FUJ00234958

Document Title:	HOST BRANCH DATABASE SUPPORT GUIDE
Document Reference:	DES/APP/SPG/0001
Document Type:	SUPPORT GUIDE
Release:	HNG-X Release 21.80
Abstract:	This Support Guide details information in support and maintenance of the Branch, the Branch Support and the Standby databases
Document Status:	APPROVED
Author & Dept:	Wing Pang, HNG-X Host Development Tony Dolton, HNG-X Host Development Gareth Seemungal, HNG-X Host Development Pete Jobson, Technical Architecture & Consulting Folusho Ogunlana, HNG-X Host Development Vivek Shrivastava, HNG-X Host Development Srinivas Marupudi, HNG-X Host Development Pankaj Yadav, HNG-X Host Development Akshya Kumar Nahak,HNG-X Host Development
External Distribution:	None
Information Classification:	See section 0.9

Approval Authorities:

Name	Role	Signature	Date
Adam Woodley	SSC		

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

FUJ00234958

0 Document Control

0.1 Table of Contents

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

FUJ00234958

0.2 Document History

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK Reference
0.1	22nd June 2009	Initial Version	N/A
0.2	18th September 2009	First major update to all sections	N/A
0.3	23rd October 2009	Updated with schedule details and other information.	N/A
0.4	29th October 2009	Updated with general review comments and additions to Streams and Standby procedures.	N/A
0.5	29th October 2009	Updated with Streams related information.	N/A
1.1	5th November 2009	Added new Hydra functionality	CP404
1.2	12th January 2010	Added Transaction Acknowledgement copy.	CP 4914S
1.3	18th January 2010	Added stock unit unlock, update outstanding recovery txn and branch rollover unlock functionality.	PC0191404, PC0191168, PC0189018
1.4	17th February	Added process BRDBX035	PC0194351
1.5	17th March 2010	Couple of corrections plus adding bookmarks for schedule document hyperlinks	N/A
1.6	17th May 2010	Couple of corrections plus adding bookmarks for schedule document hyperlinks	N/A
1.7	28th June 2010	Added BRSS schedule, TT/GREV changes	PC0200577, PC0200019
1.8	9th July 2010	Added manual start/stop feed commands	N/A
1.9	20th October 2010	Corrections due to review process (comments from SSC, ISD), section added for service outages, changes to recovery, changes to BRDB schedules (remove HYDRA)	PC0203999
1.10	27nd October 2010	Added AEI Near-Real Time Interface. New Sections – 2.3.2.2, 2.4.2 through to 2.4.2.4 Updated Sections – 2.2, 2.3.2, 2.3.4, 2.5.3 +----------+ Updated Transaction Correction templates (all templates in Section 7 – Appendix C)	CP491 +----------+ PC0195962
1.11	17th December 2010	Changes due to ISD review Changed BRDBX005 details to match new implementation	N/A
2.00	3rd February 2011	Document status set to 'APPROVED'	N/A
2.1	10th February 2011	Release 4 branch closure process BRDBX037.sh, new associated schedule + description EMDB -> BRDB description update TPoS - new table added	CP585, CP510
2.3	19th May 2011	Release 4 changes to BRDB purge process [BRDBC004]. Release 4 Capacity Management Reporting solution in BRSS (new modules) Release 5 BRDB Transaction Confirmation feed to APOP (new Host Interface feed)	PC0208496 CP639 CP629
2.4	26th May 2011	Release 5 Post Office Essentials	CP582
2.5	August 2011	Post Office Address File Processing and other amendments including Approver/Reviewer matrix updates.	CP633

FUJ00234958

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK Reference
3.0	21st September 2011	Document status set to 'APPROVED'	N/A
3.1	26th September 2011	Release 05.50 Client File Delivery changes	CP0605
3.2	27th October 2011	Interim updates relating to Releases 05.14 – 05.50	N/A
3.5	23rd January 2012	Corrections/updates based on review comments for release 5.5.	CP0605
4.0	14th February 2012	Document status set to 'APPROVED'	N/A
4.1	27th February 2012	Updates on Standby failover procedure. 6.1(6a)	PC0214200,214299
4.2	18th June 2012	Daemon feed monitoring process BRDBC041	CP741
4.3	28th June 2012	Failback audit file steps	PC0218160
4.4/4.5	11th Jan 2013	Branch financial year update script	CP859
4.6	5th Feb 2013	Corrected parameters for CP859	CP859
4.7	25th Feb 2013	Local Collect And Return	CP0911
4.8	25th Feb 2013	Further updates for Collect and Return (section 3.63.2.1).	CP0911
4.9	4th Mar 2013	Collect and Return Streams refresh steps	CP0911
4.10	22nd Mar 2013	Collect and Return update for BRDBC058	CP0911
4.11	16th May 2013	For review	CP0911
4.12	29th May 2013/10th June 2013	Corrections due to 4.11 review	N/A
5.00	21st June 2013	Document status set to 'APPROVED'	N/A
5.1	30th July 2013	Extended Trading Hours	CP0875
5.2	19th August 2013	Updated for comments received.	
5.3	19th March 2014	Release 12 upgrade to 11g, replace Streams with Goldengate, include notes on SMM [smart metering]	CP0938
5.4	28th July 2014	Royal Mail Extended Data reports	CP1318
5.5	7th November 2014	Further updates for Release 12 upgrade to Oracle 11g. Added procedure for regenerating PSE files (section 3.94.3.2).	CP0938, CP1318
6.0	15th January 2015	Issued for Approval	
6.1	06th February-2015	Update on STANDBY Section to Stop TWS House keeping and backups from running on standby. And to resync RMAN catalog once Failed back to original Configuration. Update to Schedule sectiona dding RMANbackup schedule	**PC0240668**
6.2	19-February-2015	Update comments	
7.0	20 February 2015	Issued for Approval	
7.1	24th March 2015	Update on Standby section to reinstall Oracle Fan Event Handler in BDS Hosts and Comments Update	PC0217938
7.2	02nd April 2015	Update comments and for Approval	N/A
8.0	15-Apr-2015	Approval version	
8.1	24-July-2015	BRDBC060 NRT BAP Daemon Win in Mails changes	CP1519 CP1472,CP1539
9.0	21-Aug-15	Updated review Comments. Submitted for approval	
9.1	18-Nov-2015	Changes for HNG-A Gaps (BRDBC062)	CP1653

FUJ00234958

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK Reference
9.2	14-Dec-2015	Updated for comments received (section 3.105). Corrected cross references (various) and sorted list of associated documents (section 0.4).	
10.0	14-Dec-2015	Version for approval.	
10.1	03-Feb-2016	Updated 6.6.1 Renamed Title, corrected path to audit_stdlist.sh, stated invocation should be 2 days after failback Updated 6.6.4 Corrected path from _brdb to brdb Updated Table 3: Data Guard Failover Procedure, corrected Added new section 6.1 for shutting down Goldengate Updated 6.2 step 8, added DELCRED command Added new section 6.7.5 (Goldengate) Added new section 6.7.6 (Goldengate)	
10.1	16-Feb-2016	Added new section for crontab entries to enable backups and housekeeping after failover. Also, updated entries for tnsnames.ora to enable BRDBC009 run after failover Updated, RMAN configuration for controlfile snapshot after re-instantiation Delete blackout information. Not required anymore. It automatically switches when failover occurs.	PC0249287 PC0249289
10.2	28-Jul-2016	Update Archivelog Policy Update to GG stopping and restart instructions wording.	PC0252015
10.3	19-Oct-2016	Updated with review comments Updated:Password change for RMAN and SYS USER Updated, DBFS files to be removed before re-instanciation of OLD PRIMAY as STANDY.	PC0254283
11.0	03-Jan-2017	Approval version	
11.1	16-Mar-2017	Enhanced User Management	CP1913,CP1941
11.2	26-Apr-2017	Revised following review: section 3.107.	
12.0	26-Apr-2017	Approval version	
12.1	17-May-2017	Credence to Azure Cloud file delivery	CP1955
12.2	01/08/2017	Updates from review	
13.0	16/08/2017	Approval version	CP1955
13.1	07/09/2017	Updated for HNG-X and HNG-A to CFS Interface Implementation changes. Also addressed previous comments.	CP2039
13.2	02/10/2017	Draft issue. Address comment received.	
14.0	02/10/2017	Issued for approval.	
14.1	20/12/2017	ATM Transaction Simplification	CP2076
14.2	10/01/2017	Updates from review	
15.0	15/01/2018	Issued for approval.	
15.1	10/04/2018	EUM Horizon Turning Training Controls On & Off Addition of -r optional parameter for recording MSC reference.	CP-2120 / CT2442
15.2	16/04/2018	Flexible Cash Planning Migration to TransTrack Phase 2	CP2102
16.0	14/05/2018	Issued for approval	CP2102

FUJ00234958

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK Reference
16.1	28/11/2018	POLSAP Migration to TransTrackPhase 3 and 4 + Replacing POLSAP Interfaces with Equivalents from CFS	CP2229, CP2118, CP2242
16.2	06/12/2018	Updated: POLSAP Migration to TransTrackPhase 3 and 4 + Replacing POLSAP Interfaces with Equivalents from CFS	CP2229, CP2118, CP2242
16.3	07/05/2019	Updated with Comments from Review	CP2229, CP2118, CP2242
17.0	27/06/2019	Approval version	CP2229, CP2118, CP2242
17.1	18/09/2019	Bulk Device Enrolment (for PZ & PX)	CP2434; CWO_0154
17.2	01-Oct-2019	BDAS BRSS DATA ACCESS SERVICE HOST DESIGN Also updated with comments from review of v17.1	CP2318; CT2636
18.0	16-10-2019	For Approval	
18.1	29-Jan-2020	Changes for APS/TPS decommission as part of Belfast Exit, as per (the withdrawn CP2347) and CP2392. Activity carried out under Belfast Exit Restart Phase 1.	CP2392 CWO189
18.2	23-Feb-2020	Changes for APS/TPS decommission as part of Belfast Exit to remove old brdb to aps/tps jobs and add new BRDB jobs.	CP2347
18.3	23-Feb-2020	Updated with Comments from Review	CP2347
18.4	18-Mar-2020	Updated with Comments from Review	CP2347
19.0	27-Mar-2020	For Approval	CWO189
19.1	03-Apr-2020	Bulk Device Enrolement phase 2. Decouple the direct interface between SQL Server based EMDB and BRDB. Instead all Estate information now flows through the EMDB2 interface. The old interface is redundant. Branch names/addresses now load into BRDB via the Host RDDS interface instead of the old EMDB package.	CP2515 - R20.65
20.0	16-Apr-2020	For Approval	
20.1	28-Oct-2020	To produce one or more Semi-colon separated files for undo payments transactions in payment and banking service(PBS).	R20.92,CP2459
21.0	11-Dec-2020	Approval version	
21.1	14-Dec-2020	Interface between the Branch Database and the Amazon Web Services PostgreSQL DRS application (DRS2).	R20.94,CP2459
22.0	28-Jan-2021	Approval version	R20.94,CP2459
22.1	11-Feb-2021	Updated Goldengate password reset process in the following areas: Section 6.2, part 8 Section 6.7.7	
22.2	30-Jun-2021	Transaction Correction Tool – Decommissioning Need to remove all references to the TC tool is described in Section 5.6.1 and Appendix C	CP2707,PC0295133
23.0	23-July-2021	Updated following comments from SSC. Approval version.	CP2707,PC0295133
23.1	28-Jun-2022	Belfast BRSS PCI Sanitisation	CP7367
24.0	29-Jun-2022	Approval version	CP7367

FUJ00234958

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK Reference
24.1	22-Aug-2022	Removal of jobs that copy payment card and banking transactions to DRS	CWO0462
25.0	01-sep-2022	Approved	
25.1	04-Oct-2022	Added section 3.99.2.2 (BRDB_TO_EMDB2 deadlock) and referenced KBA McLaughlinD2617J	PC0288667
26.0	09-May-2024	Approval version	

0.3 Review Details

Review Comments by :	
Review Comments to :	gareth.seemungal GRO PostOfficeAccountDocumentManagement GRO
Mandatory Review	
Role	***Name***
SSC	Adam Woodley; sscdm GRO
Application Delivery Development	Pavan Vejendla
Business Architecture & Requirements	Pete Jobson; Gareth Seemungal
Unix Team	Andrew Gibson
Optional Review	
Role	***Name***
DTS – NI Oracle Support	Stuart Johnston
DTS – NI Oracle Support	Niall McKeefry
HNG-X Host Bridge Team Lead	Gyan Patel
HNG-X Host Development	Akshyakumar Nahak
HNG-X Host Development	Prashanta Behera
Test Architect	Mark Ascott
Problem & Major Incident Management	Steve Bansal; POA DutyManager
SMC Team	Ravula Rajaramana; G09D-POA_SMC GRO
Network Operations Manager	Chris Harrison
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
PGM/DCM/TEM/0001 (DO NOT REMOVE)			Fujitsu Post Office Account HNG-X Document Template	Dimensions
DES/APP/DPR/0671			AEI Near-Real Time Design Proposal	Dimensions
DES/APP/HLD/0020			Branch Database High Level Design	Dimensions

FUJ00234958

Reference	Version	Date	Title	Source
DES/APP/HLD/0021			Branch Database Scheduling High Level Design	Dimensions
DES/APP/HLD/0023			Branch Support Database High Level Design	Dimensions
DES/APP/HLD/0025			Branch Support Database Scheduling High Level Design	Dimensions
DES/APP/HLD/0732			NRT Interface Agent High Level Design	Dimensions
DES/APP/HLD/2905			HNG-X PINPad Key Status High Level Design	Dimensions
DES/SYM/HLD/0012			SDAM Horizon Support High Level Design	Dimensions
DEV/APP/LLD/0011			Host Branch Database Gathering Optimiser Statistics	Dimensions
DEV/APP/LLD/0050			BRDB Host System Interfaces Low Level Design	Dimensions
DEV/APP/LLD/0151			Branch Support Database Low Level Design	Dimensions
DEV/APP/LLD/0152			Branch Standby Database Low Level Design	Dimensions
DEV/APP/LLD/0199			Schema Definition for the Branch Database, Standyby Branch Database and Branch Support System	Dimensions
DEV/APP/LLD/0204			Host BRDB Update Outstanding Recovery Transaction Tool Low Level Design	Dimensions
DEV/APP/LLD/0802			Host BRDB Near-Real Time Service Interface – Low Level Design	Dimensions
DEV/APP/LLD/1230			BRDB/BRSS Branch Closure and Archive Process	Dimensions
DEV/APP/LLD/1394			BRSS Host: Data Aggregation and De-normalisation Low Level Design	Dimensions
DEV/APP/LLD/1505			BRDB external txn processing BRDBC051 LLD	Dimensions
DEV/APP/LLD/2157			BRDBC055 Branch Full Event Daemon Processing LLD	Dimensions
DEV/APP/LLD/2789			BRDB Host BRDBC060 BAP Daemon Processing Low Level Design	Dimensions
DEV/APP/LLD/2917			Host BRDB Pin Pad Key Loader Low Level Design	Dimensions
DEV/APP/LLD/3347			Host BRDB ForgeRock Loader Low Level Design	Dimensions
DEV/APP/LLD/3390			Host BRDB To Credence File Generation And Reconciliation LLD	Dimensions
DEV/APP/LLD/3419			Host BRDB To CFS File Generation And Reconciliation LLD	Dimensions
DEV/APP/LLD/3461			Host BRDB ATM Operations Simplification LLD	Dimensions
DEV/APP/SPG/2469			Oracle Goldengate Replication Operational Support Guide	Dimensions
REQ/APP/AIS/1833			Telium PIN Pad Key Injection API	Dimensions
SVM/SDM/OLA/1855			Operational Agreement between Fujitsu Services and Ingenico	Dimensions
DEV/APP/LLD/3850			Host BRDB Bulk Device Enrollment LLD	Dimensions
DES/APP/IFS/3711			Branch Support Database (BRSS) to Data Access Server (BDAS) Interface Specification	Dimensions
DES/APP/AIS/3718			BRSS Data Access Server (BDAS) to Branch Hub AIS	Dimensions

FUJ00234958

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
ACE	Cisco Application Control Engine
ACFS	ASM CLUSTER File system
AEI	Application & Enrolment Identity
APOP	Automated Payment Out Pay
ASM	Automatic Storage Management
BAL	Branch Access Layer
BDB	Acronym for Branch Database
BDS	Acronym for Branch Standby Database
BKID	Banking Key Identifier
BLCS	Branch Lookup and Confirmation Service
BLOB	Binary Large Object
BOAT	Back Office Application Tower
BRDB	Branch Database Oracle SID
BRS	Acronym for Branch Support Database
CI P2a	Channel Integration Phase 2a
BAP	Barcoding All Parcels
CRS	Oracle Cluster Ready Services
CWS	Collect & Return Web Service
CWS	Collect & Return Web Service
CWC	Cash Web Community
C&R	Collect & Return
DBFS	Database File System
EUM	End User Management
FAN	Oracle Fast Application Notification
FSA	File Staging Area
GREV	Guaranteed Reversals
HLD	High Level Design
ITM	IBM Tivoli Manager
JSN	Journal Sequence Number
LCR	Logical change record (generated by the Goldengate capture process)
LPAN	Logical Processing Area Network
NAS	Network Appliance Storage
NPS	Network Persistant Store
NRT	Near-Real Time
OCR	Oracle Cluster Registry
OGG	Oracle Goldengate

FUJ00234958

Abbreviation	Definition
PAN	Processing Area Network
PPID	Pin Pad Identifier
PODG	Post Office Data Gateway
RFS	Oracle Remote File Server (a process)
RHEL	Red Hat Enterprise Linux
RMAN	Oracle Recovery Manager
SAN	Storage Area Network
SCN	Oracle System Change Number
SHLD	Schedule High Level Design
SMM	Smart Metering (NRT agent)
SQL	Structured Query Language
SSN	Session Sequence Number
TT	Track & Trace
USN	User Sequence Number (in the context of the counter user)
PZ	Payzone
PX	ParcelShop
BDAS	Branch Data Access Server

0.6 Glossary

Term	Definition
BladeFrame	A BladeFrame is a chassis which contains processing blades (pBlade) and control blades, as well as integrated interconnect and power connections. The BladeFrame is connected to networks and storage with fully redundant cables.
Branch Access Layer	The middle-tier that carries out the data storage, retrieval and transfer on behalf of the Counter.
Cluster	A 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,
Credence	This is a management information service that belongs to POL's Back Office Application Tower. For the purposes of this document, the main Credence application is supported by Accenture
Curricula	Used to describe a group of courses that a user may take in order to be qualified to sell a particular set of Post Office products. Post Office reference data mastered in MDM will provide Horizon with the association between a Curricula and the group of Restricted Products that this Curricula allows the user to sell
Database	A collection of records stored in a systematic way. The software used to manage and query records is known as the Database Management System. This document uses the term 'Database' to cover both meanings.
ForgeRock	ForgeRock Identity Management System
Host System	The collection of host systems including DRS, NPS, RDDS and RDMC
Hydra	Phase covering the dual-running of Horizon and HNG-X
Instance	A database instance – this is composed of memory structures and the Oracle background processes that run on a server.

FUJ00234958

Term	Definition
Node	Any device connected to a network such as a server. In the document, the term 'Node' includes the Oracle Instance.
Oracle Goldengate	Database replication software, superceded/replaced Streams as Oracle's strategic replication solution
pBlade	A processing blade which contains processors and memory, but not network or disk devices.
pServer	A logical representation of a pBlade.
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.

0.7 Changes Expected

Changes
Changes from time-to-time in subsequent versions of the all HLDs and LLDs may require changes to this document.

0.8 Accuracy

Fujitsu 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 Information Classification

The author has assessed the information in this document for risk of disclosure and has assigned an information classification of FUJITSU RESTRICTED (COMMERCIAL IN CONFIDENCE).

FUJ00234958

1 Introduction

1.1 Document Overview

This Support Guide details information in support of the Branch Database solution by documenting the operational processes that run for the application and in support of the infrastructure surrounding the application. Procedures for supporting and troubleshooting the Branch Database solution are also included.

The Branch Database has been designed to be able to fail over to a standby server in the event of a disaster but requires operator intervention because of the inherent complexity of the solution. Relevant procedures are provided for this purpose.

The Branch Support Database has been designed as a data store for support personnel. Keeping this database in step with BRDB is very important, the BRDB HLD indicates that the Branch Support Database should not lag BRDB by more than 15 minutes.

The BRDB schedule must run once for each and every calendar day. BRDB keeps a track of the current working day, in order to guarantee that data is correctly stored, processed and replicated.

Text which is highlighted in yellow like this indicates important information that should be noted.

1.2 Scope

This document is to serve as guide in support of the Branch and the Branch Support Databases. It is not a build manual, nor does it explain all the inner workings of Oracle or the operating system. Guidance for important tasks and troubleshooting scenarios are also included.

It is also to be noted that much of the detailed information for the support guide has already been documented in the associated specifications and designs. The main sources for this information are the BRDB High Level Design [DES/APP/HLD/0020], the BRSS High Level Design [DES/APP/HLD/0023] and the BRDB Low Level Design [DEV/APP/LLD/0151].

1.3 Assumptions

This Support Guide assumes the Branch Database has been successfully built and is in operation.

FUJ00234958

2 BRDB Host Processes

2.1 Approach used for Support Guide

Much of the relevant information for this section of the support guide has already been documented in the associated specifications and designs. The main source of information is:

The Branch Database High Level Design (DES/APP/HLD/0020)

The relevant information in this reference is already presented under repeating headings for the processes (i.e. the same headings for each process in turn), making it ideally suited for use as a support reference. This section of the document mainly serves to identify the relevant information, and indicate where it can be found. Pertinent information that is not covered by the existing documents has been added as appropriate.

The relevant process section of the Branch Database High Level Design is Section 7.2 - Host Processes.

For further information on the Host Processes and their integration in the overnight schedule, see Section 3 - BRDB Scheduling.

2.2 Table of BRDB Host Processes

The following table lists the current Branch Database Host processes, a brief description of each and the names of the executables used to run them. The process name corresponds to the name that is registered in table BRDB_PROCESSES and, where applicable, the name that is used to control processing via table BRDB_PROCESS_CONTROL.

