONED_INDEXES | BPI | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_PARTITIONED_TABLES | BPT | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_PROCESSES | BPR | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_SQL_HINTS | BSH | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_SUBPARTITION_RANGES | BSR | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_SYSTEM_PARAMETERS | BSP | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_TABLE_GROUPS | BTG | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_TABLE_PARTITIONS | BTP | | FT | | BRDB_ARCHIVE_OUTPUT | | | | |
| OPR | Operational Table | BRDB_FILE_AUDIT_TRAIL | FAT | STATUS_CHANGE_TIMESTAMP | RP | Y | BRDB_ARCHIVE_OUTPUT | | 62 | | |
| RCV | Recovery Transactions Table | BRDB_RX_RECOVERY_TRANSACTIONS | BRT | | RP | | | | 4 | | |
| RDD | RDDS Ref Data | RDDS_CHECKSUM_HIST | RCH | INSERT_TIMESTAMP | RP | Y | | | 30 | | |
| RDD | RDDS Ref Data | RDDS_CHECKSUM | RCS | INSERT_TIMESTAMP | RP | Y | | | 30 | | |
| RDD | RDDS Ref Data | BRDB_REF_DATA_SLAS | RDS | FIRST_LOG_ON | RP | Y | | | 30 | | |

POL00115409



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| REM | REM pouch | BRDB_REM_OUT_POUCH_DETAILS | PH1 | INSERT_TIMESTAMP | RP | Y | | | 5 | | AND UPPER(pouch_status) IN ('DESPATCHED','DESTROYED') |
| RTN | Report Transactional Table | BRDB_RX_REP_SESSION_DATA | BSD | | RP | | | | 62 | | |
| TCT | Transaction Correction Tool | BRDB_TXN_CORR_TOOL_JOURNAL | TCJ | JOURNAL_DATE | RP | Y | | | 4 | | |
| TPS | Hydra Processing | TPS_HYDRA_INDAY_DATA | HID | TRADING_DATE | RP | Y | | | 30 | | AND processed_yn = 'Y' |
| TPS | Hydra Processing | TPS_HYDRA_RECON_TOTALS | HRT | TRADING_DATE | RP | Y | | | 5 | | AND processed_yn IN ('Y','E') |
| TXN | Transactional Table | BRDB_RX_APS_TRANSACTIONS | BAT | | RP | | | | 4 | | |
| TXN | Transactional Table | BRDB_RX_BUREAU_TRANSACTIONS | BBT | | RP | | | | 4 | | |
| TXN | Transactional Table | BRDB_RX_EPOSS_TRANSACTIONS | BET | | RP | | | | 4 | | |
| TXN | Transactional Table | BRDB_RX_NWB_TRANSACTIONS | BNT | | RP | | | | 4 | | |
| TXN | Transactional Table | BRDB_RX_DCS_TRANSACTIONS | BFT | | RP | | | | 4 | | |
| TXN | Transactional Table | BRDB_RX_DCS_TRANSACTIONS | BFT | | RP | | | | 4 | | |
| TXN | Transactional Table | BRDB_RX_CUT_OFF_SUMMARIES | COF | | RP | | | | 4 | | |
| TXN | Transactional Table | TPS_TXN_CORRECTION_DETAILS | TCD | TCD COUNTER_PROCESSED_TIMESTAMP | RP | Y | | | 40 | | |



**Table 19 – BRDB_ARCHIVED_TABLES Metadata**

POL00115409



## 21.2.2 Analyzed Objects

| Object Name | Object Group | Object Type | Object Group Description | Analyze Method | Insert Timestamp | Partition Number | Sample Percentage | Block Sample | Method Option |
|---|---|---|---|---|---|---|---|---|---|
| BRDB_RX_MESSAGE_JOURNAL | MSJ | TABLE | Message Journal | DBMS_STATS | *SYSTIMESTAMP* | 3 | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_RX_REP_SESSION_DATA | RSD | TABLE | Report Session Data | DBMS_STATS | *SYSTIMESTAMP* | 3 | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_SU_CUT_OFF_MARKERS | RAG | TABLE | Report Aggregation | DBMS_STATS | *SYSTIMESTAMP* | | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_SU_CUT_OFF_TOTALS | RAG | TABLE | Report Aggregation | DBMS_STATS | *SYSTIMESTAMP* | | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_SU_CUT_OFF_DETAILS | RAG | TABLE | Report Aggregation | DBMS_STATS | *SYSTIMESTAMP* | | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_RX_RECOVERY_TRANSACTIONS | RCV | TABLE | Transaction Recovery | DBMS_STATS | *SYSTIMESTAMP* | 3 | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_DAILY_SUMMARY | DSM | TABLE | Transaction Summary | DBMS_STATS | *SYSTIMESTAMP* | 3 | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_DAILY_CUMULATIVE_SUMMARY | DSM | TABLE | Transaction Summary | DBMS_STATS | *SYSTIMESTAMP* | 3 | 10 | TRUE | FOR ALL INDEXED COLUMNS |
| BRDB_BROUGHT_FWD_STOCK_HIST | DSM | TABLE | Transaction Summary | DBMS_STATS | *SYSTIMESTAMP* | 3 | 10 | TRUE | FOR ALL INDEXED COLUMNS |



**Table 20 – BRDB_ANALYZED_OBJECTS Meta Data**

Note that this is currently not used by the TWS overnight batch job, which performs Schema level statistics.