No.	Executable	BRDB Process Name	Description
1.	BRDBC001	BRDBC001	Start of Day
2.	BRDBC002	BRDBC002	Message Journal Auditing
3.	BRDBX003.sh	BRDB_CNTR_REF_FROM_RDDS	BRDB Counter Reference Data from RDDS feed
4.	BRDBX003.sh	BRDB_EMDB2_INTERFACE	BRDB Estate Management Interface feed
5.	BRDBX003.sh	BRDB_DCS_TXN_TO_DRS	BRDB DCS transactions to DRS feed
6.	BRDBX003.sh	BRDB_EMDB_INTERFACE	BRDB Estate Management Interface feed. Now deprecated as of Release 20.65.
7.	BRDBX003.sh	BRDB_HOST_REF_FROM_RDDS	BRDB Host Reference Data from RDDS feed
8.	BRDBX003.sh	BRDB_MEMOS_FROM_RDDS	BRDB Desktop Memos from RDDS feed
9.	BRDBX003.sh	BRDB_NWB_TXN_TO_DRS	BRDB NWB transactions to DRS feed
10.	BRDBX003.sh	BRDB_REV_TXN_TO_NPS	BRDB Reversal Records to NPS feed
11.	BRDBX003.sh	BRDB_TT_TXN_TO_NPS	BRDB Track and Trace Records to NPS feed
12.	BRDBX003.sh	BRDB_TXN_CONF_TO_APOP	BRDB Transaction Confirmation to APOP feed
13.	BRDBX003.sh	BRDB_APS_NRT_PROCESS	Deamon task Copy AP Transaction data to BRDB_APS_MC_Txns
14.	BRDBX003.sh	BRDB_EXT_APS_COPY	Copy external AP Transaction data to BRDB_APS_MC_Txns

FUJ00234958

No.	Executable	BRDB Process Name	Description
15.	BRDBX003.sh	BRDB_SSK_TXN_TO_DRS2	Selects non-horizon transactions from BRDB_RX_DCS_TRANSACTIONS table and delivers them to DRS2.
16.	BRDBX003.sh	BRDB_ETU_TXN_TO_DRS2	Selects electronic top-up transactions from BRDB_RX_NWB_TRANSACTIONS table and delivers them to DRS2.
17.	BRDBX003.sh	BRDB_PBS_TXN_TO_DRS2	Selects all transactions from BRDB_RX_PBS_TRANSACTIONS table and delivers them to DRS2.
18.	BRDBC004	BRDBC004	Audit, Archive, Purge
19.	BRDBX005.sh	BRDBX005.sh	Gather Optimiser Statistics
20.	BRDBX006.sh	BRDBX006	File Housekeeping
21.	BRDBX007.sh	BRDB_CUMU_TXN_AGGR	Data aggregation for daily cumulative summary
22.	BRDBX007.sh	BRDB_NON_CUMU_TXN_AGGR	Data aggregation for daily summary
23.	BRDBX007.sh	OVERNIGHT_CASH_ON_HAND	Data aggregation to calculate ONCH figures.
24.	BRDBX007.sh	RAISE_FEED_DATA_EXCEPTIONS	Inserts into operational exceptions if Feed data exceptions
25.	BRDBC008	BRDBC008	Check Job Completion
26.	BRDBC009	BRDBC009	End Of Day
27.	BRDBX011.sh	BRDBX011	Updates system parameters
28.	BRDBX011.sh	BRDB_APS_NRT_PROCESS	Deamon task Copy AP Transaction data to BRDB_APS_MC_Txns
29.	BRDBX015.sh	None	Transaction correction tool
30.	BRDBX020.sh*	None	**Redundant since R2 decommissioning** File transfer for BRDB Branch Migration Status data feed
31.	BRDBX021.sh	None	**Redundant since R12.20** Pause or restart Oracle Streams propagation
32.	BRDBX030.sh	BRDBX030_INDAY	**Redundant since R2 decommissioning** Hydra XML processing (INDAY)
33.	BRDBX030.sh	BRDBX030_RECON_CATCHUP BRDBX030_RECON_NORMAL	**Redundant since R2 decommissioning** Hydra XML processing (RECON)
34.	BRDBX031.sh	BRDBX031	Reset JSN, USN and SSN
35.	BRDBC033	BRDBC033	Transaction Correction Journal Auditing
36.	BRDBX033.sh	BRDBX033_PREP_RECON_CATCHUP BRDBX033_PREP_RECON_NORMAL	**Redundant since R2 decommissioning** Hydra XML processing (RECON)
37.	BRDBX034.sh	BRDBX034	**Redundant since R2 decommissioning** Hydra - Maintain filter table of branches due to migrate and undergo 'normal' processing in BRDBX030/BRDBX033.

FUJ00234958

No.	Executable	BRDB Process Name	Description
38.	BRDBX035.sh	BRDBX035	**Redundant since R2 decommissioning** Hydra - Extracts checking version of the Branch Trading Statement report for migrating branches.
39.	GREPX001.sh	GREPX001	Create generic views for reporting
40.	GREPX002.sh	GREPX002	Create generic reports
41.	PKG_BRDB_NRT_TXN_TO_AGENT	BRDB_NRT_TXN_TO_AGENT	BRDB Near-Real Time Service Interface to Agents
42.	BRDBX036.sh	BRDBX036	Athene - performance/graphing tool
43.	BRDBX037.sh	BRDBX037 BRDB_CLR_BRANCH_DATA	BRDB Branch Closure Process
44.	BRDBC038	BRDBC038_PAF_FROM_CD BRDBC038_PAF_ADD_LOAD	PAF File Registering Daemons
45.	BRDBC038	BRDBC038_POE_FROM_POLSAP	POE File Registering Daemon
46.	BRDBC038	BRDBC038_PS_FROM_FDG BRDBC038_PG_FROM_FDG	CFD File Registering Daemons
47.	BRDBC039	BRDBC039	POE PDF Import process (invoked by BRDBC038)
48.	BRDBC040	BRDBC040	PAF Import process (invoked by BRDBC038)
49.	BRDBC051	BRDBC051_LOAD_TXNS	CFD Import Process
50.	BRDBC052	BRDBC052_TXN_ERRORS_PS BRDBC052_TXN_ERRORS_PG	CFD Error Process
51.	BRDBX053.sh	BRDBX053_POST_EXT_TXNS	CFD Posting Process
52.	BRDBX007.sh	LAST_TRADING_DATE	Set last trading date for branches in BRDB_STOCK_UNIT_ASSOCIATIONS
53.	BRDBC041	BRDB_FEED_MON	Monitor daemon feeds identified by BRDB_HOST_INTERFACE_FEEDS.USE_MONITORING = 'Y'
54.	BRDBC055	BRDBC055	Branch-Full Event Daemon
55.	BRDBC056	BRDBC056	Branch-Full End Of Day
56.	BRDBC057	BRDBC057	Items On Hand
57.	BRDBC038	BRDBC038_CR_LOAD1_BRDBC058 BRDBC038_CR_LOAD2_BRDBC058	Paystation C&R File Registering & Invocation Daemons
58.	BRDBC058	BRDBC058	Paystation C&R Processing (invoked by BRDBC038)
59.	BRDBX042.sh	BRDBX042	OGG Heartbeat process

FUJ00234958

No.	Executable	BRDB Process Name	Description
60.	ogg_monitor.sh	OGG_MONITOR	OGG process monitoring script
61.	BRDBC060	BRDBC060	NRT BAP Agent Daemon
62.	BRDBX061.sh	BRDBX061	Mails Despatch Simulator
63.	BRDBC062	BRDB_PPK_FROM_KSN	KSN Pin Pad Keys Load Process
64.	BRDBC038	BRDB_EUM_FORGEROCK_LOADER	EUM ForgeRock File Registering Daemon
65.	BRDBC066	BRDBC066	EUM ForgeRock Loader Daemon
66.	BRDBX063.sh	BRDB_TO_CREDENCE	Credence files generation
67.	BRDBX064.sh	BRDBX064.sh	Credence files zipping
68.	BRDBX065.sh	BRDBX065.sh	Credence error files reporting
69.	BRDBX068.sh	BRDB_TO_CFS, BRDB_CFS_RECON	Generate CFS interface files
70.	BRDBX069.sh	BRDBX069_BTF, BRDBX069_BTR	Zip and deliver CFS interface files
71.	BRDBX070.sh	BRDBX070.sh	Check for CFS error files
72.	BRDBC038	BRDB_AT_LOAD	ATM File Registering Daemon
73.	BRDBX067.sh	Invoked by BRDBC051	Filter ATM file and splits single ATM file into a number of text files in Oracle directory.
74.	BRDBX071.sh	BRDBX071_CREATE_BOI_FILE	Create ATM daily Report file
75.	BRDBC051	BRDBC051_LOAD_AT_TXNS	ATM Import Process
76.	BRDBX072.sh	BRDBX072_COH_FILE_TO_CWC	Cash On Hand Interface File to CWC
77.	BRDBX072.sh	BRDBX072_CIP_FILE_TO_CWC	Cash In Pouch Interface File to CWC
78.	BRDBX072.sh	BRDBX072_PAY_FILE_TO_CWC	Cash Out Payment Interface File to CWC
79.	BRDBX072.sh	BRDBX072_DEP_FILE_TO_CWC	Cash In Payment interface File to CWC
80.	BRDBC073	BRDBC073_LOAD_TC	Transaction Correction File From CFS
81.	BRDBC074	BRDBC074_PLO_LOAD	Planned Order File from CWC
82.	BRDBC075	BRDBC075_RDC_LOAD	Replenishment Delivery Notice File
83.	BRDBC076	BRDBC076_PCL_TO_CWC	Pouch collection file to CWC
84.	BRDBX083.sh	BRDBX083	BTD FRT file creation Process
85.	BRDBC084	BRDBC084	Process TA files
86.	BRDBC085	BRDBC085	Create Missing APS Directories
87.	BRDBX086.sh	BRDBX086	APS - Validate and reversals processing package
88.	BRDBX087.sh	BRDBX087	To call Client Transmission Summary package
89.	BRDBX088.sh	BRDBX088	Type X to Type G transformation
90.	BRDBX089.sh	BRDBX089	Populating reconciliation total (BRDB_APS_RECON)
91.	BRDBX090.sh	BRDBX090	APS Quarantine Report

FUJ00234958

No.	Executable	BRDB Process Name	Description
92.	BRDBX091.sh	BRDBX091	To call the BRDB APS file creation processes
93.	BRDBC092	BRDBC092_PBS_UNDO	To produce one or more Semi-colon separated files for undo payments transactions.

Table 1: Branch Processes

Note

At the time of writing, the processes/executables marked with an asterisk (*) in the table above have not yet been added to the High Level Design document, and therefore do not have the support information available for reference. They have been included here for completeness and early notification (rather than waiting until the details have been added to the design document).

Unlike other Host processes, *PKG_BRDB_NRT_TXN_TO_AGENT* does not get executed by any script in the Batch Database schedule. Instead, NRT Agents directly access the package as detailed in subsequent sections.

2.2.1 BRDB Environment Variables

The following set of environment variables are relevant for the BRDB batch users which are used by TWS when calling batch jobs. The table below is a representation of ***brdbblv1***. and includes only BRDB application related variables.

Environment Variable	Variable Value
BRDB_EXCP_USER	ORAEXCPLV/EXCPLV123
BRDB_TCT_FILE_TEMP	/app/brdb/trans/support/working
BRDB_AUDIT_FILE_TEMP	/app/brdb/trans/support/working
NCHOME	/opt/netcool
NLS_DATE_FORMAT	DD-MON-YYYY
EXPORT_DIR	/var/tmp
BRDB_TCT_AUDIT_OUTPUT	/app/brdb/trans/audit/tctaudit
BRDB_MSU_OUTPUT	/app/brdb/trans/support/reportoutput
BRDB_ARCHIVE_OUTPUT	/app/brdb/trans/support/archive
BRDB_HOST_AUDIT_OUTPUT	/app/brdb/trans/audit/hostaudit
BRDB_COUNTER_AUDIT_OUTPUT	/app/brdb/trans/audit/counteraudit
ORACLE_HOME	/u01/app/oracle/product/11.2.0/dbhome_1
OMNIHOME	/opt/netcool/omnibus
INPUTRC	/etc/inputrc
G_BROKEN_FILENAMES	1
ORACLE_SID	BRDB1
LANG	C
NETCOOL_LICENSE_FILE	27000 **IRRELEVANT**
BRDB_CONNECT_STR	BRDB
LOGNAME	brdbblv1
BRDB_SH	/app_sw/brdb/sh
HISTSIZE	1000

FUJ00234958

Environment Variable	Variable Value
REPOSITORY	/pw/stagonl/repository
LESSOPEN	|/usr/bin/lesspipe.sh %s
BRDB_MSU_WORKING	/app/brdb/trans/support/working
FAN_EVENT_LOG_DIR	/app_sw/brdb/log
BRDB_PROC	/app_sw/brdb/c
SSH_ASKPASS	/usr/libexec/openssh/gnome-ssh-askpass
BRDB_SQL	/app_sw/brdb/sql
EXCP_USER	ORAEXCPLV/EXCP123

Table 2: Branch Environment Variables

2.3 BRDB Host Processes - Overview

The BRDB Host processes and how they are implemented fall into 3 main categories:

2.3.1 Individual Programs

These are individual shell scripts or Pro*C programs that perform a specified task. Typically, they have been migrated (with minimal change) from existing Horizon processes. e.g. “Start of Day” (BRDBC001), “Audit, Archive Purge” (BRDBC004) and “File Housekeeping” (BRDBX006). They are invoked by a direct call (from a Linux shell) to an executable.

2.3.2 Interface Feeds

2.3.2.1 Host Interface Feeds

These are new for the Branch Database, and load data between the BRDB and the legacy Host systems (in both directions). There are n number of different Feeds, with each being performed by a separate, specific database package. All of the Feeds have a common interface/parameter list and are invoked via a single shell script (BRDBX003.sh). The first parameter passed to this script controls which Feed process (packaged procedure) is executed.

For example, Table of BRDB Host Processes shows that the Feed of BRDB Counter Reference Data from RDDS , is performed by a call to BRDBX003.sh with a first parameter of “BRDB_CNTR_REF_FROM_RDDS”.

The corresponding database packages are named according to the following convention:

PKG_<Feed name> e.g. BRDB_CNTR_REF_FROM_RDDS

See 2.4.1 for feed information and troubleshooting guides.

2.3.2.2 Agent Interfaces

These interfaces were introduced at HNG-X Release 3 to cater for various Near-Real Time (NRT) Service messages. SMM has been implemented as NRT interfaces within the Branch Database. Unlike other Host Interface feeds these Interfaces do not get invoked from BRDB batch schedule via shell script BRDBX003.sh; instead they get invoked directly by NRT Agents connecting to the Branch Database. Wherever applicable these interfaces re-use feed procedures and exception handling mechanisms that are common to Host Interface feeds.

FUJ00234958

2.3.3 Data Aggregations

The following data aggregation processes exist within the schedule

Aggregation Name
BRDB_CUMU_TXN_AGGR
BRDB_LAST_TRADING_DATE
BRDB_NON_CUMU_TXN_AGGR
OVERNIGHT_CASH_ON_HAND
RAISE_FEED_DATA_EXCEPTIONS

These aggregations are similar to the Interface Feeds in that different processes are invoked via a single shell script (BRDBX007.sh) with a controlling first parameter. However, they differ from the Feeds in that the program code is stored in the database as raw SQL or PL/SQL, with no corresponding database packages.

2.3.4 Support Differences

The differences between the categories outlined above will translate into variations from a support perspective. For example, issues with database links, synonyms, grants etc. may manifest as package compilation errors for the Feeds, but run-time errors for the Aggregations.

An invalid Feed package can be re-compiled for verification (before running) after certain problems have been resolved (e.g. when a missing database link has been restored). A recompilation can be performed using the “ALTER PACKAGE” command from SQL*Plus:

e.g. ALTER PACKAGE PKG_ BRDB_CNTR_REF_FROM_RDDS;

It is recommended that 'ALTER SESSION SET GLOBAL_NAMES = FALSE' is executed prior to recompiling any BRDB packages.

In contrast, an Aggregation or Pro*C executable cannot be re-validated against the database in advance, it can only be re-run.

Another difference between the categories outlined above concerns the amount of information that is output when the processes are run. The Interface Feeds and main executables (see sections 2.3.1 and 2.3.2 above) provide the option to specify a debug level in order control the amount of output from within each process/Feed. Typically, the default debug settings provide milestone information only. However, should the need arise, for example whilst investigating a possible problem, the amount of output can be easily increased via meta-data (i.e. without changing the program concerned) - the debug levels are held as numeric system parameters with a higher number (e.g. 1) producing more detailed output than a lower number (e.g. 0) - see HLD for further details.

From Support perspective, Agent Interfaces vary from Host Interface Feeds. The extent of 3rd line support required is limited within the Branch Database as operational control lies with the NRT Agents. Within the Branch Database, support will be confined to any exceptions encountered and archiving of processed messages.

The Aggregations are more limited in this respect. The mechanism that calls each Aggregation issues output and has the debug capability, but the Aggregations themselves do not.

Differences relating to the support of the program return codes when a Node/Instance failure is encountered are detailed in section 2.5.1 Program Return Code.

2.4 BRDB Host Processes – Support Details

FUJ00234958

Much of the detailed information required for support purposes is contained in the following sections of the **BRDB High Level Design**:

HLD: Section 7.2 Host Processes

This section of the HLD contains details of each of the Host Processes, and has been written with support requirements in mind. The information is presented under the following headings **for each** process:

- Application Type – indicates the programming language in which the module has been developed e.g. PL/SQL packages, Pro*C etc
- Inputs – lists the input parameters and whether they are mandatory or optional.
- Outputs – indicates the program return codes.
- Location – states the Linux directory in which the executable code resides.
- Scheduling – gives an overview of the scheduling
- Processing details – gives high level details of the processing performed, along with information on the more important and specific functionality.
- Handling Failures and Rerun ability – gives information on the likely failure conditions, plus instructions on how to proceed.

A significant part of the BRDB daily processing concerns the loading of data between the Branch database and numerous Host applications (in both directions) by the Host Interface Feeds. Because of the variety of processing involved, further details are contained in a separate section of the HLD:

HLD: Section 5.3.4 Host Interfaces

This section of the HLD contains detailed information on the data and requirements for BRDB Host Interfaces. It includes details of the data being processed, the Host applications, and how the data is selected for processing.

Although much of this information will be too detailed for initial support purposes, it is referenced here in case more detailed analysis and understanding of a process(es) is required.

2.4.1 Host Interface Feeds – additional support details

This section gives further details and support information on the Interface Feeds:

The Host Interface Feeds have been designed and written to be robust and should therefore require very little support. For example, all of the Feed processes can simply be re-run (when the underlying problem has been resolved) if they fail to complete successfully. They all write the details of any 'show-stopper' errors to the standard output, as well as logging the necessary information to the operational exceptions table (BRDB_OPERATIONAL_EXCEPTIONS). Output is also generated under normal circumstances, providing useful information on the actions performed, time taken etc.

In addition, certain foreseeable issues/events such as a Node or database instance failure have been catered for within the logic of the Feed programs and the daily schedule.

2.4.1.1 Process Control

Where relevant, the Feeds utilise the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Tables BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. This table is also used to enforce requirements such as ensuring that certain processes can only be run once for a given trading date.

FUJ00234958

2.4.1.2 FAD Hashes

As part of the high level design, the processing of the largest volumes of data has been sub-divided - into FAD hashes (currently numbering 128, ranging from 0 to 127). Under normal circumstances, the processing of the FAD Hashes is evenly distributed across the Nodes (currently numbering 4) within the Real Application Cluster (RAC).

2.4.1.3 Node/Instance Failure

If one of the Nodes or database instances goes down, the loss is automatically detected and flagged using Oracle's Fast Application Notification (FAN). FAN then allows the processing that would have normally taken place on the failed Node to be automatically re-allocated across the remaining Nodes (when the processes are re-run – see below).

Further details of the FAN event processing are contained in the HLD.

Details of how the failed Node should (when fixed) should be reintroduced to the Cluster (i.e. made available to the Host processes) are contained within the database support section of this document.

2.4.1.4 Scheduled Re-Run of Multi-Node Feeds

The daily BRDB schedule does not automatically re-run multi-node Feed processes in the event of a single or multi-node failure. If these processes/jobs were in the state of executing when a node failure is experienced they will still appear to be executing until such time as the TWS agent re-establishes communications. Operational support will be notified in the event of such a failure.

Therefore, in order to process any FAD Hashes that have been re-allocated from a failed Node, Operational support will need to be involved in any intervention.

2.4.1.5 Data Exceptions

One of the high level design assumptions was that because the Feeds load data between internal systems (to/from the Branch Access Layer and to/from the Host applications) the data being processed should be error-free. To this end, the Feeds have (where possible) been designed to perform optimally when this is the case. However, because the unexpected can (and does) happen, many of the Feeds (where appropriate) incorporate a mechanism to handle any data errors. This means loading the valid records, whilst writing any exception records to a separate exceptions table for investigation.

In order to prevent such BRDB data errors from going un-noticed, there is a job (RAISE_FEED_DATA_EXCEPTIONS) within the normal, daily schedule that highlights any such exceptions by inserting a summary record into the operational exceptions table. This record provides an alert to the SMC, and includes the following information:

- Number of interface Feeds that encountered a data exception(s)
- Total number of data exceptions
- Processing date on which the exceptions were encountered
- The names of the affected Feeds and how many exceptions each one encountered
- The name of the database table where the exceptions have been stored
- A statement/instruction to indicate that investigation is required.

It should be noted that such exceptions are DATA errors - caused by issues with the data or underlying specification of the data format - and NOT Feed errors. The presence of a data error(s) will not cause the Feed process to fail unless the quantity of such errors is significant – the allowable limit is configurable for each Feed (see 'Data Exception Thresholds' below).

The nature of this type of exception means that they are unexpected, and therefore cannot be easily fixed by a support procedure etc. The correct action from a support perspective is to notify the development

FUJ00234958

team of the situation - so that they can investigate the actual data and data specifications etc. in order to identify where the problem/discrepancy lies. They will also need to determine whether to re-process the data that could not be loaded, and if so, how it will be done.

2.4.1.6 Data Exception Thresholds

Every feed has a data exception (numeric) threshold stored in BRDB_SYSTEM_PARAMETERS identified by a parameter name of the form '<FEED NAME>_MAX_DATA_ERRORS'.

BRDBX011.sh can be used to change a threshold value e.g. the following changes the exception threshold value for the Track and Trace feed to 10,000:

```
$BRDB_SH/BRDBX011.sh -n "BRDB_TT_TXN_TO_NPS_MAX_DATA_ERRORS" -t "N" -v 10000
-r MSC  xxxxxxxxxx

Note: -r is an Optional Parameter for recording the MSC Reference through
which the Parameter is Updated.
```

2.4.2 Agent Interfaces – additional support details

This section gives further details and support information on Agent Interfaces:

The Agent Interfaces have been designed and written to be robust and should therefore require very little support. For example, if there are NRT Agent connection failures or node instance failures then NRT Agents will have to call the initialise method and continue to process NRT service messages. All procedures within the NRT Interaface return the details of any 'show-stopper' errors to the calling NRT Agent, as well as logging the necessary information to the operational exceptions table (BRDB_OPERATIONAL_EXCEPTIONS). Since Agent Interfaces are not batch jobs execution output (stdlist) is not applicable.

On Windows platforms Agent events are written to the Windows Application Event Log whilst on Linux systems Agent events are written to syslog (See DES/APP/SPG/0002 section 3.1).

2.4.2.1 Process Control

As all the procedures implemented within the package PKG_BRDB_NRT_TXN_TO_AGENT are independent, atomic and directly accessible by the NRT Agents there is no need for process control within the Branch Database for Agent Interfaces.

2.4.2.2 FAD Hashes

Similar to Host Interfaces, the processing of the largest volumes of data has been sub-divided - into FAD hashes (currently numbering 128). Under normal circumstances, the processing of the FAD Hashes is evenly distributed across the Nodes (currently numbering 4) within the Real Application Cluster (RAC).

2.4.2.3 Node/Instance Failure

If one of the Nodes or database instances goes down, the loss is automatically detected and flagged using Oracle's Fast Application Notification (FAN). The mechanism then allows the processing that would have normally taken place on the failed Node to be automatically re-allocated across the remaining Nodes.

Further details of the FAN event processing are contained in the Branch Database HLD.

Details of how the failed Node (when fixed) should be reintroduced to the Cluster (i.e. made available to the Host processes) are contained within the database support section of this document.

Details of how the NRT Agents will recover and re-connect to the Branch Database in the event of Node / Database Instance failure are contained in NRT Interface Agent High Level Design [DES/APP/HLD/0732].

2.4.2.4 AEI NRT Interface

FUJ00234958

HNG-X Counters will write NRT Service messages (triggered by AP-ADC data type AssociateNRT) to a table called BRDB_RX_NRT_TRANSACTIONS in OPS$BRDB schema of the Branch Database. These NRT messages will be set initially with a *processed_yn* value of 'N'. All such unprocessed messages will be picked up and processed, one by one, by NRT Agents.

NRT Agents – There will be four NRT Agents connecting to the Branch Database through Nodes 1|2|3|4 respectively. A NRT Agent connecting through a specific BRDB Node will connect to the Branch Database and access the AEI NRT Interface package using respective database user LVAGENTUSER{1|2|3|4}. Similarly, while processing NRT messages a NRT Agent will only process those messages allocated through FAD hash load-balancing for a particular node – this includes Node / Database Instance failure scenario also.

Processed NRT messages will be set with *processed_yn* to 'Y' and an appropriate *processed_timestamp* in BRDB_RX_NRT_TRANSACTIONS table. Such processed messages will be archived based on meta-data defined in BRDB_ARCHIVED_TABLES.

2.5 Error Logging/Notification

When an error is detected within one of the BRDB Host processes it is highlighted and logged using the following standard procedures:

2.5.1 Program Return Code

Processes that fail return a non-zero number to the calling environment. Typically, 0 represents successful completion, 1 represents a failure and 99 indicates that a Node or Instance failure has been encountered.

Note

Within the Host processes, two different mechanisms have been used to identify whether an error code encountered within a program corresponds to a Node/Instance failure:

- Dynamic - The Interface Feeds use a dynamic, meta-data driven mechanism, using BRDB_ORACLE_ERROR_CODES as a look-up table.

- 'Hard-coded' - The 2 other categories of Host process (Individual Programs and Data Aggregations) have fixed ('hard-coded') error codes within the programs.

Therefore, if another, 'new' Oracle error code is found to correspond to a Node/Instance failure (and therefore the Host processes need to return a code of 99), the support activity required will differ accordingly:

For the Interface Feeds, a new record for the error code will need to be added to the look-up table, with column INSTANCE_CONN_ERROR_YN set to 'Y'. None of the Feed programs will need to be changed.

For the other processes, the hard-coded list in each affected shell script/Pro*C program will need to be updated, and each program re-released.

2.5.2 Screen Output

Most of the BRDB Host processes will output the details of an error (what the problem is, where it was encountered etc.) to the standard output.

2.5.3 Operational Exceptions

When an error is encountered, the details are logged in table BRDB_OPERATIONAL_EXCEPTIONS, including what the error is and where and when it was encountered. Agent Interfaces also pass the exception message and Oracle database error code, if applicable, back to the calling NRT Agent.

FUJ00234958

2.5.4 Process Control

As with many existing Host applications, most of the BRDB processes use table BRDB_PROCESS_CONTROL to manage re-starting, and to control whether an invoked process should be allowed to run. This table can be queried (using SQL*Plus or TOAD) to determine when a process started and if/when it completed successfully etc. The column OPS$BRDB.BRDB_PROCESS_CONTROL.PROCESS_NAME will map to those processes listed in 2.2. This is not applicable for Agent Interfaces.

2.5.5 Feed Data Exceptions

See section 2.4.1.5 (Data Exceptions) for details.

2.6 Troubleshooting

With error logging and notification being detailed in the sections above, the other useful bit of information is that of troubleshooting failures when the reason for their failure is unclear.

In most cases the logging information displayed in stdout and the exception information available in BRDB_OPERATIONAL_EXCEPTIONS will suffice in determining the cause of a particular feed (or other scheduled job for that matter). A very useful way of determining a higher level of detail in the logging information (and possibly the exception information – however an exception is not likely to change from the original when executed a second time) is by increasing the DEBUG level of the job/feed in question. The table BRDB_SYSTEM_PARAMETERS holds a parameter for each of these which will generally be the naming convention, according to the type of job as follows: -

Feeds:	*<Feed_Name>*_DEBUG_LEVEL	e.g. BRDB_TT_TXN_TO_NPS_DEBUG_LEVEL
Jobs:	DEBUG_LEVEL_FOR_*<Job_Name>*	e.g. DEBUG_LEVEL_FOR_BRDBC001

The valid values of the debug level are from 0 to 3, 0 being default logging through to 3 for verbose.

An update to the debug level of a job or feed can be performed as follows: -

Login as a batch user or brdb, execute the following

`$BRDB_SH/BRDBX011.sh -n` *DEBUG_LEVEL_FOR_<Job_Name>* `-t N -v` *Debug_Level –r MSC Reference (Optional)*

Alternatively the following SQL update will alter the debug level:

```
UPDATE brdb_system_parameters
   SET parameter_number = <Debug_Level>
 WHERE parameter_name   = '<Job/Feed_Name>';
```

Be sure to set the parameter back to the default once the more verbose option is no longer required.

FUJ00234958

3 BRDB Scheduling

The Branch Database schedule is run each day, and controls how and when most of the processes are executed. Sections 3.1 to 3.7 describe features of the schedule as a whole, and sections 3.8 onwards describe the individual schedules that it is composed of.

3.1 Multi-Instance Batch Jobs

Scheduling HLD: Section 5.2 Common Approach for multi-instance batch jobs

The main BRDB processes are scheduled across the nodes of the Real Application Cluster (RAC). Some of these processes are simply restarted when a failure occurs, but, most are implemented with built-in delays and reruns in the case of an initial failure. This approach means that a support call is only raised when a failure condition persists i.e. after an automatic retry has been attempted.

Please note: Currently, all scheduled processes/jobs will raise an alert upon failure. Therefore in all cases Operational support will be aware of each failure and respond accordingly.

In the schedule listings from sections 3.8 onwards, only the main jobs which perform the relevant task are listed. However, they are implemented using a common schedule template consisting of the main job running on each of the four nodes, and additional jobs to perform the waiting, checking and rerunning, as per the following table.

Job Name	Job Dependency	Rerun Action
15_min_wait		
Job-Instance-1		On failure continue
Job-Instance-2		On failure continue
Job-Instance-3		On failure continue
Job-Instance-4		On failure continue
Check-Job-Instance-1	Follows 15_min_wait	
Check-Job-Instance-2	Follows 15_min_wait	
Check-Job-Instance-3	Follows 15_min_wait	
Check-Job-Instance-4	Follows 15_min_wait	
CHECK_FOR_INTRO	Follows 15_min_wait	RERUN ABENDPROMPT "One or more jobs are stuck at INTRO. Investigate before re-run."
Check-DB-Job *Job to be run on an active node*	Follows Job-Instance-1…4	On success or failure continue
15_min_wait_rerun	Follows Check-DB-Job	
Job-Instance-1-rerun	Follows Check-DB-Job	On failure continue
Job-Instance-2-rerun	Follows Check-DB-Job	On failure continue
Job-Instance-3-rerun	Follows Check-DB-Job	On failure continue
Job-Instance-4-rerun	Follows Check-DB-Job	On failure continue
Check-Job-Instance-1-rerun	Follows 15_min_wait_rerun	
Check-Job-Instance-2-rerun	Follows 15_min_wait_rerun	
Check-Job-Instance-3-rerun	Follows 15_min_wait_rerun	
Check-Job-Instance-4-rerun	Follows 15_min_wait_rerun	

FUJ00234958

Job Name	Job Dependency	Rerun Action
CHECK_FOR_INTRO_RERUN	Follows 15_min_wait_rerun	RERUN ABENDPROMPT "One or more jobs are stuck at INTRO. Investigate before re-run."
Check-DB-Job-rerun *Job to be run on an active node*	Follows Job-Instance-1…4-rerun	On failure Alert Operations
Schedule-complete	Follows Check-DB-Job, Check-DB-Job-rerun	

3.1.1 Rerunning Failed Multi-Instance Batch Jobs

If the built-in rerun of any particular multi-instance job fails then

- the cause of the failure should be resolved
- the job should be rerun on all nodes
- the associated check job should then be rerun on all nodes

3.2 Any Active Node Batch Jobs

Certain BRDB processes can be run on any node of the Real Application Cluster (RAC).

In the schedule listings from sections 3.8 onwards, only the main jobs which perform the relevant task are listed. However, they are implemented using a common schedule template consisting of the main job running on each of the four nodes, and an additional parent job to co-ordinate them, as follows:

Job Name	Job Dependency	Rerun Action
JobName		RERUN ABENDPROMPT "Unable to determine an active BRDB node. Re-run?"
JobName1	Follows JobName	STOP ABENDPROMPT "Appropriate Message"
JobName2	Follows JobName	STOP ABENDPROMPT "Appropriate Message"
JobName3	Follows JobName	STOP ABENDPROMPT "Appropriate Message"
JobName4	Follows JobName	STOP ABENDPROMPT "Appropriate Message"

In this approach, once an available node has been selected the jobs defined for the other nodes are cancelled.

3.3 Branch Database Jobs in other Schedules

(Scheduling HLD: Section 5.5 Branch Database Jobs in other schedules)

Although most of the BRDB processes are called from within the BRDB schedule, there are a number of BRDB processes called from other application TWS schedules such as RDDS. This section lists the schedules concerned.

RDDS: Scheduling HLD is DES/APP/HLD/0097

3.4 Monitoring Jobs

The BRDB schedule includes several monitoring jobs. These are jobs which raise an alert if a specified process has not been completed by a required point in time. These jobs have been collected within a single schedule, BRDB_MONITOR – see section 3.77 for details.

3.5 Repeating/Daemon Processes

The following are BRDB Host interface feeds that run as 'daemon' processes within the daily schedule:

FUJ00234958

- Guaranteed Reversals (Feed to NPS)
- Track and Trace (Feed to NPS)
- Transaction Confirmation (Feed to APOP)
- Paystation File Register (File import)
- Post&Go File Register (File import)
- Daemon Monitoring Process (Monitors selected daemon jobs e.g. Track & Trace)
- Branch-Full Event (NRT)
- Oracle Goldengate (OGG) Heartbeat (executes on one node only)
- Oracle Goldengate (OGG) Process Monitor (executes on one node only)
- BAP NRT Agent
- Track & Trace File Processing Daemon
- BRDB EUM ForgeRock File Register(File import)
- BRDB EUM ForgeRock Loader

After starting, these processes enter a cycle of 'sleep and repeat' - where they perform any necessary processing, then sleep for a pre-defined time before 'waking' and running again. Each daemon process is controlled by a separate system parameter, named after the Feed with a '_STOP_YN' suffix, as follows:

- BRDB_TT_TXN_TO_NPS_STOP_YN
- BRDB_REV_TXN_TO_NPS_STOP_YN
- BRDB_APS_NRT_PROCESS_STOP_YN
- CR_STOP_YN
- BRDB_TXN_CONF_TO_APOP_STOP_YN
- BRDB_DAEMON_MONITOR_STOP_YN
- BRDB_EOD
- PS_STOP_YN
- PG_STOP_YN
- BRDB_BRANCH_FULL_STOP_YN
- BRDBC060_STOP_YN
- BRDB_EUM_FORGEROCK_LOADER_STOP_YN
- BRDB_PLO_LOADER_STOP_YN
- BRDB_RDC_LOADER_STOP_YN
- BRDB_PCL_STOP_YN
- BRDB_PCLBI_STOP_YN
- BRDB_PDLBI_STOP_YN

When this parameter is set to 'Y' (from within the schedule using BRDBX011.sh, please refer DES/APP/HLD/0021 for more details) the daemon Feed process will stop, although **it should be noted that there will be a time delay between setting the stop flag to 'Y' and the process actually terminating**. This is because the daemon processes only check the stop flag after 'waking' from a sleep or completing processing.

File import feeds obtain their control metadata from table BRDB_EXT_INTERFACE_FEEDS.

Additional metadata concerning the feeds (e.g. sleep time) can be queried in table BRDB_HOST_INTERFACE_FEEDS as per the following:

```
SELECT interface_desc, sleep_repeat_yn, use_fad_hash_yn, sleep_repeat_secs
```

FUJ00234958


```
FROM   brdb_host_interface_feeds
WHERE  interface_feed_name = 'BRDB_TT_TXN_TO_NPS';
```

3.5.1 Node Failures

The daemon feed processes have been designed and developed to cope with node/instance failures automatically. If a FAN event occurs for a node then:

- Database Column OPS$BRBD.BRDB_OPERATIONAL_INSTANCES.IS_AVAILABLE will be set to 'N' for the failed instance
- View BRDB_FAD_HASH_CURRENT_INSTANCE will automatically redistribute the FAD_HASHes of the failed node amongst the other operational nodes.
- Each of the daemon jobs reference the view BRDB_FAD_HASH_CURRENT_INSTANCE when waking from sleep therefore the remaining operational nodes will work on any unprocessed data from the FAD_HASHes associated with the failed node.

The failed TWS job can be set to SUCC. Refer to 4.3.3 for instance recovery.

3.5.2 Manually Stopping Daemon Processes

N.B. Stopping daemon feeds could result in the breaching of one or more service level agreements.

If there is a need to stop one of the above daemons manually then running the required TWS job from the following table will accomplish this:

Feed	TWS Job
NPS Track & Trace	BRDBX011_PAUSE_NPS_TT_COPY
NPS Guraranteed Reversals	BRDBX011_PAUSE_NPS_GREV_COPY
APOP Transaction Confirmation	BRDBX011_PAUSE_APOP_TC_COPY
Paystation File Register	BRDBX011_STOP_PS
Post&Go File Register	BRDBX011_STOP_PG
Daemon Monitor	BRDBX011_PAUSE_DAEMON_MON
Branch-Full Event	BRDBX011_PAUSE_BF_TO_BLCS
OGG Heartbeat	BRDB_PSTOP_GG
OGG Process Monitor	GG_MON_STOP_02
BAP NRT Agent	BRDBX011_PAUSE_BAP_AGT
Track & Trace (Collect & Return) File Daemon	BRDBX011_STOP_CR
BRDB EUM ForgeRock	BRDBX011_STOP_EUM_LOAD
BRDB Planned Order Process	BRDBX011_STOP_PLO_LOAD
BRDB Replenishment Delivery Notice	BRDBX011_STOP_RDC_LOAD
BRDB Pouch Collection	BRDBX011_STOP_PCL
APS NRT Copy Process	BRDBX011_PAUSE_APS_NRT_PROCESS
Pouch Collection records to Arrow Stop Job	BRDBX011.sh -n BRDB_PCLBI_STOP_YN -t T -v Y
Pouch Delivery records to Arrow Stop Job	BRDBX011.sh -n BRDB_PDLBI_STOP_YN -t T -v Y

FUJ00234958

3.5.3 Manually Starting Daemon Processes

N.B. Be aware that there should only be one feed job per instance running for each daemon, ensure the jobs are NOT started more than once. Duplicate running feeds may result in a number of unexpected and unpredictable failures (TT and GREV might be subject to deadlocking for example).

If there is a need to restart a stopped daemon manually then running the required jobs (i.e. changing the start/stop flag and then restarting the daemon process on each node) from the following table will accomplish this:

Feed	TWS Job – Flag Change	TWS Job – Daemon Process
NPS Track & Trace	BRDBX011_START_NPS_TT_COPY	BRDBX003_TT_TO_NPS_1…4_NOPAGE[1]
NPS Gurarranteed Reversals	BRDBX011_START_NPS_GREV_COPY	BRDBX003_GREV_TO_NPS_1…4_NOPAGE
APOP Transaction Confirmation	BRDBX011_START_APOP_TC_COPY	BRDBX003_TC_TO_APOP_1…4_NOPAGE
Paystation File Register	N/A (BRDBC038 sets the start flag)	BRDBC038_PS_FROM_FDG
Daemon Monitor	BRDBX011_START_DAEMON_MON	BRDBC041_BRDB_DAEMON_MONITOR
Branch-Full Event	N/A(BRDBC055 sets the start flag)	BRDBC055_BF_TO_BLCS_1…4
OGG Heartbeat	N/A (BRDBX042.sh sets the start flag)	BRDB_PSTRT_GG
OGG Process Monitor	N/A (ogg_monitor.sh sets start flag)	TBC
BAP NRT Agent	N/A (BRDBC060 sets the start flag)	**BRDBC060_BAP_AGT_1…4**
Track & Trace(Collect & Return) File Daemon	N/A(BRDBC038 CR sets the start flag)	**BRDBC038_CR_LOAD2_BRDBC058**
BRDB EUM ForgeRock File Register	N/A (BRDBC038 sets the start flag)	BRDBC038_EUM_FROM_FORGEROCK
BRDB EUM ForgeRock Loader	N/A(start flag set by BRDB EUM ForgeRock File Register)	BRDBC066_EUM_FORGEROCK_LOAD
BRDB TC file Register	N/A (BRDBC038 sets the start flag)	BRDBC038_TC_FROM_CFS
BRDB TC Loader	N/A(start flag set by TC File Register)	BRDBC073_LOAD_TC
BRDB Planned Order Register	N/A (BRDBC038 sets the start flag)	BRDBC038_PLO_FROM_CWC
BRDB PLO Loader	N/A(start flag set by PLO File Register)	BRDBC074_PLO_LOAD
BRDB RDN Register	N/A (BRDBC038 sets the start flag)	BRDBC038_RDC_FROM_CWC
BRDB RDN Loader	N/A(start flag set by RDC File Register)	BRDBC075_RDC_LOAD
APS NRT Copy Process		**BRDBX011_START_APS_NRT_PROCESS**
Pouch Collection records to Arrow Stop Job		**BRDBX011.sh -n BRDB_PCLBI_STOP_YN -t T -v N**
Pouch Delivery records to Arrow Stop Job		**BRDBX011.sh -n BRDB_PDLBI_STOP_YN -t T -v N**

3.5.4 Track and Trace Feed

[1] 1..4 indicates that the job should be run concurrently on each BDB instance/node

FUJ00234958

TT transactions (in table BRDB_RX_TT_TRANSACTIONS) will be flagged with 'Y' in column PROCESSED_YN once those transactions have been inserted into the remote NPS database. Any transactions failing to be inserted due to some exception will:

- have the PROCESSED_YN flag set to 'Y' if the exception was due to some data error[2], NPS_DELIVERED_TIMESTAMP will be left as NULL to allow support groups (SMC, SSC, HOST) time to examine the exceptions before the archive/purge job removes the source rows.
- be left unprocessed if the exception is due to a network or instance failure; this allows the row to be resent once the problem has been resolved (e.g. network is back up, NPS is back up etc)

3.5.5 Guaranteed Reversals Feed

GREV transactions (in table BRDB_RX_GUARANTEED_REVERSALS) will be flagged with 'Y' in column PROCESSED_YN once those transactions have been inserted into the remote NPS database. Any transactions failing to be inserted due to some exception will:

- have the PROCESSED_YN flag set to 'Y' if the exception was due to some data error[3], NPS_DELIVERED_TIMESTAMP will be left as NULL to allow support groups (SMC, SSC, HOST) time to examine the exceptions before the archive/purge job removes the source rows
- be left unprocessed if the exception is due to a network or instance failure; this allows the row to be resent once the problem has been resolved (e.g. network is back up, NPS is back up etc)

3.5.6 Transaction Confirmation Feed to APOP

Transaction Confirmation feed to APOP differs from other Host Interface feeds in terms of transferring transactions across to the remote APOP Database. Instead of inserting transactions into a target table in the remote database the feed will invoke a PL/SQL package in the remote APOP Database and pass the required transaction details as input parameters. The call to the remote PL/SQL package is made for every unprocessed transaction on a record-by-record basis.

If a successful response is received from the remotely called package then the APOPConfirm transaction in BRDB_RX_NRT_TRANSACTIONS table will be stamped as processed:

- processed_yn flag will be set to 'Y'
- processed_timestamp will be set to systimestamp
- update_timestamp will be set to systimestamp

If an unsuccessful response is received then

- 'retry_attempts' value will be incremented by 1
- update_timestamp will be set to systimestamp

However, the transaction belonging to the unsuccessful transfer will remain unprocessed and the feed will pick the record up for transfer in its next processing cycle. If the number of re-try attempts exceeds a set threshold value, as defined by a parameter called 'BRDB_TXN_CONF_TO_APOP_RETRY_ATTEMPTS' in System Parameters, then the feed will log an exception in BRDB_HOST_INTERFACE_FEED_EXCP table. Still, the feed will continue to re-process the transaction in its every processing cycle until the remote PL/SQL package returns a successful response.