## 21.2.3 Exception Codes

The complete list of application exception codes will be defined in the Branch Database Support Guide.

## 21.2.4 Files to Housekeep

| Directory Name | File Name | Retention | Delete |
|---|---|---|---|

POL00115409



| | | Period | Subdirectories? |
|---|---|---|---|
| ${BRDB_ARCHIVE_OUTPUT} | *.* | 5 | N |
| | | | |

**Table 21 – BRDB_FILES_TO_HOUSEKEEP Meta Data**

A number of standard Oracle directories which produce on going trace files and core dumps will be housekept by /usr/local/bin/RMANBackup.sh:

/u01/admin/BRDB/adump – Audit_File_Dest: These contain copies of the SYS.AUD$ tables which are archived after 5 days. This directory's '*.aud' files should be retained for 7 days.

/u01/admin/BRDB/bdump – Background_Dump_Dest: trace '*.trc' files and core dumps can be tidied after 28 days.

/u01/admin/BRDB/cdump – Core_Dump_Dest: trace '*.trc' files and core dumps '*.dmp' can be tidied after 28 days.

/u01/admin/BRDB/udump – User_Dump_Dest: trace '*.trc' files and core dump '*.dmp' files can be tidied after 28 days.

$ORACLE_HOME/rdbms/audit – This contains ASM instance audit logs and the '*.aud' files can be tidied after 28 days.

$ORACLE_HOME/rdbms/log – trace '*.trc' files can be tidied after 28 days.



Any trace or core dump files required for Oracle support calls will have been uploaded to Oracle Metalink as evidence files within the retention period.

### 21.2.5 Processes

| Process Name | Process Description | Process Day Multiple Runs | Process in Use |
|---|---|---|---|
| BRDBC001 | Start of Day | N | Y |
| BRDBC002 | Audit Store Auditing | N | Y |
| BRDBX003 | Common Legacy Host Interface | N | Y |
| BRDBC004 | Audit, Archive and DB-Copy | N | Y |
| BRDBX005 | Gather Optimizer Statistics | N | Y |
| BRDBX006 | File Housekeeping | N | Y |
| BRDBX007 | Data Aggregation Tool | N | Y |
| BRDBC008 | Check Job Completion | Y | Y |
| BRDBC009 | End Of Day | N | Y |
| BRDBX011 | Update System Parameters | Y | Y |
| BRDBX013 | Updates the operational instances | Y | Y |
| BRDBX015 | Transaction Correction Tool | Y | Y |
| BRDBX020 | Transfer to DWh | N | Y |
| BRDBX021 | Pause / Start Streams Replication | Y | Y |
| BRDBX030 | Hydra specific XML Data Processor | N | Y |
| BRDBX031 | Reset JSN SSN USN | N | Y |
| BRDBX033 | End Of Day Prep Hydra XML | N | Y |

POL00115409



| | | | |
|---|---|---|---|
| BRDBX034 | RDDS migrating branch pull | N | Y |

**Table 22 – BRDB_PROCESSES Meta Data**

# 21.3 Fad-Hash Mappings

## 21.3.1 Fad-Hash Values

This table contains 128 entries in total of a number ranging from 0…127.

## 21.3.2 Operational Instances

| Instance Id | Instance Description | Is Available | Host Name | Service Name |
|---|---|---|---|---|
| 1 | First Oracle Instance | Y | LPRABDB001 | BRDB1 |
| 2 | Second Oracle Instance | Y | LPRABDB002 | BRDB2 |
| 3 | Third Oracle Instance | Y | LPRABDB003 | BRDB3 |
| 4 | Fourth Oracle Instance | Y | LPRABDB004 | BRDB4 |

**Table 23 – BRDB_OPERATIONAL_INSTANCES Meta Data**



## 21.3.3 Fad-Hash Instance Mapping

The attached mappings spreadsheet provides the Instances and their priorities for each Fad-Hash value.

"Fad-Hash Spreadsheet.xls"

## 21.3.4 Fad-Hash Outlet Mapping

The attached mappings spreadsheet provides the list of Outlets that have been assigned Fad-Hash values.