Before an APOPConfirm transaction can be transferred to the remote APOP Database the feed will perform a set of validations to ensure that the NRT Payload is valid and to ensure that all required transactional details to be passed as input parameter to the PL/SQL package are available. If any of the

[2] As defined in table OPS$BRDB.BRDB_ORACLE_ERROR_CODES where data_error_yn = 'Y'

[3] As defined in table OPS$BRDB.BRDB_ORACLE_ERROR_CODES where data_error_yn = 'Y'

FUJ00234958

validation check fails then the following attributes will be updated against the transaction and an exception will be logged in BRDB_HOST_INTERFACE_FEED_EXCP table:

- processed_yn flag will be set to 'Y'
- update_timestamp will be set to systimestamp

'processed_timestamp' column will be left as NULL to indicate that the transaction was not transferred to the remote APOP Database. Note that transactions that fail during validation checks won't be re-processed in the feed's next processing cycle i.e., retry attempt is not applicable to such transactions as no matter how many times the invalid transactions are re-processed they will fail the validation checks every time due to invalid NRT Payload content.

Detailed information on this feed is available in the low level design document DEV/APP/LLD/0050.

3.5.7 Paystation File Register

Paystation files (`PS??????????.TP_`) in `/app/brdb/trans/externalinterface/input_share` are registered by BRDBC038 and made ready for import by BRDBC051.

3.5.8 Post&Go File Register

Post&Go files (`PG??????????.TP_`) in `/app/brdb/trans/externalinterface/input_share` are registered by BRDBC038 and made ready for import by BRDBC051.

3.5.9 Daemon Monitoring process

Executable BRDBC041 (runs once per node) monitors all feeds in table BRDB_HOST_INTERFACE_FEEDS where column USE_MONITORING = 'Y'.

The common feed script BRDBX011.sh package function (PKG_BRDB_FEED_COMMON.run_feed) invokes a heartbeat function to set an update timestamp in BRDB_HOST_IF_FEEDS_MONITOR with the current UTC date/time.

BRDBC041 will abend if the following is true:

- <feed> is set to be monitored [BRDB_HOST_INTERFACE_FEEDS.USE_MONITORING = 'Y']
- The above feed has been initiated [BRDB_SYSTEM_PARAMETER.PARAMETER_TEXT = 'N' for <feed_name>_STOP_YN
- The UTC last heartbeat timestamp is earlier than (current UTC minus the (BRDB_HOST_INTERFACE_FEEDS .feed sleep_repeat_secs + BRDB_HOST_INTERFACE_FEEDS.timeout_threshold)).

3.5.10 Branch-Full Event Daemon

Executable BRDBC055 (runs once per node) polls new Branch_Full event transactions in table BRDB_RX_NRT_TRANSACTIONS where column PROCESSED_YN = 'N'.and CLIENT_NAME = 'BranchFull' and CLIENT_ROUTING_NAME = 'BLCS'.

Branch-Full event transactions (in table BRDB_RX_NRT_TRANSACTIONS) will be flagged with 'Y' in column PROCESSED_YN once those transactions have been successful inserted into the BRDB_BRANCH_FULL_EVENTS and written out to the Branch-Full event file.

BRDBC055 will abend if any transactions failing to be inserted into BRDB_BRANCH_FULL_EVENTS or written to the Branch-Full event file due a Oracle or filesystem error,

3.5.11 Oracle Goldengate Heartbeat Process

The aim of this daemon process (/app_sw/brdb/sh/BRDBX042.sh) is to induce regular 'pings' to the target replicated system (via OGG). A regular heartbeat allows replication performance to be measured and helps spot replication failures earlier than would be possible otherwise.

FUJ00234958

BRDBX042.sh logs into BRDB and invokes package OPS$BRDB.PKG_BRDB_OGG_HB. The package updates a single row in table OPS$BRDB.OGG_HEARTBEAT_SOURCE, setting the update_timestamp to the current date/time every n seconds (where n is system parameter OGG_HB_SLEEP_INTERVAL). This process will exit when system parameter OGG_HB_STOP_YN is set to 'Y'.

3.5.12 Oracle Goldengate Process Monitor

This daemon process (/u02/goldengate/poa/sh/ogg_monitor.sh) periodically obtains the status of the relevant OGG processes via OGG commandline program /u02/goldengate/ggsci. The statuses are inserted into table OPS$BRDB.BRDB_BRSS_GG_MONITORING.

3.5.13 BAP NRT Daemon

BRDBC060 (runs per node) polls new BAP transactions in table BRDB_RX_NRT_TRANSACTIONS where column PROCESSED_YN = 'N' and CLIENT_NAME = 'PS2DBarcode' and CLIENT_ROUTING_NAME = 'POLBAP'.

BAP event transactions (in table BRDB_RX_NRT_TRANSACTIONS) will be flagged with 'Y' in column PROCESSED_YN once those transactions have been successful extracted and written out to the Pre_Advice file.

BRDBC060 will abend if any transactions failing written to the Pre_Advice file due a Oracle or filesystem error.

3.5.14 EUM ForgeRock File Register Daemon

EUM ForgeRock files(FR*.XM_) in `/app/brdb/trans/externalinterface/input_share` are registered by BRDBC038 and made ready for loading by BRDBC066

3.5.15 EUM ForgeRock Loader Daemon

BRDBC066 polls the newly register files in table BRDB_FILE_AUDIT_TRAIL where column PROCESS_NAME = 'BRDB_EUM_FORGEROCK_LOADER' and FILE_STATUS = 'N'.

The file_status will set to 'C' or 'E' once the files have been loaded successful or failure respectively.

3.6 File Import Daemons (BRDBC038)

File imports are controlled by process BRDBC038 which in turn spawns child processes [BRDBC039, BRDBC040, BRDBC058] if applicable. The following are BRDB file imports that occur within the daily schedule:

- Post Office Essentials (POe) POLSAP PDF Load process (BRDB_POE_FROM_POLSAP) [invokes BRDBC039]
- Postcode Address File (PAF) Complete Load Process (BRDB_PAF_FROM_CD) [invokes BRDBC040]
- Postcode Address File (PAF) Incremental/Additional Load Process (BRDB_PAF_ADD_LOAD) [invokes BRDBC040]
- CFD Paystation File Register Daemon [register only, no invocation]
- CFD Post&Go File Register Daemon [register only, no invocation]
- Collect & Return Paystation File Register & Load process [invokes BRDBC058]
- EUM ForgeRock File Register Daemon[register only, no invocation]
- ATM Daily Withdrawals File Register and Load[register only, no invocation]
- Transaction Correction File Register and Load[register only, no invocation]
- Planned Order File Register and Load[register only, no invocation]

FUJ00234958

- Replenishment Delivery Notice File Register and Load[register only, no invocation]
- Transaction Acknowledgements Validate and Load

BRDBC038 uses the metadata stored in BRDB table BRDB_EXT_INTERFACE_FEEDS (see next section below) to control its behaviour - it can act as a daemon process (with a sleep repeat loop) or as a one off import.

Each instance of BRDBC038 will

- look in the INPUTSHARE_DIR_NAME directory for any files that fit the format mask as defined in EXT_FILENAME_SEARCH_PATTERN.
- Each relevant file is registered in BRDB_FILE_AUDIT_TRAIL
    - file is copied to AUDIT_DIR_NAME (if IS_AUDITABLE='Y')
    - file is copied to BRDB_INPUT_DIR_NAME
    - file is deleted from INPUTSHARE_DIR_NAME
- The command COMMAND_TO_RUN is invoked to process the registered files (if COMMAND_OR_SCHEDULE = 'Command').

FUJ00234958

3.6.1 BRDB_EXT_INTERFACE_FEEDS Table[4]

Column Name	Data Type	Description
EXT_INTERFACE_FEED_NAME	VARCHAR2(30)	Unique name of feed - Primary key
EXT_INTERFACE_DESC	VARCHAR2(250)	Description of interface feed
INPUTSHARE_DIR_NAME	VARCHAR2(128)	Share (source files) path
AUDIT_DIR_NAME	VARCHAR2(128)	Optional - audit directory to copy files in Share to
BRDB_INPUT_DIR_NAME	VARCHAR2(128)	Input directory to move files from Share into
BRDB_LOAD_DIR_NAME	VARCHAR2(128)	Local working directory accessible by Oracle [dir BRDB_LOAD_DIR]
OUTPUT_SHARE_DIR_NAME	VARCHAR2(128)	Share (output files) path
BRDB_OUTPUT_DIR_NAME	VARCHAR2(128)	Output directory to move files into share from
EXT_FILENAME_SEARCH_PATTERN	VARCHAR2(128)	String to search for files in sInputShareDir
COMMAND_OR_SCHEDULE	VARCHAR2(8)	Issue command or generate schedule Value: Command – Invoke COMMAND_TO_RUN Value: Schedule – Do not invoke any command, leave to TWS
COMMAND_TO_RUN	VARCHAR2(200)	Invoke Path + executable Note if sExecutePerFile = Y then invoke Path + executable + path_of_file/filename
REMOTE_APPLICATION	VARCHAR2(8)	Description of remote application (e.g. POLSAP)
PROCESSED_SUFFIX	VARCHAR2(3)	File extension to rename existing extension once processing is completed on a file
SLEEP_REPEAT_YN	VARCHAR2(1)	Daemon (sleep and loop) or execute once flag Value: Y – Daemon feed

[4] Extracted from DEV/APP/LLD/1354

FUJ00234958

Column Name	Data Type	Description
		N — Process is invoked once by TWS
EXECUTE_PER_FILE	VARCHAR2(1)	Execute command for each file found or at end
WAIT_FOR_SCHEDULE_COMPLETE	VARCHAR2(1)	Wait for schedule job to finish before creating next job
IS_AUDITABLE	VARCHAR2(1)	Copy file to audit directory Y or N Y — Copy appropriate files in SHARE to audit dir N — Skip copying to audit dir
SLEEP_REPEAT_SECS	NUMBER(5)	Time to sleep between iterations for a daemon feed. Time to sleep when looking for at least 1 file to process in a non-daemon feed.
ALERT_AFTER_SECS	NUMBER(5)	Number of iterations without finding a file to process before recording exception 0 — No exception logged if zero files found > 0 — Log exception if non-daemon process and zero files found within timeframe -1 — Log exception if daemon process and zero files found on exit of loop

FUJ00234958

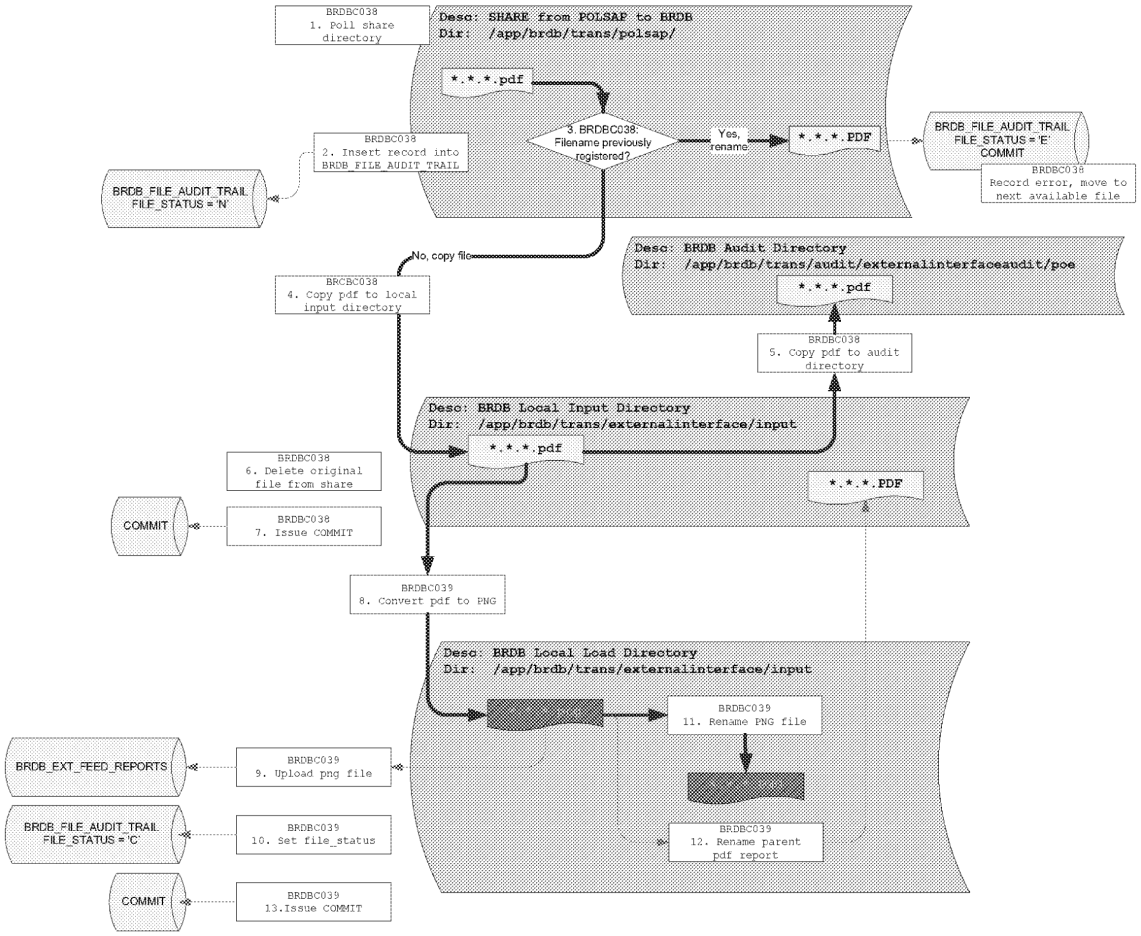
3.6.2 Single Node Job

File import daemons are designed to run on only one node at any one time (See 3.2)

3.6.3 Post Office Essentials [BRDBC039]

POLSAP PDF files are made available to BRDB via a share. BRDBC038 registers all relevant PDF files first and then invokes BRDBC039 which

- Loops through files in BRDB_FILE_AUDIT_TRAIL (where process_name = 'BRDB_POE_FROM_POLSAP' and file_status = 'N')
- converts each PDF to one or more PNG files (1 PNG for each PDF page)
- uploads each PNG file into BRDB table OPS$BRDB.BRDB_EXT_FEED_REPORTS
- sets the column FILE_STATUS in BRDB_FILE_AUDIT_TRAIL to 'C' (complete)
- exceptions are logged in OPS$BRDB.BRDB_HOST_INTERFACE_FEED_EXCP
- each processed file (whether PDF or PNG) has its extension uppercased in order to allow BRDB housekeeping to remove after an appropriate period of time has elapsed.



FUJ00234958

3.6.3.1 External Feed Metadata

COLUMN NAME	DESCRIPTION	VALUE
Note: This metadata is stored in BRDB_EXT_INTERFACE_FEEDS, identified by the row "WHERE ext_interface_feed_name = 'BRDB_POE_FROM_POLSAP'		
INPUTSHARE_DIR_NAME	file source share	/app/brdb/trans/polsap
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/input
AUDIT_DIR_NAME	BRDB audit directory	/app/brdb/trans/audit/externalinterfaceaudit/poe
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
EXT_FILENAME_SEARCH_PATTERN	File wildcard	*.*.*.pdf
COMMAND_TO_RUN	Command that BRDBC038 runs	$ $BRDB_PROC/BRDBC039
EXECUTE_PER_FILE	Child process exec per file?	N
REMOTE_APPLICATION	Data description	POLSAP
PROCESSED_SUFFIX	File post-process suffix indicator	PDF

3.6.4 BRDB Postcode Address File Complete [BRDBC040]

3.6.4.1 Process Overview

The means by which the Post Office queries British postcodes via the Counter, was through the solution known as QAS. QAS was hosted on an Apache Web Server (Windows Server) in the datacentre and the data provided through a service.

BRDB PAF is known as *PAF Replacement* because it replaced the previous solution (provided by an external provider) with an in-house solution accessed by the counter directly within the Branch Database.

The *Load Process* at a very high level does in essence: -

- Find and validate files
- Prepare the table and load the data
- Ready the table for access by the estate and complete.

It is important to note that there are two PAF tables. The main table, referred to as PAF_ADDRESS_POINT and then a secondary table, PAF_ADDRESS_POINT_SAV which holds the data from the previous load iteration of the load process. When the load process is therefore envoked, the older table is prepared and loaded such that, should there be a failure of any kind during the load process, the risk to the estate of not being able to access PAF data is non existent.

3.6.4.2 Process Execution and Flow

BRDBC040 gets executed by the BRDBC038 parent process (see section 3.6). *BRDB_PAF_FROM_CD* is the "external feed" identifier for *BRDB PAF Complete* and is specifically executed as a process when the following call is made: -

```
${BRDB_PROC}/BRDBC038 BRDB_PAF_FROM_CD ^BRDBBDAY^
```

Section 3.6 details the activities of BRDBC038, but for completeness it is mentioned here too. BRDBC038, in the context of *BRDB PAF Complete* (please see the table below – section 3.6.4.4 - for all metadata values, including file formats, directories, et cetera) has the following logic flow: -

i. It looks for the files in the *INPUTSHARE_DIR_NAME* directory, of the form defined for *EXT_FILENAME_SEARCH_PATTERN*, which in this case is: **`*compstc*.*.paf`**

ii. For every file found: -
   a. It registers the file in the table **`BRDB_FILE_AUDIT_TRAIL`** with a **`file_status`** of **`'N'`** (for New)

FUJ00234958

b. Copies the file from the source directory (see i. above) to the *BRDB_INPUT_DIR_NAME* directory
c. Only once all files are successfully complete, will the transaction commit, i.e. *all* files will either show a file_status of **'N'** or there will be no record at all

iii. It then executes BRDBC040 using the command-line call in *COMMAND_TO_RUN* , which in this case is (see also section 3.79.1): -

```
${BRDB_PROC}/BRDBC040 BRDB_PAF_FROM_CD
```

iv. BRDBC040 then using the file-metadata found in **BRDB_FILE_AUDIT_TRAIL** will verify that all file headers *and* all file trailers are valid and expected

v. It then prepares the database table PAF_ADDRESS_POINT_SAV for loading by: -
a. Truncating the table and …
b. Removing the primary key and all remaining indexes

vi. It then calls the PAF Importer (pafimporter.jar Java program) which loads the data (~30 million rows) one file at a time.

The importer can be configured using loader properties found in **/app_sw/brdb/java/paf/config/pafimport.properties** such as commit size, amoungst others. The importer also uses a posttown-to-county mapping file (**/app_sw/brdb/java/paf/config/post_town_counties_mapping.csv**) when importing the PAF data in order to populate county data correctly, which is not likely to create any problems but is merely mentioned here for completeness.

BRDBC040 passes three parameters to the PAF Importer: -
- The type of load, in this case a "full" load ('M')
- The absolute path of the file to load (executed in order at a global level)
- The table in which to load the data

vii. It then, having loaded all files successfully, will insert all records from **PAF_ADDRESS_POINT** into **PAF_ADDRESS_POINT_SAV**, previously added by an execution of the *PAF Additional* process (see Section 3.6.5) prior to the execution of this process. This insert will include a SQL query based on the following predicate:

```
... WHERE additional_data = 'T'
```

viii. It then performs some post-load processing to finalise the PAF table for access by the estate, this includes: -
a. Creating the primary key and all other indexes (of which there are 14; with the PK, 15)
b. Analyzing the table, providing Oracle with accurate statistics.
c. Updating BRDB metadata in BRDB_SYSTEM_PARAMETERS with the value of the current LIVE synonym. The parameter is called *PAF_TABLE_SET* and will have a value of 'A' or 'B', depending on whichever table is the live table.
d. Finally, the synonyms that dictate which table is primary and which the secondary, are then switched. In this case the secondary table is loaded and then becomes the primary at the end of the process, i.e. the synonym switch is the very last step.

e.g. Assume that the **PAF_ADDRESS_POINT_A** table is the table being loaded (this is the case if the **PAF_ADDRESS_POINT_SAV** synonym references this table). When the switch occurs, the **PAF_ADDRESS_POINT_A** table is assigned the **PAF_ADDRESS_POINT** synonym and the "B" table (former primary) the **PAF_ADDRESS_POINT_SAV** synonym.

ix. BRDBC040 then finishes by setting the **file_status** for all files to **'C'** and completes, handing control back to BRDBC038.

3.6.4.3 PAF Load Process - Failure and Recovery

FUJ00234958

BRDBC040 is *not* re-runnable. There are a number of reasons for this, the most important of which is the fact that this process deals with files which are delivered by an *external* party. Therefore the cause of the failure must be determined in order to find the best possible set of recovery actions to perform, including the possibility that the files are corrupt or that they contain erroneous data.

Should there be a failure during this load process, the TWS *stdout* job log will be required in order to determine what the next step should be in order to get the PAF data loaded with the least amount of hassle as possible.

Note: In most cases though, the understanding (at the time of writing) is that the process is likely to be started again from the beginning (see Section 3.6.6) in preparation for the BRDB scheduled processes to pick-up and re-process the files once again.

3.6.4.3.1 Failure Scenario 1 – Pre-load Failure

Scenario 1 assumes that the PAF Importer (Java program) has not yet been called by BRDBC040 and a failure occurs:

i. The TWS job log will be required to determine the point of failure.

ii. The likely causes are: -
   a. Available space in the *BRDB_INPUT_DIR_NAME* (see 3.6.4.4) directory has been exhausted. Solution: Free up disk space for the required files.
   b. The file being read has been removed mid-execution. Solution: Find the cause of the removal.
   c. The files being loaded have already been loaded, i.e. as every file name should be unique, if files are named the same as files which have been loaded before, the process will skip and inform of a duplicate. Solution: Determine the reason for the file being duplicated. This should never be the case unless files are manually created/renamed.

3.6.4.3.2 Failure Scenario 2 – PAF Loader (Java code) Failure

Scenario 2 assumes that the load process has been running for a length of time and having loaded 1 or more files, fails:

i. The TWS job log will be required to determine the error

ii. The likely causes are: -
   a. Available space in either the PAF_DATA or PAF_INDEX tablespaces has been exhausted. Solution: Increase the size of the tablespaces
   b. An erroneous record has been read by the PAF Importer. Solution: An exercise to determine the erroneous record will be required. Activities in this regard could include comparing the failed file to that of a previous (successful) month.
   c. The file being read has been removed mid-execution. Solution: Find the cause of the removal.