BRDB_FAD_HASH_OUTLET_MAPPING011(

# 21.4 Host Processing Metadata

## 21.4.1 Host Interface Feeds

The table defines the names of the Host Interface Feeds.

| Interface Feed Name | Interface Description | Uses Fad-Hash? | Uses Auto Repeat? | Auto Repeat Seconds? |
|---|---|---|---|---|
| BRDB_APS_TXN_FROM_TPS | BRDB APS transactions from TPS feed | Y | N | |
| BRDB_APS_TXN_TO_APS | BRDB APS transactions to APS feed | Y | N | |
| BRDB_APS_TXN_TO_TPS | BRDB APS transactions to TPS feed | Y | N | |

POL00115409



| | | | | |
|---|---|---|---|---|
| BRDB_BDC_TXN_FROM_TPS | BRDB BDC transactions from TPS feed | Y | N | |
| BRDB_BDC_TXN_TO_TPS | BRDB BDC transactions to TPS feed | Y | N | |
| BRDB_CASH_TO_LFS | BRDB Cash Declarations to LFS feed | Y | N | |
| BRDB_CNTR_REF_FROM_RDDS | BRDB Counter Reference Data from RDDS feed | N | N | |
| BRDB_CUTOFF_SUMM_TO_TPS | BRDB Cutoff Summaries to TPS feed | Y | N | |
| BRDB_DCS_TXN_FROM_TPS | BRDB DCS transactions from TPS feed | Y | N | |
| BRDB_DCS_TXN_TO_DRS | BRDB DCS transactions to DRS feed | Y | N | |
| BRDB_DCS_TXN_TO_TPS | BRDB DCS transactions to TPS feed | Y | N | |
| BRDB_EMDB_INTERFACE | BRDB Estate Management Interface feed | N | N | |
| BRDB_EPOSS_EVNT_TO_TPS | BRDB EPOSS events to TPS feed | Y | N | |
| BRDB_EPOSS_TXN_FROM_TPS | BRDB EPOSS transactions from TPS feed | Y | N | |
| BRDB_EPOSS_TXN_TO_TPS | BRDB EPOSS transactions to TPS feed | Y | N | |
| BRDB_HOST_REF_FROM_RDDS | BRDB Host Reference Data from RDDS feed | N | N | |
| BRDB_INDAY_XML_FROM_TPS | BRDB In-Day Migration Blob from TPS feed | N | N | |
| BRDB_MEMOS_FROM_RDDS | BRDB Desktop Memos from RDDS feed | N | N | |
| BRDB_NWB_TXN_FROM_TPS | BRDB NWB transactions from TPS feed | Y | N | |
| BRDB_NWB_TXN_TO_DRS | BRDB NWB transactions to DRS feed | Y | N | |
| BRDB_NWB_TXN_TO_TPS | BRDB NWB transactions to TPS feed | Y | N | |
| BRDB_PCOL_TO_LFS | BRDB Pouch Collections to LFS feed | Y | Y | 1 |
| BRDB_PDEL_TO_LFS | BRDB Pouch Deliveries to LFS feed | Y | Y | 1200 |
| BRDB_PLO_FROM_LFS | BRDB Planned Order details from LFS feed | Y | N | |
| BRDB_RDC_FROM_LFS | BRDB Replenishment Delivery details from LFS feed | Y | N | |
| BRDB_RECON_XML_FROM_TPS | BRDB Reconciliation Blob from TPS feed | N | N | |
| BRDB_REF_COPY_FROM_TPS | BRDB Outlets/Transaction Modes from TPS feed | N | N | |
| BRDB_REV_TXN_TO_NPS | BRDB Reversal Records to NPS feed | Y | Y | 0 |
| BRDB_TT_TXN_TO_NPS | BRDB Track and Trace Records to NPS feed | Y | Y | 180 |
| BRDB_TXN_CORR_FROM_TPS | BRDB Transaction Corrections from TPS feed | N | N | |
| BRDB_TXN_TOT_TO_APS | BRDB Transaction Totals to APS feed | Y | N | |
| BRDB_TXN_TOT_TO_TPS | BRDB Transaction Totals to TPS feed | Y | N | |

**Table 24 – BRDB_HOST_INTERFACE_FEEDS Meta Data**



## 21.4.2 Host Aggregations

The table defines the names of the Host Aggregation SQLs.

POL00115409



| Aggregation Name | Aggregation Description | Uses Fad-Hash? | SQL Id | Aggregation SQL |
|---|---|---|---|---|
| BRDB_NON_CUMU_TXN_AGGR | Daily Non-cumulative Summary is a Daily summarisation at SU, TP, BP, Product and Transaction Mode level | Y | 1 | *To be defined in the Low Level Designs* |
| BRDB_CUMU_TXN_AGGR | Daily Cumulative Summary is a Cumulative summarisation at SU, TP and BP level | Y | 1 | |
| BRDB_TPS_TXN_TOTALS | Outlet level transaction totals of quantities and amounts. This information is passed to TPS-Host to facilitate its transaction reconciliation. | Y | 1 | |
| BRDB_APS_TXN_TOTALS | Outlet level transaction totals of quantities and amounts. This information is passed to APS-Host to facilitate its transaction reconciliation. | Y | 1 | |
| OVERNIGHT_CASH_ON_HAND | The Overnight Cash statement is prepared by aggregating all current Cash declarations made at the Branch | Y | 1 | |
| | | | 2 | |
| | | | 3 | |

**Table 25 – BRDB_HOST_AGGREGATIONS Meta Data**



# 21.5 Branch Specific Metadata

## 21.5.1 Branch Roles

The table defines the names of the Branch Roles to be pre-created in the database.

| Branch User Role | Role Description | Assignable Role | Branch Specific User | Start Date | End Date |
|---|---|---|---|---|---|
| ADMIN | Adminstrator | Y | N | 01/01/1900 | 31/12/3000 |
| AUDITOR | Auditor | Y | N | 01/01/1900 | 31/12/3000 |
| AUDITOR E | Emergency Manager | Y | N | 01/01/1900 | 31/12/3000 |
| MIGRATE | Migrate | Y | N | 01/01/1900 | 31/12/3000 |
| MANAGER | Manager | Y | Y | 01/01/1900 | 31/12/3000 |
| SUPERVISOR | Supervisor | Y | Y | 01/01/1900 | 31/12/3000 |
| CLERK | Clerk | Y | Y | 01/01/1900 | 31/12/3000 |
| TRAINER | Trainer | Y | N | 01/01/1900 | 31/12/3000 |
| SETUP | Setup | Y | N | 01/01/1900 | 31/12/3000 |
| SUPPORT | Support | Y | N | 01/01/1900 | 31/12/3000 |
| ENGINEER | Engineer | Y | N | 01/01/1900 | 31/12/3000 |
| SETUP | Setup | Y | N | 01/01/1900 | 31/12/3000 |

**Table 26 – BRDB_BRANCH_ROLES Meta Data**

## 21.5.2 Branch Service Roles

The table defines the relationship between Services and Branch Roles to be pre-created in the database.