3.6.4.3.3 Failure Scenario 3 – Post-load Failure

Scenario 3 assumes that the load process has completed successfully with all data having been loaded. As the post-load process is an Oracle PL/SQL procedure:

i. The TWS job log will be required to determine the Oracle error.

ii. The likely causes are: -
   a. Available space in either the PAF_INDEX or BRDB_TEMP4 tablespaces has been exhausted. Solution: Increase the size of either of the tablespaces.
   b. An unexpected Oracle error occurred. Solution: Once the error is known, and the appropriate advice from the DBA Support or Host Development teams has been sought, the appropriate task to correct the error can be undertaken.
   c. The associated BRDB instance either crashed or was mistakenly shutdown during the process. Solution: Startup the instance.

FUJ00234958

3.6.4.3.4 Recovery Tasks

Following a failure of BRDBC040, a number of tasks will be required, the first of which are described in the sections prior to this. It is important to know: -

i. In the first instance why the PAF Load process failed (see all point i.'s above) and …

ii. Thereafter determining the extent to which the job had completed, e.g. which of the above failure scenarios is applicable.

Once the failure is known, the likely recovery task(s) would include analysis and investigation (initially by Development) and then actions on the LIVE server to follow; the solutions to most of which, are detailed in the above scenarios.

Ultimately though, the re-running of the Load process will need to occur and the following is a set of guidelines and tasks to complete in order to successfully re-run BRDBC040. Invariably all failure scenarios and subsequent recovery will include a combination of the following sections.

3.6.4.3.4.1 Scenarios Regarding File Processing

In all failure scenarios, the table **`BRDB_FILE_AUDIT_TRAIL`** will show a **`file_status`** of **`'E'`** ('Errorred') for all files processed in that run of the PAF Loader. The following SQL will help show file status' (change accordingly - in the SQL below - to the date the job ran): -

```
SELECT file_name, file_status
  FROM ops$brdb.brdb_file_audit_trail
 WHERE process_name = 'BRDB_PAF_FROM_CD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%'
 ORDER BY status_change_timestamp;
```

In every scenario then, the files will either be in the source directory (*INPUTSHARE_DIR_NAME*) or the input directory (*BRDB_INPUT_DIR_NAME*) and an entry for each file will exist in the database. Therefore in order to re-run the process: -

1. Either the conditions at which the original process ran, need to be re-created
   a. All files need to be located and moved back to the *source* directory ensuring that the file extentions are all **`*.paf`** and not **`*.PAF`**
   b. The file entries (in **`BRDB_FILE_AUDIT_TRAIL`**) for this particular instance of BRDBC040 must be removed

```
DELETE
  FROM ops$brdb.brdb_file_audit_trail
 WHERE process_name = 'BRDB_PAF_FROM_CD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%';
```

2. Or artificial conditions need to be created and BRDBC040 manually re-run
   a. All files in the *input* directory need to be located and then ensure that file extentions are all **`*.paf`** and not **`*.PAF`**
   b. The file entries (in **`BRDB_FILE_AUDIT_TRAIL`**) *for this particular instance of BRDBC040*, setting **`file_status`** to **`'N'`** ('New')

```
UPDATE ops$brdb.brdb_file_audit_trail
   SET file_status = 'N'
 WHERE process_name = 'BRDB_PAF_FROM_CD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%';
```

   c. Manually execute BRDBC040 as specified against point (iii.) of section 3.6.4.2

3.6.4.3.4.2 Scenarios Regarding Data Loading

As above, in all failure scenarios, the table **`BRDB_FILE_AUDIT_TRAIL`** will show a **`file_status`** of **`'E'`** ('Errorred') for all files processed in that run of the PAF Loader. The **`PAF_ADDRESS_POINT_SAV`** table will either be partially, or fully populated or not at all.

FUJ00234958

This section is relevant to *Failure Scenarios 2 or 3* above. Therefore in order to re-run the process: -

1. Either the table is partially populated, in which case a re-run of the process (referring to section 3.6.4.3.4.1) is required.
2. Or the table is completely and correctly populated. In order to not have the initial load process repeated (and waste time and resource repeating it), manual actions to complete the process are recommended: -
    a. Determine which table the synonym PAF_ADDRESS_POINT_SAV currently references:

```
SELECT table_name
  FROM all_synonyms
 WHERE synonym_name = 'PAF_ADDRESS_POINT_SAV';
```

    b. Check to see whether the table has had any indexes created on it.

```
SELECT COUNT(1)
  FROM all_indexes
 WHERE table_name = '<TABLE_FROM_ABOVE_SQL>'
   AND owner      = 'PAF_OWNER';
```

    c. ***If (b) is NO*** and in order to *not* have the entire load process repeated, execute the following to complete the process: -
        i. Create a dummy index:

```
CREATE INDEX paf_owner.pap_x_c_ind
  ON paf_owner.<TABLE_FROM_ABOVE_SQL> (county)
  TABLESPACE paf_index INITRANS 32
  STORAGE (BUFFER_POOL KEEP) UNUSABLE;
```

    d. ***If (b) is YES*** then execute the post-load process (***as brdbblv4 on BRDB4***): -

```
EXEC paf_owner.pkg_brdb_paf_common.post_paf_dataload;
```

    e. Update all file entries (in **`BRDB_FILE_AUDIT_TRAIL`**) for this particular instance of BRBC040, setting **`file_status`** to **`'C'`** ('Complete'): -

```
UPDATE ops$brdb.brdb_file_audit_trail
   SET file_status = 'C'
 WHERE process_name = 'BRDB_PAF_FROM_CD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%';
```

3.6.4.3.4.3 Switching Synonyms

This section details the switching of the PAF table synonyms in the event this task is required. It is highly unlikely that this section will ever be used. However in a scenario where it is found that the full data having just been loaded is in some way causing a problem or is corrupt, then the following commands would help in enabling a synonym switch, effectively allowing the former LIVE (now secondary) table to be made LIVE (primary) again:

1. First determine which PAF table is being referenced as the primary table: -

```
SELECT synonym_name, table_name
  FROM all_synonyms
 WHERE table_owner = 'PAF_OWNER'
   AND synonym_name LIKE 'PAF_ADDRESS%';
```

2. Then update the BRDB metadata to reflect the change to *new primary*: -

```
UPDATE ops$brdb.brdb_system_parameters
   SET parameter_text = '<LIVE_TABLE_SUFFIX>'
 WHERE parameter_name = 'PAF_TABLE_SET';
```

FUJ00234958


```
e.g. ...   SET parameter_text = 'A'
```

3. Then make the switch by first changing the secondary to the primary and then visa-versa: -

```
CREATE OR REPLACE PUBLIC SYNONYM paf_address_point
   FOR paf_owner.<SECONDARY_TABLE_FROM_ABOVE>;

CREATE OR REPLACE PUBLIC SYNONYM paf_address_point_sav
   FOR paf_owner.<PRIMARY_TABLE_FROM_ABOVE>;
```

3.6.4.4 External Feed Metadata

COLUMN NAME	DESCRIPTION	VALUE
Note: This metadata is stored in BRDB_EXT_INTERFACE_FEEDS, identified by the row "WHERE ext_interface_feed_name = 'BRDB_PAF_FROM_CD'".		
INPUTSHARE_DIR_NAME	PAF file source directory (DAT)	/app/brdb/trans/support/working
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/input
AUDIT_DIR_NAME	BRDB audit directory	N/A
BRDB_LOAD_DIR_NAME	BRDB PAF load directory	/app/brdb/trans/externalinterface/loaddir
EXT_FILENAME_SEARCH_PATTERN	PAF file wildcard	*compstc*.*.paf
COMMAND_TO_RUN	Command that BRDBC038 runs	${BRDB_PROC}/BRDBC040 BRDB_PAF_FROM_CD
EXECUTE_PER_FILE	BRDBC038 number of executions	N
REMOTE_APPLICATION	Data description	POLPAFM
PROCESSED_SUFFIX	File post-process suffix indicator	PAF

FUJ00234958

3.6.5 BRDB Postcode Address File Additional [BRDBC040]

The majority of the information in Section 3.6.4 *BRDB Postode Address File Complete* is applicable here, however this section is concerned more with the information pertaining to the *Additional* Load Process.

3.6.5.1 Process Overview

This process differs from the *PAF Complete* process in that the main table that this particular process accesses is PAF_ADDRESS_POINT and is the LIVE table used by the Counter. This process does not reference or work with the PAF_ADDRESS_POINT_SAV table in any way.

3.6.5.2 Process Execution and Flow

The *PAF Additional* process *adds* data to PAF_ADDRESS_POINT table when required. This process is triggered when an *additional* file is found by the process

As in Section 3.6.4, BRDBC040 gets executed by the BRDBC038 parent process. However, *BRDB_PAF_ADD_LOAD* is the "external feed" identifier for *BRDB PAF Additional*. As in the case of *PAF Complete*, it is executed as a process when the following call is made: -

```
${BRDB_PROC}/BRDBC038 BRDB_PAF_ADD_LOAD ^BRDBBDAY^
```

BRDBC038, in the context of *BRDB PAF Additional* (see table below in Section 3.6.5.4 for related metadata) has the following logic flow: -

i. It looks for the files in the *INPUTSHARE_DIR_NAME* directory, of the form defined for *EXT_FILENAME_SEARCH_PATTERN*, which in this case is: **`*compstd*.*.paf`**

ii. There is expected to ever only be a single file for every execution of this job. When the file is found: -
   a. It registers the file in the table **`BRDB_FILE_AUDIT_TRAIL`** with a **`file_status`** of **`'N'`**
   b. Copies the file from the source directory to the *BRDB_INPUT_DIR_NAME* directory
   c. Creates an additional copy of the file in the *AUDIT_DIR_NAME* directory
   d. Only once the fileis successfully complete, will the transaction commit, i.e. an entry will either show a file_status of **`'N'`** or not at all

iii. It then executes BRDBC040 using the command-line call in *COMMAND_TO_RUN* , which in this case is (see also section 3.80.1): -

```
${BRDB_PROC}/BRDBC040 BRDB_PAF_ADD_LOAD
```

iv. BRDBC040 then using the file-metadata found in **`BRDB_FILE_AUDIT_TRAIL`** will verify that the file header *and* it's trailer is valid and expected

v. It then calls the PAF Importer (pafimporter.jar Java program) which: -

   a. Will delete all records in **`PAF_ADDRESS_POINT`** added by a previous execution of a *PAF Additional* process (between the last execution of *PAF Complete* and now). This delete is based on the following predicate:

```
... WHERE additional_data = 'T'
```

   b. Will then Load only the new records found in the *PAF Additional* file.

   BRDBC040 passes three parameters to the PAF Importer for *PAF Additional*: -
   - The type of load, in this case a "additional" load ('D')
   - The absolute path of the file to load (executed at a global level)
   - The table in which to load the data

FUJ00234958

NOTE:

- All subsequent *additional* files should be cumulative, i.e. should include all data delivered by POL in previous *additional* files.
- Any subsequent *PAF Complete* loads should always include all data previously delivered by POL in *additional* files.

vi. BRDBC040 then finishes by setting the `file_status` for all files to `'C'` and completes, handing control back to BRDBC038.

3.6.5.3 PAF Load Process - Failure and Recovery

BRDBC040 is *not* re-runnable. Should there be a failure during this load process, the TWS *stdout* job log will be required in order to determine what the next step should be in order to get the PAF data loaded.

Note: In most cases though, the understanding (at the time of writing) is that the process is likely to be started again from the beginning (see Section 3.6.6) in preparation for the BRDB scheduled processes to pick-up and re-process the file(s) once again.

3.6.5.3.1 Failure Scenario 1 – Pre-load Failure

Scenario 1 assumes that the PAF Importer (Java program) has not yet been called by BRDBC040 and a failure occurs:

i. The TWS job log will be required to determine the point of failure.

ii. The likely causes are: -
   a. Available space in the *BRDB_INPUT_DIR_NAME* (see 3.6.4.4) directory has been exhausted. Solution: Free up disk space for the required files.
   b. The file being read has been removed mid-execution. Solution: Find the cause of the removal.
   c. The file being loaded has already been loaded. Solution: Determine the reason for the file being duplicated. This should never be the case unless the file was manually created/renamed.

3.6.5.3.2 Failure Scenario 2 – PAF Loader (Java code) Failure

Scenario 2 assumes that the load process has been executed and fails:

i. The TWS job log will be required to determine the error

ii. The likely causes are: -
   a. Available space in either the PAF_DATA or PAF_INDEX tablespaces has been exhausted. Solution: Increase the size of the tablespaces
   b. The DELETE of records in the table (marked additional_data = 'T') has failed. Solution: See following section.
   c. The INSERT of records in the table has failed (similar to (a.) above). Solution: See following section.
   d. An erroneous record has been read by the PAF Importer. Solution: An exercise to determine the erroneous record will be required. Activities in this regard could include comparing the failed file to that of a previous (successful) month.
   e. The file being read has been removed mid-execution. Solution: Find the cause of the removal.

3.6.5.3.3 Recovery Tasks

Following a failure of BRDBC040, a number of tasks will be required, the first of which are described in the sections prior to this. It is important to know: -

i. In the first instance why the PAF Load process failed (see all point i.'s above) and …

ii. Thereafter determining the extent to which the job had completed, e.g. which of the above failure scenarios is applicable.

FUJ00234958

Once the failure is known, the likely recovery task(s) would include analysis and investigation (initially by Development) and then actions on the LIVE server to follow; the solutions to most of which, are detailed in the above scenarios.

Ultimately though, the re-running of the Load process will need to occur and the following is a set of guidelines and tasks to complete in order to successfully re-run BRDBC040.

3.6.5.3.3.1 Scenarios Regarding File Processing

In all failure scenarios, the table **BRDB_FILE_AUDIT_TRAIL** will show a **file_status** of **'E'** ('Errorred') for any files processed in that run of the PAF Loader. The following SQL will help show file status' (change accordingly - in the SQL below - to the date the job ran): -

```
SELECT file_name, file_status
  FROM ops$brdb.brdb_file_audit_trail
 WHERE process_name = 'BRDB_PAF_ADD_LOAD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%'
 ORDER BY status_change_timestamp;
```

In every scenario then, the files will either be in the source directory (*INPUTSHARE_DIR_NAME*) or the input directory (*BRDB_INPUT_DIR_NAME*) and an entry for each file will exist in the database. Therefore in order to re-run the process: -

1. Either the conditions at which the original process ran, need to be re-created
   a. The file needs to be located and moved back to the *source* directory ensuring that it's file extention is ***.paf** and not ***.PAF**
   b. The file entry (in **BRDB_FILE_AUDIT_TRAIL**) for this particular instance of BRDBC040 must be removed

```
DELETE
  FROM ops$brdb.brdb_file_audit_trail
 WHERE process_name = 'BRDB_PAF_ADD_LOAD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%';
```

2. Or artificial conditions need to be created and BRDBC040 manually re-run. As *PAF Additional* processes a single file, the benefits of leaving just that single file in the target directory are outweighed by the benefits of a clean run (as in 1. above).
   a. The file in the *input* directory needs to be located and ensure that it's extention is ***.paf** and not ***.PAF**
   b. The file entry (in **BRDB_FILE_AUDIT_TRAIL**) *for this particular instance of BRDBC040*, setting **file_status** to **'N'** ('New')

```
UPDATE ops$brdb.brdb_file_audit_trail
   SET file_status = 'N'
 WHERE process_name = 'BRDB_PAF_ADD_LOAD'
   AND file_name LIKE '%<TODAY YYYYMMDD>%';
```

   c. Manually execute BRDBC040 as specified against point (iii.) of section 3.6.5.2

3.6.5.3.3.2 Scenarios Regarding Data Loading

As above, in all failure scenarios, the table **BRDB_FILE_AUDIT_TRAIL** will show a **file_status** of **'E'** ('Errorred') for the file processed in that run. The **PAF_ADDRESS_POINT** table will have *additional data*, either partially deleted or inserted or neither (old data still exists).

This section is relevant to *Failure Scenarios 2* above. Therefore in order to re-run the process: -

1. Either the table still has all the additional data from the previous run populated, in which case a re-run of the process is required.
2. Or the table is partially populated, in which case a re-run of the process is required. Counting the number of records which need to be deleted is not the best of ideas, but will give an indication of

FUJ00234958

the extent to which the process failed; whether that was a failure of the delete or the insert it is difficult to tell without the TWS stdout log/evidence: -

```
SELECT COUNT(1)
  FROM paf_address_point
 WHERE additional_data = 'T';
```

3.6.5.4 External Feed Metadata

COLUMN NAME	DESCRIPTION	VALUE
Note: This metadata is stored in BRDB_EXT_INTERFACE_FEEDS, identified by the row "WHERE ext_interface_feed_name = 'BRDB_PAF_ADD_LOAD'".		
INPUTSHARE_DIR_NAME	PAF file source directory (DAT)	/app/brdb/trans/support/working
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/input
AUDIT_DIR_NAME	BRDB audit directory	/app/brdb/trans/audit/externalinterfaceaudit/paf
BRDB_LOAD_DIR_NAME	BRDB PAF load directory	/app/brdb/trans/externalinterface/loaddir
EXT_FILENAME_SEARCH_PATTERN	PAF file wildcard	*compstd*.*.paf
COMMAND_TO_RUN	Command that BRDBC038 runs	${BRDB_PROC}/BRDBC040 BRDB_PAF_ADD_LOAD
EXECUTE_PER_FILE	BRDBC038 number of executions	N
REMOTE_APPLICATION	Data description	POLPAFD
PROCESSED_SUFFIX	File post-process suffix indicator	PAF

3.6.6 BRDB Postcode Address File – End-to-End Process

This section exists to give background information on the *current* end-to-end process; from receiving the files from the Post Office to the final data load.

The process is as follows:

1. POL to Refdata: Fujitsu receives the files from the Post Office on a CD in compressed format
2. Refdata: The Reference Data team "unpack" the files into a format recognised by the DAT Host process (*.gz) that will copy the files.
3. Refdata to DAT: The files are then manually copied to a local SAMBA share which is mounted to the DAT server. The target directory on the DAT server is specified as **`/bvnw01/rdmc/Z_PAF`**. This becomes the source for the next step.
4. DAT to BRDB: A script (**`paf_copy.ksh`**) is then executed, which will unzip the files, rename them to a filename format expected by the BRDB TWS Schedule and then copies them to a separate, but locally mounted NAS share specified as **`/nas/brdb_sup/working`**. This share is a NAS share and as such is mounted locally mounted on all nodes of the BRDB cluster as **`/app/brdb/trans/support/working`**. As mentioned in previous sections, this directory is seen by **`BRDBC040`** as the *INPUTSHARE_DIR_NAME* and is the directory from which the process finds the files to process.
5. BRDB: When the PAF-related TWS Scheduled jobs are executed the files are "picked up" and processed as described above.

3.6.7 Client File Delivery [CP0605]

File based transactions produced by external terminals (e.g. Paystation) are

FUJ00234958

- Placed in BRDB_INPUT_DIR_NAME (see 'External Feed Metadata' below) by PODG
- registered via BRDBC038
- validated & staged via BRDBC051
- returned to the originator via BRDBC052 i.e. validation errors are returned to 3rd party providers via FTMS
- Updated by BRDBX003.sh BRDB_XDATA_TXN_TO_PS for Paystation APS records
- posted to BRDB via BRDBX053.sh

3.6.7.1 Paystation External Feed Metadata

COLUMN NAME	DESCRIPTION	PAYSTATION VALUE
INPUTSHARE_DIR_NAME	PODG drop location	/app/brdb/trans/externalinterface/input_share
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/externaltxns
AUDIT_DIR_NAME	BRDB audit directory	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
OUTPUTSHARE_DIR_NAME	PODG pickup location	/app/brdb/trans/externalinterface/output_share
BRDB_OUTPUT_DIR_NAME	BRDB local output	/app/brdb/trans/externalinterface/output
EXT_FILENAME_SEARCH_PATTERN	File wildcard	PS??????????.TP_
REMOTE_APPLICATION	Data description	PS
PROCESSED_SUFFIX	File post-process suffix	TPP

3.6.7.2 Paystation Preprocessor Command

awk -f $BRDB_SH/PS.awk -v OUTDIR=#OUTDIR# #INPUTDIR#/#FILENAME#

3.6.7.3 Post&Go External Feed Metadata

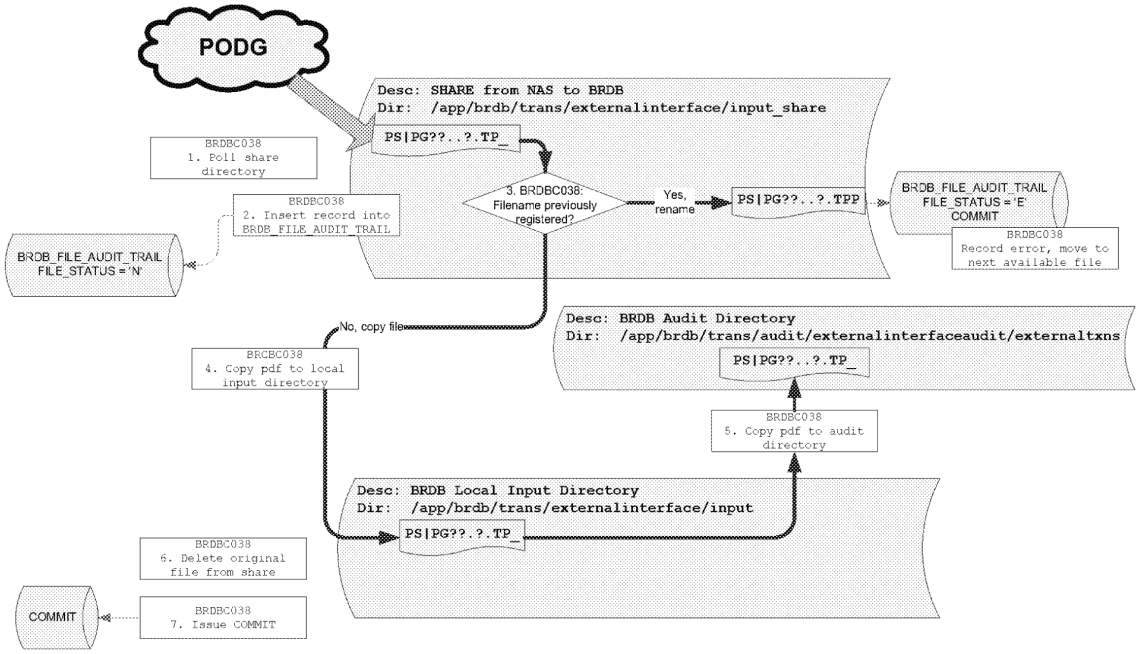
COLUMN NAME	DESCRIPTION	POST&GO VALUE
INPUTSHARE_DIR_NAME	PODG drop location	/app/brdb/trans/externalinterface/input_share
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/externaltxns
AUDIT_DIR_NAME	BRDB audit directory	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
OUTPUTSHARE_DIR_NAME	PODG pickup location	/app/brdb/trans/externalinterface/output_share
BRDB_OUTPUT_DIR_NAME	BRDB local output	/app/brdb/trans/externalinterface/output
EXT_FILENAME_SEARCH_PATTERN	File wildcard	PG??????????.TP_
REMOTE_APPLICATION	Data description	PG
PROCESSED_SUFFIX	File post-process suffix	TPP