| Branch Role | Branch Service Name | Start Date | End Date |
|---|---|---|---|
| ADMIN | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| ADMIN | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| ADMIN | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| ADMIN | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |
| ADMIN | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| ADMIN | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| ADMIN | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| ADMIN | LockBranchService | 01/01/1900 | 31/12/3000 |
| ADMIN | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| ADMIN | PollingService | 01/01/1900 | 31/12/3000 |
| ADMIN | ReportingService | 01/01/1900 | 31/12/3000 |
| ADMIN | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| ADMIN | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| ADMIN | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| ADMIN | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| ADMIN | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| ADMIN | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| ADMIN | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| ADMIN | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |
| ADMIN | recoveryService | 01/01/1900 | 31/12/3000 |
| AUDITOR | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| AUDITOR | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| AUDITOR | CutOffReportService | 01/01/1900 | 31/12/3000 |
| AUDITOR | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| AUDITOR | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| AUDITOR | LockBranchService | 01/01/1900 | 31/12/3000 |
| AUDITOR | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| AUDITOR | PollingService | 01/01/1900 | 31/12/3000 |
| AUDITOR | ReportingService | 01/01/1900 | 31/12/3000 |
| AUDITOR | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| AUDITOR | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| AUDITOR | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| AUDITOR | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| AUDITOR | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |
| AUDITOR | recoveryService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| AUDITOR E | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| AUDITOR E | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | CutOffReportService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | DownloadDeliveryService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| AUDITOR E | GetValidationCharacter | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| AUDITOR E | IsVRMValid | 01/01/1900 | 31/12/3000 |
| AUDITOR E | LabelEventService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | LockBranchService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | PollingService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| AUDITOR E | ReportingService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| AUDITOR E | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| AUDITOR E | StockUnitManagement | 01/01/1900 | 31/12/3000 |
| AUDITOR E | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | UpdatePSBarcodesStateService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | ValidateChecksumService | 01/01/1900 | 31/12/3000 |
| AUDITOR E | banking | 01/01/1900 | 31/12/3000 |
| AUDITOR E | recoveryService | 01/01/1900 | 31/12/3000 |
| CLERK | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| CLERK | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| CLERK | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| CLERK | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| CLERK | CutOffReportService | 01/01/1900 | 31/12/3000 |
| CLERK | DownloadDeliveryService | 01/01/1900 | 31/12/3000 |
| CLERK | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| CLERK | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| CLERK | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| CLERK | GetValidationCharacter | 01/01/1900 | 31/12/3000 |
| CLERK | IsVRMValid | 01/01/1900 | 31/12/3000 |
| CLERK | LabelEventService | 01/01/1900 | 31/12/3000 |
| CLERK | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| CLERK | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| CLERK | PollingService | 01/01/1900 | 31/12/3000 |
| CLERK | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| CLERK | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| CLERK | ReportingService | 01/01/1900 | 31/12/3000 |
| CLERK | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| CLERK | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| CLERK | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| CLERK | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| CLERK | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| CLERK | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| CLERK | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |
| CLERK | UpdatePSBarcodesStateService | 01/01/1900 | 31/12/3000 |
| CLERK | ValidateChecksumService | 01/01/1900 | 31/12/3000 |
| CLERK | banking | 01/01/1900 | 31/12/3000 |
| CLERK | recoveryService | 01/01/1900 | 31/12/3000 |
| ENGINEER | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| MANAGERS | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| MANAGERS | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| MANAGERS | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| MANAGERS | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| MANAGERS | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |
| MANAGERS | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |
| MANAGERS | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| MANAGERS | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| MANAGERS | CutOffReportService | 01/01/1900 | 31/12/3000 |
| MANAGERS | DownloadDeliveryService | 01/01/1900 | 31/12/3000 |
| MANAGERS | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| MANAGERS | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| MANAGERS | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| MANAGERS | GetValidationCharacter | 01/01/1900 | 31/12/3000 |
| MANAGERS | IsVRMValid | 01/01/1900 | 31/12/3000 |
| MANAGERS | LabelEventService | 01/01/1900 | 31/12/3000 |
| MANAGERS | LockBranchService | 01/01/1900 | 31/12/3000 |
| MANAGERS | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| MANAGERS | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| MANAGERS | PollingService | 01/01/1900 | 31/12/3000 |
| MANAGERS | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| MANAGERS | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| MANAGERS | ReportingService | 01/01/1900 | 31/12/3000 |
| MANAGERS | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| MANAGERS | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| MANAGERS | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| MANAGERS | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| MANAGERS | StockUnitManagement | 01/01/1900 | 31/12/3000 |
| MANAGERS | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| MANAGERS | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| MANAGERS | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| MANAGERS | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| MANAGERS | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| MANAGERS | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| MANAGERS | UpdatePSBarcodesStateService | 01/01/1900 | 31/12/3000 |
| MANAGERS | ValidateChecksumService | 01/01/1900 | 31/12/3000 |
| MANAGERS | banking | 01/01/1900 | 31/12/3000 |
| MANAGERS | recoveryService | 01/01/1900 | 31/12/3000 |
| MIGRATE | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| MIGRATE | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| MIGRATE | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| MIGRATE | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| MIGRATE | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |
| MIGRATE | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |
| MIGRATE | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| MIGRATE | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| MIGRATE | CutOffReportService | 01/01/1900 | 31/12/3000 |
| MIGRATE | DownloadDeliveryService | 01/01/1900 | 31/12/3000 |
| MIGRATE | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| MIGRATE | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| MIGRATE | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| MIGRATE | GetValidationCharacter | 01/01/1900 | 31/12/3000 |
| MIGRATE | IsVRMValid | 01/01/1900 | 31/12/3000 |
| MIGRATE | LabelEventService | 01/01/1900 | 31/12/3000 |
| MIGRATE | LockBranchService | 01/01/1900 | 31/12/3000 |
| MIGRATE | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| MIGRATE | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| MIGRATE | PollingService | 01/01/1900 | 31/12/3000 |
| MIGRATE | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| MIGRATE | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| MIGRATE | ReportingService | 01/01/1900 | 31/12/3000 |
| MIGRATE | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| MIGRATE | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| MIGRATE | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| MIGRATE | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| MIGRATE | StockUnitManagement | 01/01/1900 | 31/12/3000 |
| MIGRATE | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| MIGRATE | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| MIGRATE | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| MIGRATE | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| MIGRATE | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| MIGRATE | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| MIGRATE | UpdatePSBarcodesStateService | 01/01/1900 | 31/12/3000 |
| MIGRATE | ValidateChecksumService | 01/01/1900 | 31/12/3000 |
| MIGRATE | banking | 01/01/1900 | 31/12/3000 |
| MIGRATE | recoveryService | 01/01/1900 | 31/12/3000 |
| SETUP | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| SETUP | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| SETUP | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| SETUP | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| SETUP | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |
| SETUP | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |
| SETUP | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| SETUP | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| SETUP | CutOffReportService | 01/01/1900 | 31/12/3000 |
| SETUP | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| SETUP | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| SETUP | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| SETUP | GetValidationCharacter | 01/01/1900 | 31/12/3000 |
| SETUP | IsVRMValid | 01/01/1900 | 31/12/3000 |
| SETUP | LabelEventService | 01/01/1900 | 31/12/3000 |
| SETUP | LockBranchService | 01/01/1900 | 31/12/3000 |
| SETUP | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| SETUP | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| SETUP | PollingService | 01/01/1900 | 31/12/3000 |
| SETUP | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| SETUP | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| SETUP | ReportingService | 01/01/1900 | 31/12/3000 |
| SETUP | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| SETUP | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| SETUP | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| SETUP | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| SETUP | StockUnitManagement | 01/01/1900 | 31/12/3000 |
| SETUP | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| SETUP | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| SETUP | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| SETUP | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| SETUP | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| SETUP | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |
| SETUP | ValidateChecksumService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| SETUP | banking | 01/01/1900 | 31/12/3000 |
| SETUP | recoveryService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | CutOffReportService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | DownloadDeliveryService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | GetValidationCharacter | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | IsVRMValid | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | LabelEventService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | LockBranchService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | PollingService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | ReportingService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | UpdatePSBarcodesStateService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | ValidateChecksumService | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | banking | 01/01/1900 | 31/12/3000 |
| SUPERVISORS | recoveryService | 01/01/1900 | 31/12/3000 |
| SUPPORT | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| SUPPORT | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| SUPPORT | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |
| SUPPORT | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| SUPPORT | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| SUPPORT | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| SUPPORT | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| TRAINER | AddValidPouchesToCollGroupSvc | 01/01/1900 | 31/12/3000 |
| TRAINER | AttachUserToStockUnitService | 01/01/1900 | 31/12/3000 |
| TRAINER | BasketSettlementService | 01/01/1900 | 31/12/3000 |
| TRAINER | ChangeUserOptionsService | 01/01/1900 | 31/12/3000 |
| TRAINER | ChangeUserRoleService | 01/01/1900 | 31/12/3000 |
| TRAINER | CloseBranchUserAccountService | 01/01/1900 | 31/12/3000 |
| TRAINER | CreateBranchUser | 01/01/1900 | 31/12/3000 |
| TRAINER | CutOffDespatchReportService | 01/01/1900 | 31/12/3000 |
| TRAINER | CutOffReportService | 01/01/1900 | 31/12/3000 |
| TRAINER | DownloadDeliveryService | 01/01/1900 | 31/12/3000 |
| TRAINER | GenericOnlineService | 01/01/1900 | 31/12/3000 |
| TRAINER | GetBranchTradingPeriodService | 01/01/1900 | 31/12/3000 |
| TRAINER | GetOutstandingDeclarations | 01/01/1900 | 31/12/3000 |
| TRAINER | GetValidationCharacter | 01/01/1900 | 31/12/3000 |
| TRAINER | IsVRMValid | 01/01/1900 | 31/12/3000 |
| TRAINER | LabelEventService | 01/01/1900 | 31/12/3000 |
| TRAINER | LockBranchService | 01/01/1900 | 31/12/3000 |
| TRAINER | LockStockUnitService | 01/01/1900 | 31/12/3000 |
| TRAINER | PackageInventoryService | 01/01/1900 | 31/12/3000 |
| TRAINER | PollingService | 01/01/1900 | 31/12/3000 |
| TRAINER | ReconcileBarCodesService | 01/01/1900 | 31/12/3000 |
| TRAINER | RecordStockDeclarations | 01/01/1900 | 31/12/3000 |
| TRAINER | ReportingService | 01/01/1900 | 31/12/3000 |
| TRAINER | ResetTrainingData | 01/01/1900 | 31/12/3000 |
| TRAINER | RolloverBranchService | 01/01/1900 | 31/12/3000 |
| TRAINER | SecureExistingReversalService | 01/01/1900 | 31/12/3000 |
| TRAINER | SetPasswordOnline | 01/01/1900 | 31/12/3000 |
| TRAINER | SetPasswordOtherUser | 01/01/1900 | 31/12/3000 |
| TRAINER | StockUnitManagement | 01/01/1900 | 31/12/3000 |
| TRAINER | StockUnitRolloverBPService | 01/01/1900 | 31/12/3000 |
| TRAINER | StockUnitRolloverTPService | 01/01/1900 | 31/12/3000 |
| TRAINER | UnlockBranchService | 01/01/1900 | 31/12/3000 |
| TRAINER | UnlockStockUnitService | 01/01/1900 | 31/12/3000 |
| TRAINER | UpdateChecksumService | 01/01/1900 | 31/12/3000 |
| TRAINER | UpdateCutOffReportService | 01/01/1900 | 31/12/3000 |

POL00115409



| | | | |
|---|---|---|---|
| TRAINER | UpdatePSBarcodesStateService | 01/01/1900 | 31/12/3000 |
| TRAINER | ValidateChecksumService | 01/01/1900 | 31/12/3000 |
| TRAINER | banking | 01/01/1900 | 31/12/3000 |
| TRAINER | recoveryService | 01/01/1900 | 31/12/3000 |

**Table 28 – BRDB_BRANCH_SERVICES Meta Data**

POL00115409



# 22 Appendix E – Suggested Oracle Initialisation Parameters

This section provides the minimum recommended values for certain initialisation parameters for the Branch Database and the Branch Standby Database.

## 22.1 Branch Database Parameters

### 22.1.1 Core Parameters

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| DB_NAME | BRDB | Name of the Branch Database |
| DB_BLOCK_SIZE | 8KB | DB_BLOCK_SIZE specifies (in kilo bytes) the size of Oracle database blocks.<br>The value of this parameter must be a multiple of the physical block size at the device level (512 bytes). |

**Table 29 –Core BRDB Initialisation Parameters**



### 22.1.2 Listener Configuration Parameters

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| LOCAL_LISTENER | "LISTENER_*hostname*" | Name of the Listener that the Branch Database instance registers itself with. |

**Table 30 –Net Services specific Initialisation Parameters**

### 22.1.3 Tuning related Parameters

#### 22.1.3.1 Instance Tuning

The performance of an Oracle RAC cluster is heavily dependent on the values of database tuning parameters. The following parameters are amongst the most important from a tuning perspective and their recommended minimum values are listed below:

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| ARCHIVE_LAG_TARGET | 900 | Ensures that the Branch Standby Database lags no more than 15 minutes behind the Branch Database. |
| COMPATIBLE | 10.2.0.4.0 | This parameter specifies the release with which Oracle must maintain compatibility. |
| CONTROL_FILE_RECORD_KEEP_TIME | 21 | Needed for RMAN repository records |
| DB_BLOCK_SIZE | 8KB | DB_BLOCK_SIZE specifies (in bytes) the size of Oracle database blocks. The value of this parameter must be a multiple of the physical block size at the device level (512 bytes?). |
| DB_FILE_MULTIBLOCK_READ_COUNT | 16 | High value ensures that more number of blocks are read in a single I/O operation during a sequential read thus improving full-table-scan performance. The high setting is especially useful for Counter Reporting, stock queries and rollovers. |

POL00115409



| | | |
|---|---|---|
| DB_WRITER_PROCESSES | 1 | 1 db-writer process |
| FAST_START_PARALLEL_ROLLBACK | HIGH | Parameter determines the maximum number of processes that can exist for performing parallel rollback.<br>The HIGH setting will allow faster instance recovery from failures. |
| OPEN_CURSORS | 1024 | Set to a high value to allow the BAL to maintain higher number of cursors open. |
| OPEN_LINKS | 16 | Required for database links. |
| PGA_AGGREGATE_TARGET | 5GB | 5GB allows for an average of just over 1 MB per connection (at maximum connections) and 2MB at average number of connections. |
| PROCESSES | 2048 | Set to a high value to allow for high values of other derived parameters. |
| SGA_TARGET | 20GB | A large SGA size will ensure that more data remains in the cache thus improving performance of token authentication and counter reporting, amongst other things. |
| SKIP_UNUSABLE_INDEXES | TRUE | This will allow processes performing DML operations to continue unchanged if any index partitions or sub-partitions need to be disabled at a later stage. |
| TIMED_STATISTICS | TRUE | The Athene Metron performance monitoring software requires this parameter to be set. |

**Table 31 – Instance Tuning specific BRDB Initialisation Parameters**



### 22.1.3.2 Database Tuning

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| FAST_START_MTTR_TARGET | 60 | Based on this parameter value, Oracle works out the db_block_max_dirty_target, which tells DBWR not to have more than certain amount of dirty blocks in the buffers. |
| DB_BLOCK_SIZE | 8KB | DB_BLOCK_SIZE specifies (in bytes) the size of Oracle database blocks. The value of this parameter must be a multiple of the physical block size at the device level (512 bytes).<br>A block size of 8K is a nice balance between the 4K value recommended by Oracle for store & forward systems and 16K value for DSS-type systems. |
| UNDO_MANAGEMENT | AUTO | This is mandatory when using ASM.<br>*Would not choose to use Rollback Segments anyway.* |
| UNDO_RETENTION | 900 | Set to a (relatively) low value, as there are very few long running queries in the Branch Database. Low undo retention will help free up space in the UNDO tablespaces more quickly. |

**Table 32 – Database Tuning specific BRDB Initialisation Parameters**

## 22.1.4 Parameters for Standby Implementation

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| DB_UNIQUE_NAME | BRDB | Unique name for primary database |
| DG_BROKER_START | TRUE | Allows the Data Guard Broker to start functioning |
| DB_BROKER_CONFIG_FILE1 | *Actual values to be provided by Development.* | The first copy of the Data Guard Broker Configuration file resides within ASM |
| DB_BROKER_CONFIG_FILE1 | *Actual values to be provided by Development.* | Second copy of the Data Guard Broker Configuration file resides outside ASM on the OCFS managed file system |
| DB_FILE_NAME_CONVERT | | The value can be left as NULL as the Diskgroup names will be identical since the storage for BRDB and STBY will be managed by different ASM |

POL00115409



| | | |
|---|---|---|
| | | instances. |
| JOB_QUEUE_PROCESSES | 64 | Used for oracle job maintenance windows. |
| FAL_SERVER | SBRDB | Specifies the network service name that the standby database should use to connect to the FAL server. |
| FAL_CLIENT | BRDB | Specifies the network service name that the FAL server should use to connect to the standby database. |
| LOG_ARCHIVE_CONFIG | 'DG_CONFIG=(BRDB,SBRDB)' | Enables the transmission of redo logs to the standby database. |
| LOG_ARCHIVE_DEST_1 | *Actual values to be provided by Development.* | This is used to specify the destination of the archived redo-logs to assist in BRDB recovery. Use ASM disk group BRDB_FLASH for location.<br>Use parameters MANDATORY, VALID_FOR=(ALL_LOGFILES, ALL_ROLES).<br>This parameter has nothing to do with replication to standby. |
| LOG_ARCHIVE_DEST_2 | *Actual values to be provided by Development.* | SERVICE should point to the single node standby instance (STBY1).<br>Use parameters OPTIONAL, VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)<br>Use LGWR ASYNC NOAFFIRM |
| LOG_ARCHIVE_FORMAT | 'brdb_%T_%S_%R.arc' | %S corresponds to the sequence number (zero-filled) and %R corresponds to the reset-logs ID. The %T, which is Real Application Clusters specific, corresponds to the thread number (zero-filled). |
| LOG_ARCHIVE_MAX_PROCESSES | 4 | Specify more than 1 to allow for high archive workload at peak transaction times. |
| LOG_FILE_NAME_CONVERT | | The value can be left as NULL as the Diskgroup names will be identical since the storage for BRDB and STBY will be managed by different ASM instances. |
| STANDBY_FILE_MANAGEMENT | AUTO | Ensures that when datafiles are added to or dropped from the primary database, corresponding changes are made automatically to the standby database. |