3.6.7.4 Post&Go Preprocessor Command

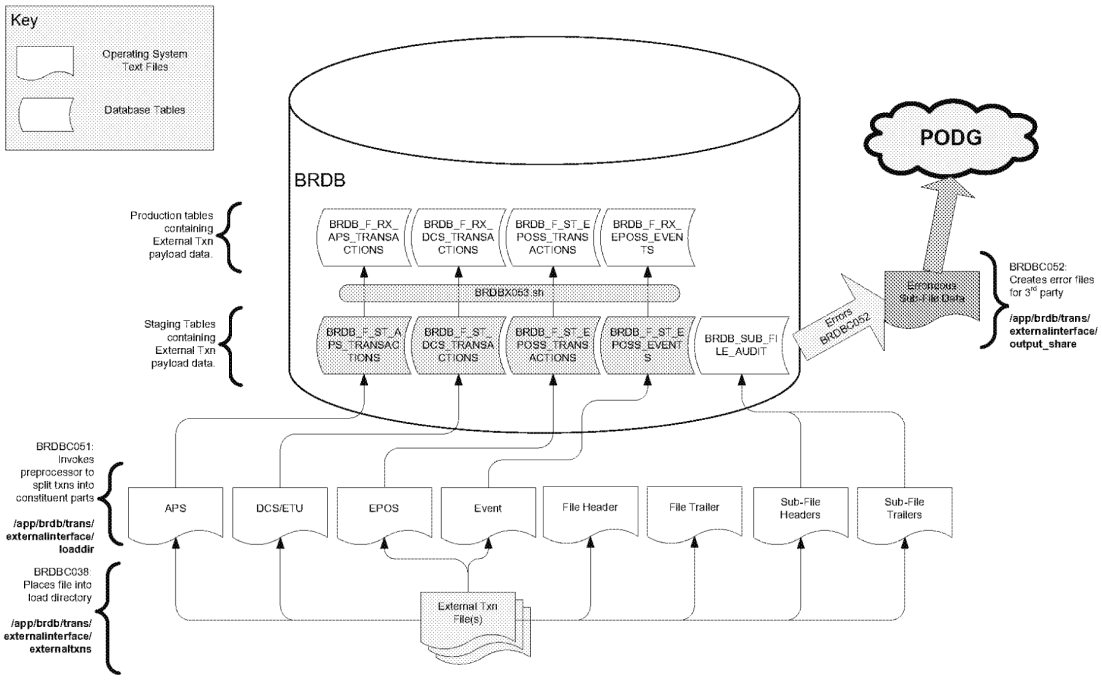
awk -f $BRDB_SH/PG.awk -v OUTDIR=#OUTDIR# #INPUTDIR#/#FILENAME#

3.6.7.5 CFD BRDBC038/File Daemon

FUJ00234958



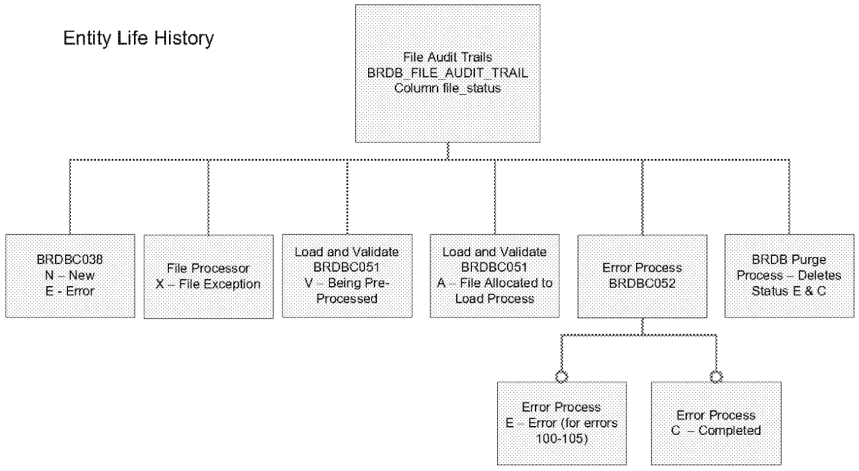
3.6.7.6 CFD Validation & Staging, Error Processing, Posting



3.6.7.7 CFD BRDB_FILE_AUDIT_TRAIL Entity Life History

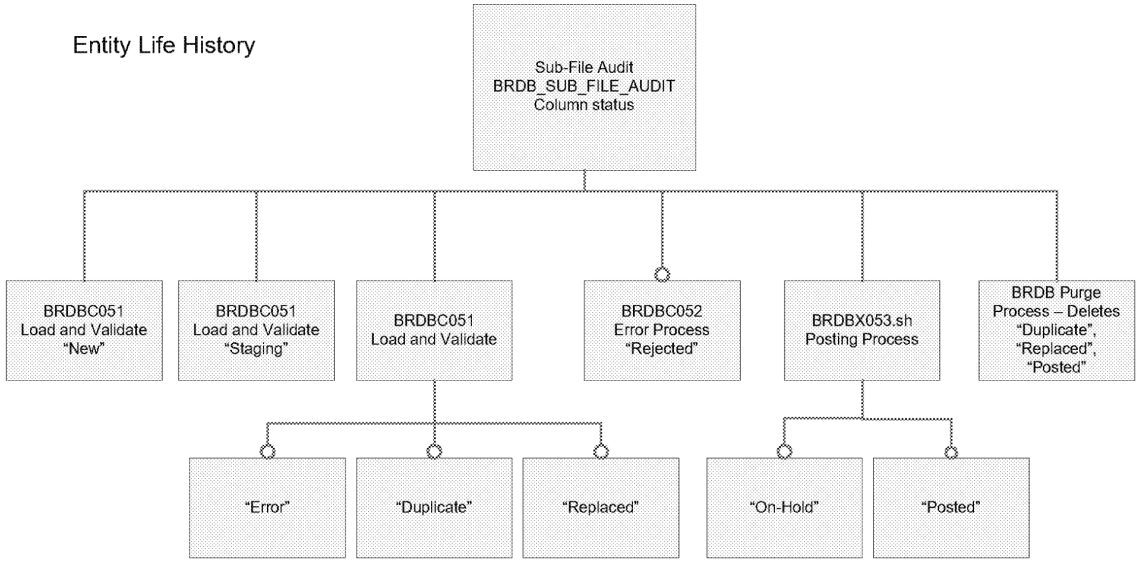
Status changes for BRDB_FILE_AUDIT_TRAIL.FILE_STATUS

FUJ00234958



3.6.7.8 CFD BRDB_SUB_FILE_AUDIT Entity Life History

Status changes for BRDB_SUB_FILE_AUDIT.STATUS



3.6.8 Collect & Return [CP0911, CP1472]

Files (containing collect & return transactions - PS *.CR_) produced by Paystation terminals are

- Placed in BRDB_INPUT_DIR_NAME (see 'External Feed Metadata' below) by PODG
- Registered via BRDBC038 (all relevant files are registered first prior to being validated)

FUJ00234958

- Validated, staged & loaded into BRDB via BRDBC058, populating Track&Trace and Items on hand tables
- Error files (PS*.CRX) are place in the output directory for PODG

3.6.8.1 Paystation External Feed Metadata

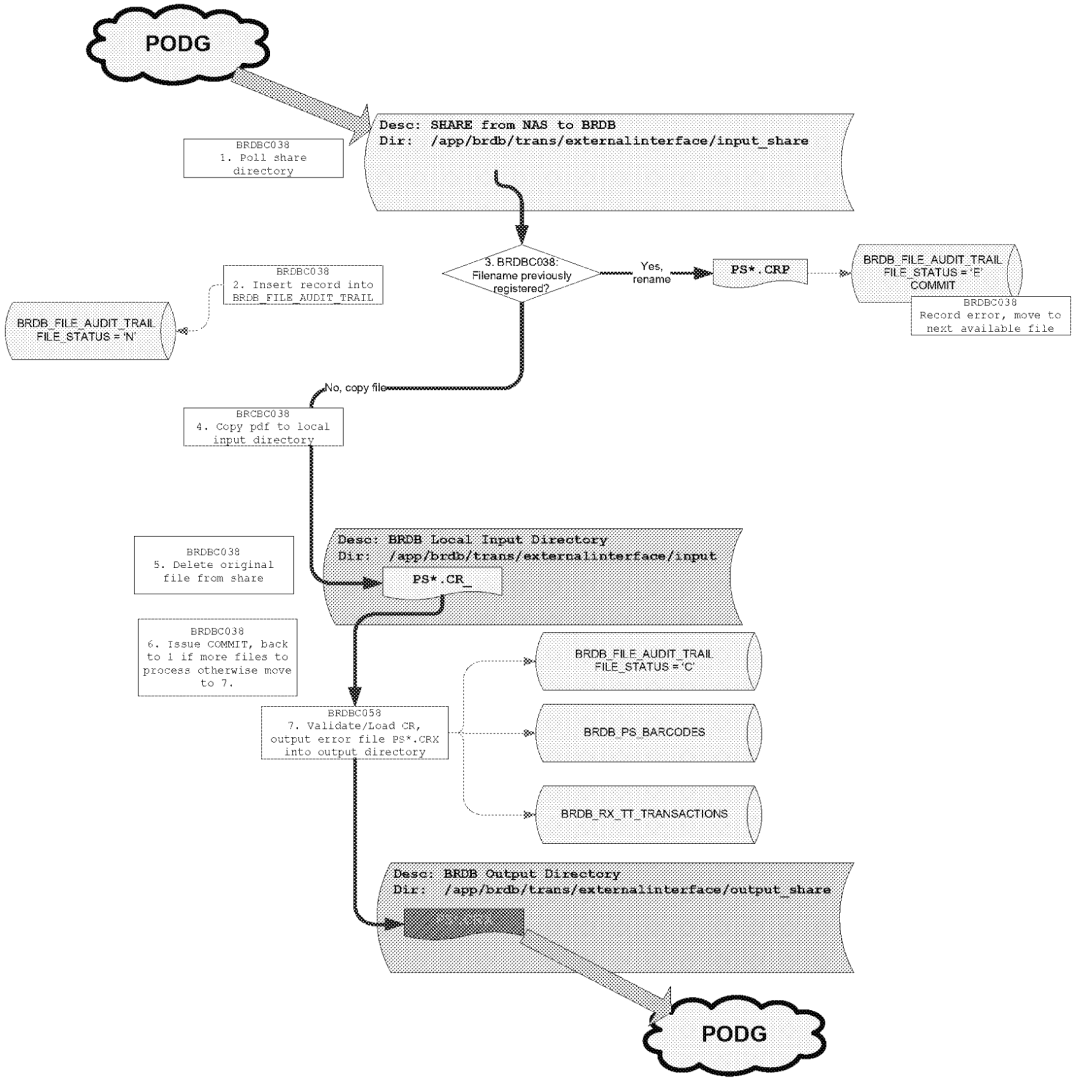
COLUMN NAME	DESCRIPTION	PAYSTATION VALUE
INPUTSHARE_DIR_NAME	PODG drop location	/app/brdb/trans/externalinterface/input_share
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/externaltxns
AUDIT_DIR_NAME	BRDB audit directory	NULL
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
OUTPUTSHARE_DIR_NAME	PODG pickup location	/app/brdb/trans/externalinterface/output_share
BRDB_OUTPUT_DIR_NAME	BRDB local output	/app/brdb/trans/externalinterface/output
EXT_FILENAME_SEARCH_PATTERN	File wildcard	PS??????????.CR_
REMOTE_APPLICATION	Data description	PS
PROCESSED_SUFFIX	File post-process suffix	CRP

3.6.8.2 Paystation Preprocessor Command

awk -f /app_sw/brdb/sh/CR.awk -v OUTDIR=#OUTDIR# #INPUTDIR#/#FILENAME#

3.6.8.3 C&R BRDBC038/BRDBC058

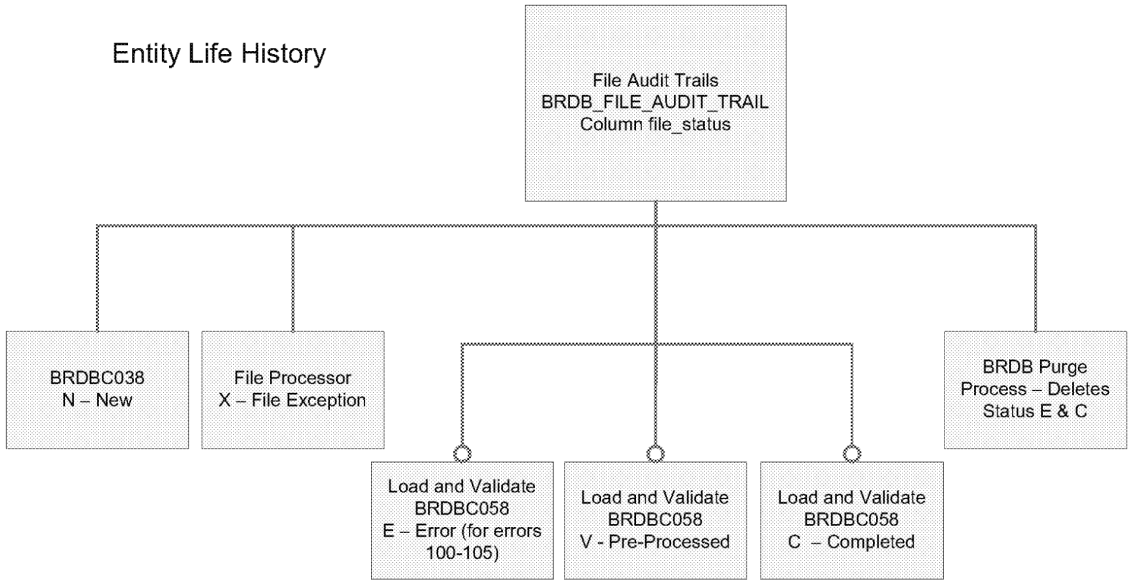
FUJ00234958



3.6.8.4 CFD BRDB_FILE_AUDIT_TRAIL Entity Life History

FUJ00234958

Status changes for BRDB_FILE_AUDIT_TRAIL.FILE_STATUS



3.6.9 ENHANCED USER MANAGEMENT [CP1913]

XML file based on POID and related training data from ForgeRock are:

- Placed in BRDB_INPUT_DIR_NAME (see 'External Feed Metadata' below) by PODG
- registered via BRDBC038 File Input Daemon
- loading via BRDBC066 ForgeRock Loader Daemon

3.6.9.1 External Feed Metadata

COLUMN NAME	DESCRIPTION	VALUE
Note: This metadata is stored in BRDB_EXT_INTERFACE_FEEDS, identified by the row "WHERE ext_interface_feed_name = 'BRDB_EUM_FORGEROCK_LOADER '".		
INPUTSHARE_DIR_NAME	PODG drop location	/app/brdb/trans/externalinterface/input_share
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/input
AUDIT_DIR_NAME	BRDB audit directory (used for error files)	/app/brdb/trans/audit/externalinterfaceaudit/ externaltxns
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
EXT_FILENAME_SEARCH_PATTERN	EUM file wildcard	FR*.XM_
REMOTE_APPLICATION	Data description	FROCK
PROCESSED_SUFFIX	File post-process suffix indicator	EUM

FUJ00234958

3.6.9.2 BRDBC066 Loader Deamon

ForgeRock files are picked up and register via BRDBC038 File Input Deamon process; BRDBC066 Loader Deamon performs the followings:

- Loops through files in BRDB_FILE_AUDIT_TRAIL (where process_name = 'BRDB_EUM_FORGEROCK_LOADER' and file_status = 'N')
- Load each file into BRDB tables OPS$BRDB. BRDB_POID_USER_DETAILS and BRDB_POID_CURRICULA
- sets the column FILE_STATUS in BRDB_FILE_AUDIT_TRAIL to 'C' (complete) or 'E'(error)
- delete file in BRDB_INPUT_DIR_NAME(/app/brdb/trans/externalinterface/input)
- rename file extension in BRDB_LOAD_DIR_NAME(/app/brdb/trans/externalinterface/loaddir) to 'EUM'
- exceptions are logged in OPS$BRDB.BRDB_OPERATIONAL_EXCEPTIONS
- File and record errors are logged in OPS$BRDB.BRDB_FILE_ERRORS
- Errors and warning nessages are written to the error file located in AUDIT_DIR_NAME(/app/brdb/trans/audit/externalinterfaceaudit/ externaltxns) with the file extension of 'ERR'

3.6.10 ATM Daily Withdrawals File [CP2076]

The module will load the ATM withdrawal file (AT???????????.CSV) provided by Bank of Ireland.

- ATM withdrawal transactions files placed in BRDB_INPUT_DIR_NAME by PODG
- registered via BRDBC038 (i.e. BRDBC038 AT <YYYYMMDD>)
- validated & staged via BRDBC051 (i.e. BRDBC051 <YYYYMMDD>)
- Error returned to the originator via BRDBC052

3.6.10.1 ATM File Metadata

COLUMN NAME	DESCRIPTION	VALUE
Note: This metadata is stored in BRDB_EXT_INTERFACE_FEEDS, identified by the row "WHERE ext_interface_feed_name = 'AT'		
INPUTSHARE_DIR_NAME	file source share	/app/brdb/trans/externalinterface/input_share
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/externaltxns
AUDIT_DIR_NAME	BRDB audit directory	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
EXT_FILENAME_SEARCH_PATTERN	File wildcard	AT???????????.CSV
COMMAND_TO_RUN	Command that BRDBC038 runs	BRDB_EXT_AT
EXECUTE_PER_FILE	Child process exec per file?	Y
REMOTE_APPLICATION	Data description	AT
PROCESSED_SUFFIX	File post-process suffix indicator	TPP
EXT_INTERFACE_FEED_NAME	FEED Name	AT
SLEEP_REPEAT_YN		N
SLEEP_REPEAT_SECS		0

FUJ00234958

COLUMN NAME	DESCRIPTION	VALUE
OUTPUTSHARE_DIR_NAME	BRDB outputshare directory	/app/brdb/trans/externalinterface/output_share
BRDB_OUTPUT_DIR_NAME	BRDB output directory	/app/brdb/trans/externalinterface/output

3.6.10.2 ATM Preprocessor Command

awk -f FAT.awk -F',' -v OUTDIR=<OUTPUT DIRECTORY> <$BRDB_SH/ATM_FILTER.TXT> <ATM FILENAME>

awk -f AT.awk -F',' -v OUTDIR=<OUTPUT DIRECTORY> <ATM FILENAME>

3.6.11 BRDB_TA_LOAD[CP2347]

Transaction Acknowledgements validate and load the are:

- Placed in /app/brdb/trans/externalinterface/externaltxns/input (see 'External Feed Metadata' below) by PODG
- registered via BRDBC038 File Input Daemon

3.6.11.1 External Feed Metadata

COLUMN NAME	DESCRIPTION	VALUE
Note: This metadata is stored in BRDB_EXT_INTERFACE_FEEDS, identified by the row "WHERE ext_interface_feed_name = 'TA ".		
INPUTSHARE_DIR_NAME	PODG drop location	/app/brdb/trans/externalinterface/input_share
BRDB_INPUT_DIR_NAME	BRDB input directory	/app/brdb/trans/externalinterface/externaltxns
AUDIT_DIR_NAME	BRDB audit directory (used for error files)	NULL
BRDB_LOAD_DIR_NAME	BRDB load directory	/app/brdb/trans/externalinterface/loaddir
EXT_FILENAME_SEARCH_PATTERN	EUM file wildcard	??????????????????.tan
REMOTE_APPLICATION	Data description	Credence
PROCESSED_SUFFIX	File post-process suffix indicator	TAN

3.6.11.2 BRDBC066 Loader Deamon

ForgeRock files are picked up and register via BRDBC038 File Input Deamon process; BRDBC066 Loader Deamon performs the followings:

- Loops through files in BRDB_FILE_AUDIT_TRAIL (where process_name = 'BRDB_EUM_FORGEROCK_LOADER' and file_status = 'N')
- Load each file into BRDB tables OPS$BRDB. BRDB_POID_USER_DETAILS and BRDB_POID_CURRICULA
- sets the column FILE_STATUS in BRDB_FILE_AUDIT_TRAIL to 'C' (complete) or 'E'(error)
- delete file in BRDB_INPUT_DIR_NAME(/app/brdb/trans/externalinterface/input)
- rename file extension in BRDB_LOAD_DIR_NAME(/app/brdb/trans/externalinterface/loaddir) to 'EUM'
- exceptions are logged in OPS$BRDB.BRDB_OPERATIONAL_EXCEPTIONS
- File and record errors are logged in OPS$BRDB.BRDB_FILE_ERRORS

FUJ00234958

- Errors and warning nessages are written to the error file located in AUDIT_DIR_NAME(/app/brdb/trans/audit/externalinterfaceaudit/ externaltxns) with the file extension of 'ERR'

3.7 BRDB Schedules and Failover

The Scheduling tool used for running BRDB (and other HNG-X schedules) is TWS. TWS needs to undergo a number of steps in a failover scenario. These are detailed in the relevant TWS and scheduling documentation. However, it is still the case that TWS (as with other applications) requires the DNS reconfigured before post-failover testing can begin. To clarify, failover refers only to the database failover from the primary database cluster (IRRELEVANT) to the standby database cluster IRRELEVANT IRRELEVANT and *not* a full campus failover, e.g. IRE11 to IRE19.

See Steps [7.] and [8.] of Section 6.1 for more on allowing applications seamless access to BRDB on database primary-to-standby cluster post-failover.

3.8 Schedule BRDB_PAUSE_FEED3

This schedule is run daily. It stops the two NPS copy processes prior to the starting of the daily BRDB schedule. It consists of two tasks which can be run on any active node; see section 3.2 above for details. Only the two parent jobs are included here, which are:

BRDBX011_PAUSE_NPS_TT_COPY

BRDBX011_PAUSE_NPS_GREV_COPY

3.8.1 BRDBX011_PAUSE_APS_NRT_PROCESSDependencies

Schedule BRDB_PAUSE_FEED3 depends on the completion of schedule BRDB_BKP_COMPL.

3.8.2 Job BRDBX011_PAUSE_NPS_TT_COPY

This job stops the copying of Track and Trace transactions to NPS, by setting a system parameter (see section 3.5).

3.8.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TT_TXN_TO_NPS_STOP_YN and value "Y" (i.e. System parameter in BRDB_SYSTEM_PARAMETER.parameter_text named 'BRDB_TT_TXN_TO_NPS_STOP_YN' is set to 'Y').

3.8.2.2 Rerun Action

Rerun the job once the underlying problem has been resolved, unless the the node on which it was running is now down; rerun one of the cancelled jobs from one of the other instances instead.

3.8.3 Job BRDBX011_PAUSE_NPS_GREV_COPY

This job stops the copying of Reversals transactions to NPS, by setting a system parameter (see section 3.5).

3.8.3.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_REV_TXN_TO_NPS_STOP_YN and value "Y" (i.e. System parameter in BRDB_SYSTEM_PARAMETER.parameter_text named 'BRDB_REV_TXN_TO_NPS_STOP_YN' is set to 'Y').

3.8.3.2 Rerun Action

FUJ00234958

Rerun the job once the underlying problem has been resolved, unless the the node on which it was running is now down; rerun one of the cancelled jobs from one of the other instances instead.

3.8.4 Job BRDBX011_PAUSE_APS_NRT_PROCESS

This job stops the copying APS transactions, by setting a system parameter (see section 3.5).

3.8.4.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_APS_NRT_PROCESS_STOP_YN and value “Y” (i.e. System parameter in BRDB_SYSTEM_PARAMETER.parameter_text named BRDB_APS_NRT_PROCESS_STOP_YN is set to 'Y').

3.8.4.2 Rerun Action

Rerun the job once the underlying problem has been resolved, unless the the node on which it was running is now down; rerun one of the cancelled jobs from one of the other instances instead.

3.9 Schedule BRDB_STARTUP

This schedule is run daily. It runs the BRDB start of day utility. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDBC001 is included here.

Additional monitoring is required so that an alert is raised if this job has not completed by 06:00. This is implemented within the BRDB_MONITOR schedule – see section 3.77.

3.9.1 Dependencies

Schedule BRDB_STARTUP depends on the completion of schedule BRDB_PAUSE_FEED3.

3.9.2 Job BRDBC001

This job runs the BRDB start of day utility in order to create "n" partitions ahead; n partition/ days (which pre Release 9 is one day) will be 7 days in advance and this can be configurable via ‘PARTITIONS_AHEAD’ BRDB System Parameters. BRDB's system date is incremented by one.

3.9.2.1 Implementation

This job is implemented by a call to the executable BRDBC001.

3.9.2.2 Rerun Action

Check the partition metadata is as expected (refer to 5.3.3.1), if the metadata appears OK then fix the underlying problem (that caused the abend), raise a high priority call with 4th line support and then rerun the job.

Only rerun the failed instance of the job if the current time is before the time threshold specified by the system parameter ‘PARTITIONS_EXPIRED_TIME’. If the current time is beyond that value, invoke BRDBC001 with no input parameters, i.e no DATE parameter. Invoking the program in this mode will create only one set of partitions, regardless of the value defined in system parameter ‘PARTITIONS AHEAD’.If the rerun fails then do not attempt to rerun a 3rd time, liase with 4th line support - the resolution should be reached before 6 p.m. that day.

3.10 Schedule BRDB_START_FEED3

This schedule is run daily. It prepares for the running of the two NPS copy processes by reversing the changes that stopped them earlier in the schedule. It consists of two tasks which can be run on any active node; see section 3.2 above for details. Only the two parent jobs are included here, which are:

FUJ00234958

BRDBX011_START_NPS_TT_COPY

BRDBX011_START_NPS_GREV_COPY

3.10.1 BRDBX011_START_APS_NRT_PROCESSDependencies

Schedule BRDB_START_FEED3 depends on the completion of schedule BRDB_STARTUP.

3.10.2 Job BRDBX011_START_NPS_TT_COPY

This job prepares for the starting of the copying of Track and Trace transactions to NPS, by setting a system parameter (see section 3.5).

3.10.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TT_TXN_TO_NPS_STOP_YN and value “N”.

3.10.2.2 Rerun Action

Alert Operations on failure.

3.10.3 Job BRDBX011_START_NPS_GREV_COPY

This job prepares for the starting of the copying of Reversals transactions to NPS, by setting a system parameter (see section 3.5).

3.10.3.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_REV_TXN_TO_NPS_STOP_YN and value “N”.

3.10.3.2 Rerun Action

Alert Operations on failure.

3.10.4 Job BRDBX011_START_APS_NRT_PROCESS

This job starts copying APS transactions, by setting a system parameter (see section 3.5).

3.10.4.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_APS_NRT_PROCESS_STOP_YN and value “N” (i.e. System parameter in BRDB_SYSTEM_PARAMETER.parameter_text named BRDB_APS_NRT_PROCESS_STOP_YN is set to 'N').

3.10.4.2 Rerun Action

Alert Operations on failure.

3.11 Schedule BRDB_TT_TO_NPS3

This schedule is run daily to start the Track and Trace NPS data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_TT_TO_NPS_1…4_NOPAGE.

3.11.1 Dependencies

Schedule BRDB_TT_TO_NPS3 depends on the completion of schedule BRDB_START_FEED3.

3.11.2 Job BRDBX003_TT_TO_NPS_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Track and Trace transactions to NPS.

FUJ00234958

3.11.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_TT_TXN_TO_NPS.

3.11.2.2 Database Link Information

NBX_TT_HARVESTER_AGENT_1@NPS1

3.11.2.3 Rerun Action

Rerun on failure. See 3.5.1

3.12 Schedule BRDB_GREV_NPS3

This schedule is run daily to start the Reversals NPS data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_GREV_TO_NPS_1…4_NOPAGE.

3.12.1 Dependencies

Schedule BRDB_GREV_NPS3 depends on the completion of schedule BRDB_START_FEED3.

3.12.2 Job BRDBX003_GREV_TO_NPS_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Reversals transactions to NPS.

3.12.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_REV_TXN_TO_NPS.

3.12.2.2 Database Link Information

NBX_GREV_AGENT_1@NPS2

3.12.2.3 Rerun Action

Rerun on failure. See 3.5.1

3.13 Schedule BRDB_PAUSE_FEED1

This schedule is run daily at 07:50. It stops the two NPS copy processes and one (CR) file deamon process prior to the start of day processing. It consists of three tasks which can be run on any active node; see section 3.2 above for NPS copy processes details and section 3.98 below for (CR) file deamon. Only the three parent jobs are included here, which are:

BRDBX011_PAUSE_NPS_TT_COPY

BRDBX011_PAUSE_NPS_GREV_COPY

BRDBX011_STOP_CR

BRDBX011_PAUSE_APS_NRT_PROCESS

Additional monitoring is required so that an alert is raised if this job has not completed by 08:00. This is implemented within the BRDB_MONITOR schedule – see section 3.77.

3.13.1 Dependencies

Schedule BRDB_PAUSE_FEED1 depends on the completion of schedules BRDB_STARTUP and BRDB_START_FEED3.

3.13.2 Job BRDBX011_PAUSE_NPS_TT_COPY

FUJ00234958

This job stops the copying of Track and Trace transactions to NPS, by setting a system parameter (see section 3.5).

3.13.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TT_TXN_TO_NPS_STOP_YN and value “Y”.

3.13.2.2 Rerun Action

Alert Operations on failure.

3.13.3 Job BRDBX011_PAUSE_NPS_GREV_COPY

This job stops the copying of Reversals transactions to NPS, by setting a system parameter (see section 3.5).

3.13.3.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_REV_TXN_TO_NPS_STOP_YN and value “Y”.

3.13.3.2 Rerun Action

Alert Operations on failure.

3.13.4 Job BRDBX011_STOP_CR

This job stops the (CR) file deamon , by setting a system parameter (see section 3.98).

3.13.4.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name CR_STOP_YN and value “Y”.

3.13.4.2 Rerun Action

Alert Operations on failure.

3.13.5 Job BRDBX011_PAUSE_APS_NRT_PROCESS

This job stops the copying APS transactions, by setting a system parameter (see section 3.5).

3.13.5.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_APS_NRT_PROCESS_STOP_YN and value “Y” (i.e. System parameter in BRDB_SYSTEM_PARAMETER.parameter_text named BRDB_APS_NRT_PROCESS_STOP_YN is set to 'Y').

3.13.5.2 Rerun Action

Rerun the job once the underlying problem has been resolved, unless the the node on which it was running is now down; rerun one of the cancelled jobs from one of the other instances instead.

3.14 Schedule BRDB_COMPLETE

This schedule is run daily. It checks that the BRDB schedule has completed and creates a flag file via the job CREATE_BRDB_COMPLETE_FLAG.

3.14.1 Dependencies

FUJ00234958

Schedule BRDB_COMPLETE depends on the completion of schedules BRDB_BKP_COMPL, BRDB_STARTUP and BRDB_PAUSE_FEED1.

3.14.2 Job CREATE_BRDB_COMPLETE_FLAG

This job creates the flag file /opt/tws/FLAGS/BRDB_COMPLETE_FLAG.

3.14.2.1 Implementation

This job is implemented by a call to the "touch" command with the relevant file name.

3.14.2.2 Rerun Action

*** Prompts for rerun – action? **

3.15 Schedule BRDB_SOD

This schedule is run daily at 08:00. It checks that the BRDB has completed start of day processing.

3.15.1 Dependencies

Schedule BRDB_COMPLETE depends on the existence of the flag files /opt/tws/FLAGS/BRDB_COMPLETE.flag and /opt/tws/FLAGS/BRDB_BKUP_COMPLETE.flag.

3.15.2 Job DELETE_BRDB_COMPLETE_FLAG

This job deletes the flag file /opt/tws/FLAGS/BRDB_complete.FLAG.

3.15.2.1 Implementation

This job is implemented by a call to the "rm" command with the relevant file name.

3.15.2.2 Rerun Action

Alert Operations on failure?

3.15.3 Job DELETE_BRDB_COMPLETE_FLAG

This job deletes the flag file /opt/tws/FLAGS/BRDB_BKUP_complete.FLAG.

3.15.3.1 Implementation

This job is implemented by a call to the "rm" command with the relevant file name.

3.15.3.2 Rerun Action

Alert Operations on failure?

3.16 Schedule BRDB_START_FEED1

This schedule is run daily at 08:02. It prepares for the running of the two NPS copy processes by reversing the changes that stopped them earlier in the schedule. It consists of two tasks which can be run on any active node; see section 3.2 above for details. Only the two parent jobs are included here, which are:

BRDBX011_START_NPS_TT_COPY

BRDBX011_START_NPS_GREV_COPY

BRDBX011_START_APS_NRT_PROCESS

3.16.1 Dependencies

Schedule BRDB_START_FEED1 depends on the completion of schedule BRDB_SOD.

FUJ00234958

3.16.2 Job BRDBX011_START_NPS_TT_COPY

This job prepares for the starting of the copying of Track and Trace transactions to NPS, by setting a system parameter (see section 3.5).

3.16.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TT_TXN_TO_NPS_STOP_YN and value “N”.

3.16.2.2 Rerun Action

Alert Operations on failure.

3.16.3 Job BRDBX011_START_NPS_GREV_COPY

This job prepares for the starting of the copying of Reversals transactions to NPS, by setting a system parameter (see section 3.5).

3.16.3.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_REV_TXN_TO_NPS_STOP_YN and value “N”.

3.16.3.2 Rerun Action

Alert Operations on failure.

3.16.4 Job BRDBX011_START_APS_NRT_PROCESS

This job starts copying APS transactions, by setting a system parameter (see section 3.5).

3.16.4.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_APS_NRT_PROCESS_STOP_YN and value “N” (i.e. System parameter in BRDB_SYSTEM_PARAMETER.parameter_text named BRDB_APS_NRT_PROCESS_STOP_YN is set to 'N').

3.16.4.2 Rerun Action

Alert Operations on failure.

3.17 Schedule BRDB_START_APOP

This schedule is run daily at 08:02. It prepares for the running of the APOP copy process by reversing the changes that stop them from running. It consists of a single two task which can be run on any active node. Only the parent job is included here, which is:

BRDBX011_START_APOP_TC_COPY

3.17.1 Dependencies

Schedule BRDB_START_APOP depends on the completion of schedule BRDB_SOD.

3.17.2 Job BRDBX011_START_APOP_TC_COPY

This job prepares for the starting of copying Transaction Confirmations to APOP, by setting a system parameter (see section 3.5).

3.17.2.1 Implementation

FUJ00234958

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TXN_CONF_TO_APOP_STOP_YN and value "N".

3.17.2.2 Rerun Action

Alert Operations on failure.

3.18 Schedule BRDB_TT_TO_NPS1

This schedule is run daily at 08:05 to restart the Track and Trace NPS data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_TT_TO_NPS_1…4_NOPAGE.

3.18.1 Dependencies

Schedule BRDB_TT_TO_NPS1 depends on the completion of schedule BRDB_START_FEED1.

3.18.2 Job BRDBX003_TT_TO_NPS_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Track and Trace transactions to NPS.

3.18.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_TT_TXN_TO_NPS.

3.18.2.2 Database Link Information

NBX_TT_HARVESTER_AGENT_1@NPS1

3.18.2.3 Rerun Action

Rerun on failure. See 3.5.1

3.19 Schedule BRDB_GREV_NPS1

This schedule is run daily at 08:05 to restart the Reversals NPS data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_GREV_TO_NPS_1…4_NOPAGE.

3.19.1 Dependencies

Schedule BRDB_GREV_NPS1 depends on the completion of schedule BRDB_START_FEED1.

3.19.2 Job BRDBX003_GREV_TO_NPS_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Reversals transactions to NPS.

3.19.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_REV_TXN_TO_NPS.

3.19.2.2 Database Link Information

NBX_GREV_AGENT_1@NPS2

3.19.2.3 Rerun Action

Rerun on failure. See 3.5.1

3.20 Schedule BRDB_TC_TO_APOP

This schedule is run daily at 08:05 to start the Transaction Confirmation to APOP data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_TC_TO_APOP_1…4_NOPAGE.

FUJ00234958

3.20.1 Dependencies

Schedule BRDB_TC_TO_APOP depends on the completion of schedule BRDB_START_APOP.

3.20.2 Job BRDBX003_TC_TO_APOP_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Transaction Confirmations to APOP.

3.20.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_TXN_CONF_TO_APOP.

3.20.2.2 Database Link Information

APOPBRDB@APOP

3.20.2.3 Rerun Action

Rerun on failure. See 3.5.1

3.21 Schedule BRDB_START_MON

This schedule is run daily at 08:10 to set the Daemon Monitoring Process (BRDBC041) STOP_YN flag to 'N'. It consists of a single task which is run on one node by a job named BRDBX011_START_DAEMON_MON.

3.21.1 Dependencies

Schedule BRDB_START_MON depends on the completion of schedule BRDB_SOD.

3.21.2 Job BRDBX011_START_DAEMON_MON

This job (one node) sets the BRDB_DAEMON_MONITOR_STOP_YN flag to 'N'.

3.21.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant feed name BRDB_DAEMON_MONITOR_STOP_YN.

3.21.2.2 Database Link Information

N/A

3.21.2.3 Rerun Action

Rerun on failure.

3.22 Schedule BRDB_FEED_MON

This schedule is run daily to start the multi-node Daemon Monitoring processes (BRDBC041). It consists of a single task which is run on each active node by jobs named BRDBC041_BRDB_DAEMON_MONITOR_1…4.

3.22.1 Dependencies

Schedule BRDB_FEED_MON depends on the completion of schedules:

- BRDB_START_FEED1
- BRDB_START_APOP

FUJ00234958

- BRDB_START_MON

3.22.2 Job BRDBC041_BRDB_DAEMON_MONITOR_1…4

These jobs (one per node) start the daemon monitoring process than acts as a watchdog for the other daemon jobs (e.g. Track&Trace, GREV, etc).

3.22.2.1 Implementation

These jobs are implemented by a call to the pro*c executable BRDBC041 specifying the TWS date and instance ID.

3.22.2.2 Database Link Information

N/A

3.22.2.3 Rerun Action

If this job fails then it may suggest a monitored feed (e.g. GREV) has timed out - indicating a problem elsewhere in BRDB.

Once the root cause of the failure is resolved then restart the monitored feed (in the above example then, GREV) and then rerun this job on the node that it failed on.

3.23 Schedule BRDB_PAUSE_MON

This schedule is run daily at 20:00 to set the Daemon Monitoring Process (BRDBC041) STOP_YN flag to 'Y'. It consists of a single task which is run on one node by a job named BRDBX011_PAUSE_DAEMON_MON.

3.23.1 Dependencies

Schedule BRDB_PAUSE_MON depends on the completion of schedule BRDB_SOB & BRDB_START_MON.

3.23.2 Job BRDBX011_PAUSE_DAEMON_MON

This job (one node) sets the BRDB_DAEMON_MONITOR_STOP_YN flag to 'Y'.

3.23.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant feed name BRDB_DAEMON_MONITOR_STOP_YN.

3.23.2.2 Database Link Information

N/A

3.23.2.3 Rerun Action

Rerun on failure.

3.24 Schedule BRDB_SOB

This schedule is run daily at 19:00. It marks the start of the evening BRDB schedule.

3.24.1 Dependencies

None.

FUJ00234958

3.24.2 Job COMPLETE

This job simply echoes a message before exiting.

3.24.2.1 Implementation

This job is implemented by a call to the echo command.

3.24.2.2 Rerun Action

None.

3.25 Schedule BRDB_ONCH_AGG

This schedule is run daily. It aggregates the overnight cash on hand (ONCH) figures as well as setting the last good ONCH date for relevant rows in column OPS$BRDB.BRDB_BRANCH_STOCK_UNITS.LAST_GOOD_ONCH_DATE. It performs two tasks, firstly running the aggregation itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX007_ONCH_AGG_1…4 are included here. The second task checks for completion of the previous task, and can be run on any active node; see section 3.2 above for details. Only the parent job BRDBC008_CHECK_ONCH_AGG is included here.

3.25.1 Dependencies

Schedule BRDB_ONCH_AGG depends on the completion of schedule BRDB_SOB.

Job BRDBC008_CHECK_ONCH_AGG depends on jobs BRDBX007_ONCH_AGG_1…4.

3.25.2 Job BRDBX007_ONCH_AGG_1…4

These jobs (one per node) perform the aggregation of the overnight cash on hand (ONCH) figures.

3.25.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX007.sh specifying the relevant aggregation name OVERNIGHT_CASH_ON_HAND.

3.25.2.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.25.3 Job BRDBC008_CHECK_ONCH_AGG

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.25.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant aggregation name OVERNIGHT_CASH_ON_HAND.

3.25.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.26 Schedule BRDB_CLR_BRANCH

This schedule runs after BRDB_FROM_EMDB2 completes and will not initiate if time threshold 01:05 has been breached. The called job archives and then deletes transactions for all closed branches. This schedule is run on 1 instance at any one time.

3.26.1 Dependencies

FUJ00234958

Schedule BRDB_CLR_BRANCH depends on the completion of schedule BRDB_FROM_EMDB2. This job is stopped at 01:05 irrespective of whether it has completed already (outstanding transactions will be rolled back and picked up the following night).

3.26.2 Job BRDBX037_CLEAR_BRDATA

This job runs the BRDB automated closure process (BRDBX037.sh). Transactions are committed by FAD_HASH (not individually by branch).

3.26.2.1 Implementation

This job is implemented by a call to the shell script BRDBX037.sh, along with the TWS business date and instance number.

The process identifies all branches to be cleared by the following query

```
SELECT fad_hash, branch_accounting_code
FROM   brdb_branch_info
WHERE  branch_status = 'Closed'
AND    suspend_distribution = 'Y'
```

All transactions for those closed branches in a number of tables (identified in column BRDB_CLEARED_CONTROL_DATA.source_table) are loaded into archive tables (identified in column BRDB_CLEARED_CONTROL_DATA.target_table) and then deleted from the original tables.

Note that these transactions are not replicated to BRSS, BRSS has an equivalent process (BRSSX037.sh) that carries the closures independently of BRDB.

Closed, cleared and archived branches are recorded in table BRDB_CLEARED_CLOSURE_DATA, with column brdb_closure_date identifying when the branch was cleared on BRDB.

3.26.2.2 Exceptions

BRDBX037.sh checks each branch (to be cleared) has not traded within the last 5 days by querying BRDB_BRANCH_NODE_INFO.last_logout_timestamp.

If a branch does show activity then an exception is logged in BRDB_OPERATIONAL_EXCEPTIONS with an exception code of BRDB35110. The branch will continue to log exceptions until the last logout timestamp is older than TWS business date - 5 days.

3.26.2.3 Rerun Action

This job can be rerun if ISD's opinion is that there is enough of a window to process at least one FAD HASH before 01.05am. If there is not enough time to complete then the following night's schedule will pick up from where BRDBX037 stopped previously.

Alert Operations on failure.

3.27 Schedule BRDB_PAUSE_APOP

This schedule is run daily at 20:00. It stops the APOP feed process, to allow the APOP batch jobs to run overnight without activity occurring in the relevant tables. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job is included here, which is:

BRDBX011_PAUSE_APOP_TC_COPY

3.27.1 Dependencies

FUJ00234958

Schedule BRDB_PAUSE_APOP depends on the completion of schedule BRDB_SOB, BRDB_START_APOP.

3.27.2 Job BRDBX011_PAUSE_APOP_TC_COPY

This job stops the copying of Transaction Confirmations to APOP, by setting a system parameter (see section 3.5).

3.27.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TXN_CONF_TO_APOP_STOP_YN and value "Y".

3.27.2.2 Rerun Action

Alert Operations on failure.

3.28 Schedule BRDB_NWB_TO_DRS

3.29 Following PBS Decommissioning, this schedule no longer runsSchedule BRDB_DCS_TO_DRS

Following PBS decommissioning, this schedule no longer runs..