**Table 33 – Standby Database specific BRDB Initialisation Parameters**

### 22.1.5 Parameters for Streams Replication

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| GLOBAL_NAMES | TRUE | Needs to be set to this value for every database participating in Streams replication. |

**Table 34 – Streams Replication specific BRDB Initialisation Parameters**

## 22.2 Branch Standby Database Parameters

Set the following initialization parameters for the Branch Standby Database (in addition to the ones already present):

| Parameter Name | Recommended Value | Description / Reason |
|---|---|---|
| COMPATIBLE | 10.2.0.4.0 | This parameter specifies the release with which Oracle must maintain compatibility. |
| DB_BLOCK_SIZE | 8KB | DB_BLOCK_SIZE specifies (in kilo bytes) the size of Oracle database blocks.<br>Set to same value as primary. |
| DB_BROKER_CONFIG_FILE1 | *Actual values to be* | The first copy of the Data Guard Broker Configuration file resides within |

POL00115409



| | | |
|---|---|---|
| | *provided by Development.* | ASM |
| DB_BROKER_CONFIG_FILE1 | *Actual values to be provided by Development.* | Second copy of the Data Guard Broker Configuration file resides outside ASM on the OCFS managed file system |
| DB_FILE_NAME_CONVERT | | The value can be left as NULL as the Diskgroup names will be identical since the storage for BRDB and STBY will be managed by different ASM instances. |
| DB_NAME | BRDB | Same name to be used across databases replicated using DataGuard |
| DB_UNIQUE_NAME | SBRDB | Unique name for standby database |
| DG_BROKER_START | TRUE | Allows the Data Guard Broker to start functioning |
| FAL_SERVER | BRDB, | Specifies the network service name that the standby database should use to connect to the FAL server. |
| FAL_CLIENT | '(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=lprpbds001-vip)(PORT=1529)))(CONNECT_DATA=(SERVICE_NAME=SBRDB_XPT)(INSTANCE_NAME=SBRDB1)(SERVER=dedicated)))' | Specifies the network service name that the FAL server should use to connect to the standby database. |
| INSTANCE_NAME (1) | SBRDB | The instance names of the standby nodes are unique while in standby mode. Once the standby takes on primary responsibilities, its instance names need to change to BRDB1…4 and the instance names of the old primary (new standby) should change to STBY1…4. |
| LOG_ARCHIVE_CONFIG | 'DG_CONFIG=(SBRDB,BRDB)' | |
| LOG_ARCHIVE_DEST_1 | *Actual values to be provided by Development.* | This is used to specify the destination of the archived redo-logs to assist in recovering the standby database. Use ASM disk group BRDB_FLASH for location.<br>Use parameters MANDATORY, VALID_FOR=(ALL_LOGFILES, ALL_ROLES).<br>This parameter has nothing to do with data guard replication from the Branch Database. |
| LOG_ARCHIVE_DEST_2 | | Parameter does not need to be specified. In the event of a corruption of the Standby database along with the loss of the Archive log files, the database will be recreated from Live. |
| LOG_ARCHIVE_FORMAT | 'brdb_%T_%S_%R.arc' | %S corresponds to the sequence number (zero-filled) and %R corresponds to the reset-logs ID. The %T, which is Real Application Clusters specific, corresponds to the thread number (zero-filled). |
| LOG_FILE_NAME_CONVERT | | The value can be left as NULL as the Diskgroup names will be identical since the storage for BRDB and STBY will be managed by different ASM instances. |
| SGA_TARGET | 20GB | A large SGA size will ensure that more data remains in the cache thus improving performance of token authentication and counter reporting, amongst other things. |
| STANDBY_FILE_MANAGEMENT | AUTO | Ensures that when data files are added to or dropped from the primary database, corresponding changes are made automatically to the standby database. |



**Table 35 –BRDB-Standby Initialisation Parameters**

In addition, log and archived-redo log file name conversion must be set up to conform to the file naming conventions.

POL00115409



## 22.3 Role Reversal Parameter Changes

Branch Database and Branch Database Standby role reversal is covered in the Branch Database Support Guide, See Ref [R43].