3.30 Schedule BRDB_DRS_COMPL

Following PBS decommissioning this schedule no longer runs..

3.31 Schedule BRDB_XFR_COMPL

This schedule is run daily. It marks the end of the transfer schedule.

3.31.1 Dependencies

Schedule BRDB_XFR_COMPL depends on the completion of schedules BRDB_DRS_COMPL, BRDB_ZIP_CRED, BRDB_ZIP_BTR, BRDB_BDC_TO_FRES, BRDB_AP_RECON .

3.31.2 Job COMPLETE

This job simply echoes a message before exiting.

3.31.2.1 Implementation

This job is implemented by a call to the echo command.

3.31.2.2 Rerun Action

None.

3.32 Schedule BRDB_FEED_ERRORS

This schedule is run daily. It runs the process to raise operation exceptions for data feed errors. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDBX007_RAISE_FEED_DATA_EXCEPTIONS is included here.

3.32.1 Dependencies

3.32.2 Schedule BRDB_FEED_ERRORS depends on the completion of schedule BRDB_XFR_COMPL BRDB_PAFAD_LOAD

FUJ00234958

BRDB_POE_LOAD BRDB_PPK_LOAD BRDB_EUM_LOAD BRDB_PLO_LOAD, BRDB_RDC_LOAD.Job BRDBX007_RAISE_FEED_DATA_EXCEPTIONS

This job runs the process to raise operation exceptions for data feed errors.

3.32.2.1 Implementation

This job is implemented by a call to the shell script BRDBX007.sh specifying the relevant process name RAISE_FEED_DATA_EXCEPTIONS.

3.32.2.2 Rerun Action

Alert Operations on failure.

3.33 Schedule BRDB_NCU_TXN_AGG

This schedule is run daily at 1:15. It performs data aggregation for the daily summary. It performs two tasks, firstly running the aggregation itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX007_NON_CUMU_TXN_TOTALS_1…4 are included here. The second task checks for completion of the previous task, and can be run on any active node; see section 3.2 above for details. Only the parent job BRDBC008_CHECK_NON_CUMU_TXN_AGGR is included here.

3.33.1 Dependencies

Job BRDBC008_CHECK_NON_CUMU_TXN_AGGR depends on jobs BRDBX007_NON_CUMU_TXN_TOTALS_1…4 & BRDB_TXN_POST.

3.33.2 Job BRDBX007_NON_CUMU_TXN_TOTALS_1…4

These jobs (one per node) perform data aggregation for the daily summary.

3.33.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX007.sh specifying the relevant aggregation name BRDB_NON_CUMU_TXN_AGGR.

3.33.2.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.33.3 Job BRDBC008_CHECK_NON_CUMU_TXN_AGGR

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.33.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant aggregation name BRDB_NON_CUMU_TXN_AGGR.

3.33.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.34 Schedule BRDB_CU_TXN_AGG

This schedule is run daily. It performs data aggregation for the daily cumulative summary. It performs two tasks, firstly running the aggregation itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX007_CUMU_TXN_AGGR_1…4 are included here. The second task checks for completion of the previous task, and can be run on any active node;

FUJ00234958

see section 3.2 above for details. Only the parent job BRDBC008_CHECK_CUMU_TXN_AGGR is included here.

3.34.1 Dependencies

Schedule BRDB_CU_TXN_AGG depends on the completion of schedule BRDB_NCU_TXN_AGG.

Job BRDBC008_CHECK_CUMU_TXN_AGGR depends on jobs BRDBX007_CUMU_TXN_AGGR_1…4.

3.34.2 Job BRDBX007_CUMU_TXN_AGGR_1…4

These jobs (one per node) perform data aggregation for the cumulative daily summary.

3.34.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX007.sh specifying the relevant aggregation name BRDB_CUMU_TXN_AGGR.

3.34.2.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.34.3 Job BRDBC008_CHECK_CUMU_TXN_AGGR

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.34.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant aggregation name BRDB_CUMU_TXN_AGGR.

3.34.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.35 Schedule BRDB_BBNI_MAINT

This schedule is run daily. It runs the BRDB utility to reset sequence numbers. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDBX031_JSN_USN_SSN is included here.

3.35.1 Dependencies

Schedule BRDB_BBNI_MAINT depends on the completion of schedule BRDB_CU_TXN_AGG.

3.35.2 Job BRDBX031_JSN_USN_SSN

This job runs the BRDB utility that resets the sequence numbers.

3.35.2.1 Implementation

This job is implemented by a call to the shell script BRDBX031.sh.

3.35.2.2 Rerun Action

*** Prompts for rerun – action? **

3.36 Schedule BRDB_SUMMARY_DTE

This schedule is run daily. It sets the last daily summary date. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDBX011_SET_DAILY_SUMMARY_DATE is included here.

3.36.1 Dependencies

FUJ00234958

Schedule BRDB_SUMMARY_DTE depends on the completion of schedule BRDB_BBNI_MAINT.

3.36.2 Job BRDBX011_SET_DAILY_SUMMARY_DATE

This job sets the last daily summary date, a system parameter.

3.36.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_LAST_DAILY_SUMMARY_DATE and relevant date value.

3.36.2.2 Rerun Action

Alert Operations on failure.

3.37 Schedule BRDB_AGG_COMPL

This schedule is run daily. It marks the end of the aggregation schedule.

3.37.1 Dependencies

Schedule BRDB_AGG_COMPL depends on the completion of schedules BRDB_SUMMARY_DTE .

3.37.2 Job COMPLETE

This job simply echoes a message before exiting.

3.37.2.1 Implementation

This job is implemented by a call to the echo command.

3.37.2.2 Rerun Action

None.

3.38 Schedule BRDB_FROM_RDDS

This schedule is run daily at 00:10. It runs the Host Reference Data from RDDS data feed. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDBX003_REFDATA_FROM_RDDS is included here.

3.38.1 Dependencies

Schedule BRDB_FROM_RDDS depends on the completion of schedules BRDB_SOB and RDDS_COPY_SCHED.

3.38.2 Job BRDBX003_REFDATA_FROM_RDDS

This job runs the Host Reference Data from RDDS data feed.

3.38.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_HOST_REF_FROM_RDDS. The job populates the following BRDB tables via the RDDS database link:

All tables in the BRDB that are prefixed with 'RDDS_See DEV/APP/LLD/0050 for detailed information.

3.38.2.2 Database Link Information

RDDSBRDB@RDDS

3.38.2.3 Rerun Action

*** Prompts for rerun – action? **

FUJ00234958

3.39 Schedule BRDB_AUD_FEED

This schedule is run daily at 01:05. It performs journal auditing. It performs three tasks, firstly running the message journal auditing on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBC002_AUDIT_1…4 are included here. The second task checks for completion of the previous task, and can be run on any active node; see section 3.2 above for details. Only the parent job BRDBC008_CHECK_AUDIT_FEED is included here. The third task performs Transaction Correction journal auditing, and can be run on any active node; again see section 3.2 above for details. Only the parent job BRDBC033_AUDIT is included here.

Additional monitoring is required so that an alert is raised if this job has not completed by 04:00. This is implemented within the BRDB_MONITOR schedule – see section 3.77.

3.39.1 Dependencies

Schedule BRDB_AUD_FEED depends on the completion of schedule BRDB_SOB.

Job BRDBC008_CHECK_AUDIT_FEED depends on jobs BRDBC002_AUDIT_1…4.

Job BRDBC033_AUDIT depends on job BRDBC008_CHECK_AUDIT_FEED.

3.39.2 Job BRDBC002_AUDIT_1…4

These jobs (one per node) generate text files for the input day's auditable messages.

3.39.2.1 Implementation

These jobs are implemented by a call to the executable BRDBC002.

Outputs files to the following directories below.

Usage	BRDBBLV1 Environment Variable
Working directory	BRDB_AUDIT_FILE_TEMP
BRDB reports directory	BRDB_COUNTER_AUDIT_OUTPUT

3.39.2.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.39.3 Job BRDBC008_CHECK_AUDIT_FEED

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.39.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDBC002.

3.39.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.39.4 Job BRDBC033_AUDIT

This job generates text files for the input day's auditable transaction correction messages.

3.39.4.1 Implementation

This job is implemented by a call to the executable BRDBC033.

Outputs files to the following directories below.

FUJ00234958

Usage	BRDBBLV1 Environment Variable
Working directory	BRDB_TCT_FILE_TEMP
BRDB reports directory	BRDB_TCT_AUDIT_OUTPUT

3.39.4.2 Rerun Action

As specified in section 3.1.1, alert Operations if rerun fails.

3.40 Schedule BRDB_ORA_STATS

This schedule, which runs daily, gathers statistics on date range partitioned tables every Monday (excluding English bank holidays) and daily for all other tables (with stale statistics). It consists of a single task which can be run on any active node; see section 3.1.1 above for details. Only the parent job BRDBX005_SCHEMA is included here.

3.40.1 Dependencies

Schedule BRDB_ORA_STATS depends on the completion of schedules BRDB_AUD_FEED, BRDB_AGG_COMPL and BRDB_XFR_COMPL.

3.40.2 Job BRDBX005_SCHEMA

This job gathers the Oracle optimiser statistics.

3.40.2.1 Implementation

This job is implemented by a call to the shell script BRDBX005.sh. The input parameters [-i & -s] are present for backward compatibility only.

Statistics for tables as per those in table BRDB_ANALYZED_OBJECTS are normally gathered on a Monday (controlled by system parameter BRDBX005_GATHER_WEEK_DAY), those statistics are then copied into future partitions every night (until the following Monday).

Stale statistics for tables not present in BRDB_ANALYZED_OBJECTS are gathered every night.

3.40.2.1.1 Associated BRDB System Parameters

Parameter Name	Parameter Value	Description
DEBUG_LEVEL_FOR_BRDBX005	3 [from parameter_number]	Controls detail of stdlist output
BRDBX005_ADJUST_HIGH_LOW_FLAG	Y [from parameter_text]	Controls method of copy table stats
BRDBX005_GATHER_WEEK_DAY	MON [from parameter_text]	Day to gather stats on partitioned tables
BRDBX005_EXPORT_STATS	N [from parameter_text]	Controls whether stats are copied to BRDB_OBJECT_STATS_ARC

3.40.2.2 Rerun Action

The statistics gathering job is able to resume from where it last failed so it is feasible to rerun the job (if the failure was, for example, due to a full tablespace then that would need resolving first).

3.41 Schedule BRDB_ADMIN

This schedule is run daily. It performs administration of the BRDB database. It includes two tasks which can be run on any active node; see section 3.2 above for details. Only the parent jobs BRDBC004 and BRDBX006 are included here. It also includes two tasks which are run on each active node by jobs named BRDB_HKP_ORAFILES1 and BRDB_HKP_ORAFILES2.

FUJ00234958

3.41.1 Dependencies

Schedule BRDB_ADMIN depends on the completion of schedules BRDB_AUD_FEED, BRDB_AGG_COMPL and BRDB_XFR_COMPL.

3.41.2 Job BRDBC004

This job runs the Audit, Archive and Purge process. See Section 5.7 for the latest in BRDBC004 archival and purging logic.

3.41.2.1 Implementation

This job is implemented by a call to the executable BRDBC004.

3.41.2.2 Rerun Action

*** Prompts for rerun – action? **

3.41.3 Job BRDBX006

This job runs the BRDB File Housekeeping process.

3.41.3.1 Implementation

This job is implemented by a call to the shell script BRDBX006.sh.

3.41.3.2 Rerun Action

*** Prompts for rerun – action? **

3.41.4 Job BRDB_HKP_ORAFILES1

This job (run on each node) runs the Oracle File Housekeeping process for the BRDB.

3.41.4.1 Implementation

This job is implemented by a call to the shell script HouseKeepOrafiles.sh with the database name BRDB.

3.41.4.2 Rerun Action

*** Prompts for rerun – action? **

3.41.5 Job BRDB_HKP_ORAFILES2

This job (run on each node) runs the Oracle File Housekeeping process for ASM.

3.41.5.1 Implementation

This job is implemented by a call to the shell script HouseKeepOrafiles.sh with the database name "+ASM".

3.41.5.2 Rerun Action

*** Prompts for rerun – action? **

3.42 Schedule BRDB_PAUSE_FEED2

This schedule is run daily. It stops the two NPS copy processes and one (CR) file deamon process prior to end of day processing. It consists of three tasks which can be run on any active node; see section 3.2 above for NPS copy processes details and section 3.98 below for (CR) file deamon. Only the three parent jobs are included here, which are:

BRDBX011_PAUSE_NPS_TT_COPY

BRDBX011_PAUSE_NPS_GREV_COPY

FUJ00234958

BRDBX011_STOP_CR

3.42.1 Dependencies

Schedule BRDB_PAUSE_FEED2 depends on the completion of schedule BRDB_ADMIN.

3.42.2 Job BRDBX011_PAUSE_NPS_TT_COPY

This job stops the copying of Track and Trace transactions to NPS, by setting a system parameter (see section 3.5).

3.42.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TT_TXN_TO_NPS_STOP_YN and value “Y”.

3.42.2.2 Rerun Action

Alert Operations on failure.

3.42.3 Job BRDBX011_PAUSE_NPS_GREV_COPY

This job stops the copying of Reversals transactions to NPS, by setting a system parameter (see section 3.5).

3.42.3.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_REV_TXN_TO_NPS_STOP_YN and value “Y”.

3.42.3.2 Rerun Action

Alert Operations on failure.

3.42.4 Job BRDBX011_CR

This job stops the (CR) file deamon, by setting a system parameter (see section 3.98).

3.42.4.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name CR_STOP_YN and value “Y”.

3.42.4.2 Rerun Action

Alert Operations on failure

3.43 Schedule BRDB_EOD

This schedule is run daily. It runs the BRDB end of day utility. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDBC009 is included here.

Additional monitoring is required so that an alert is raised if this job has not completed by 04:00. This is implemented within the BRDB_MONITOR schedule – see section 3.77.

3.43.1 Dependencies

Schedule BRDB_EOD depends on the completion of schedule BRDB_PAUSE_FEED2.

3.43.2 Job BRDBC009

This job runs the BRDB end of day utility; resets BRDB_OPERATIONAL_INSTANCES.IS_AVAILABLE to 'Y' if the instance was previously down but is now available.

FUJ00234958

3.43.2.1 Implementation

This job is implemented by a call to the executable BRDBC009.

3.43.2.2 Rerun Action

*** Prompts for rerun – action? **

3.44 Schedule BRDB_START_FEED2

This schedule is run daily. It prepares for the running of the two NPS copy processes by reversing the changes that stopped them earlier in the schedule. It consists of two tasks which can be run on any active node; see section 3.2 above for details. Only the two parent jobs are included here, which are:

BRDBX011_START_NPS_TT_COPY

BRDBX011_START_NPS_GREV_COPY

3.44.1 Dependencies

Schedule BRDB_START_FEED2 depends on the completion of schedule BRDB_EOD.

3.44.2 Job BRDBX011_START_NPS_TT_COPY

This job prepares for the starting of the copying of Track and Trace transactions to NPS, by setting a system parameter (see section 3.5).

3.44.2.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_TT_TXN_TO_NPS_STOP_YN and value “N”.

3.44.2.2 Rerun Action

Alert Operations on failure.

3.44.3 Job BRDBX011_START_NPS_GREV_COPY

This job prepares for the starting of the copying of Reversals transactions to NPS, by setting a system parameter (see section 3.5).

3.44.3.1 Implementation

This job is implemented by a call to the shell script BRDBX011.sh specifying the relevant system parameter name BRDB_REV_TXN_TO_NPS_STOP_YN and value “N”.

3.44.3.2 Rerun Action

Alert Operations on failure.

3.45 Schedule BRDB_TT_TO_NPS2

This schedule is run daily to restart the Track and Trace NPS data feed after end of day processing. It consists of a single task which is run on each active node by jobs named BRDBX003_TT_TO_NPS_1…4_NOPAGE.

3.45.1 Dependencies

Schedule BRDB_TT_TO_NPS2 depends on the completion of schedule BRDB_START_FEED2.

3.45.2 Job BRDBX003_TT_TO_NPS_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Track and Trace transactions to NPS.

3.45.2.1 Implementation

FUJ00234958

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_TT_TXN_TO_NPS.

3.45.2.2 Database Link Information

NBX_TT_HARVESTER_1@NPS2

3.45.2.3 Rerun Action

Rerun on failure.

3.46 Schedule BRDB_GREV_NPS2

This schedule is run daily to restart the Reversals NPS data feed after end of day processing. It consists of a single task which is run on each active node by jobs named BRDBX003_GREV_TO_NPS_1…4_NOPAGE.

3.46.1 Dependencies

Schedule BRDB_GREV_NPS2 depends on the completion of schedule BRDB_START_FEED2.

3.46.2 Job BRDBX003_GREV_TO_NPS_1…4_NOPAGE

These jobs (one per node) start the feed that copies the Reversals transactions to NPS.

3.46.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_REV_TXN_TO_NPS.

3.46.2.2 Database Link Information

NBX_GREV_AGENT_1@NPS1

3.46.2.3 Rerun Action

Rerun on failure.

3.47 Schedule BRDB_START_BKP

This schedule is run daily. It marks the start of the backup schedule.

3.47.1 Dependencies

Schedule BRDB_START_BKP depends on the completion of schedule BRDB_EOD.

3.47.2 Job COMPLETE

This job simply echoes a message before exiting.

3.47.2.1 Implementation

This job is implemented by a call to the echo command.

3.47.2.2 Rerun Action

None.

3.48 Schedule BRDB_BACKUP_0

This schedule is run on Sundays and Wednesdays. It performs the level 0 backup. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDB_LVL0_BACKUP is included here.

3.48.1 Dependencies

FUJ00234958

Schedule BRDB_BACKUP_0 depends on the completion of schedule BRDB_START_BKP.

3.48.2 Job BRDB_LVL0_BACKUP

This job performs the file transfer for the BRDB Branch Migration Status data feed.

3.48.2.1 Implementation

This job is implemented by a call to the shell script RMANBackup.sh with database name BRDB and level value 0.

3.48.2.2 Rerun Action

*** Prompts for rerun – action? **

3.49 Schedule BRDB_BACKUP_1

This schedule is run on every day **except** Sundays and Wednesdays. It performs the level 1 backup. It consists of a single task which can be run on any active node; see section 3.2 above for details. Only the parent job BRDB_LVL1_BACKUP is included here.

3.49.1 Dependencies

Schedule BRDB_BACKUP_1 depends on the completion of schedule BRDB_START_BKP.

3.49.2 Job BRDB_LVL1_BACKUP

Kicks off an RMAN level 1 backup.

3.49.2.1 Implementation

This job is implemented by a call to the shell script RMANBackup.sh with database name BRDB and level value 1.

3.49.2.2 Rerun Action

*** Prompts for rerun – action? **

3.50 Schedule BRDB_BKP_COMPL

This schedule is run daily. It checks that the backup schedule has completed and creates a flag file via the job CREATE_BRDB_BKUP_COMPLETE_FLAG.

3.50.1 Dependencies

Schedule BRDB_BKP_COMPLETE depends on the completion of whichever of schedule BRDB_BACKUP_0 or BRDB_BACKUP_1 that applies on the appropriate day.

3.50.2 Job CREATE_BRDB_COMPLETE_FLAG

This job creates the flag file /opt/tws/FLAGS/BRDB_BKUP_complete.FLAG.

3.50.2.1 Implementation

This job is implemented by a call to the “touch” command with the relevant file name.

3.50.2.2 Rerun Action

*** Prompts for rerun – action? **

3.51 Schedule BRDB_MONITOR

FUJ00234958

This schedule is run daily. It checks that other jobs have completed by a specified time. (See section 3.4.)

3.51.1 Dependencies

None

3.51.2 Job BRDB_MON_STARTUP

This checks that the BRDB_STARTUP job has completed by the required time . schedule completes by 05:30 hrs (or 06:30 hrs when a Level 0 backup runs).

3.51.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and time.

3.51.2.2 Rerun Action

None.

3.51.3 Job BRDB_MON_PAUSE_FEED1

This checks that the BRDB_PAUSE_FEED1 job has completed by the required time of 07:59.

3.51.3.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and time.

3.51.3.2 Rerun Action

None.

3.51.4 Job BRDB_MON_AUD_FEED

This checks that the BRDB_AUD_FEED job has completed by the required time of 04:00 ***(or 03:00??)**.

3.51.4.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and time.

3.51.4.2 Rerun Action

None.

3.51.5 Job BRDB_MON_EOD

This checks that the BRDB_EOD job has completed by the required time of 04:00.

3.51.5.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and time.

3.51.5.2 Rerun Action

None.

FUJ00234958

3.52 Schedule BRDB_POE_LOAD

This schedule is run daily and converts any available POLSAP PDF documents into PNG format and loads into table OPS$BRDB.BRDB_EXT_FEED_REPORTS.

3.52.1 Job BRDBC038_POE_FROM_POLSAP

3.52.1.1 Implementation

This job calls executable BRDBC038 which will look for any PDFs in the POLSAP share directory (see table below for details). If no files are found then sleep for 600 seconds, look again - do this for 3 iterations and log an exception if no files found but exit 0.

The following is a list of directories used by this job: -

Note: The list is stored as values in the following table columns for the row *"WHERE ext_interface_feed_name = 'BRDB_POE_FROM_POLSAP'.*

Description	Column Name	Value
POLSAP share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/polsap
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/input
BRDB audit directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/poe
BRDB PNG load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.52.1.2 File Retention Periods

Processed PDF & PNG files (i.e. those with an uppercase extension) will be retained on the BRDB file system as per the metadata defined in BRDB_FILES_TO_HOUSEKEEP.

3.52.1.3 Rerun Action

Correct the root cause of the failure and rerun the job.

FUJ00234958

3.53 Schedule BRDB_PAFCD_LOAD

This schedule is run *every Sunday at 13:30* only and loads the latest PAF files (postcode data files) received from the Post Office, if available.

3.53.1 Job BRDBC038_PAF_FROM_CD

3.53.1.1 Implementation

For more on this process, please see Section 3.6.4

This job calls executable BRDBC038 in the following way: -

```
${BRDB_PROC}/BRDBC038 BRDB_PAF_FROM_CD ^BRDBBDAY^
```

BRDBC038 will attempt to find the PAF files, of the form `*compstc*.*.paf`, in the *INPUTSHARE_DIR_NAME* (see table below for details), register their existence within the database, copy them to *BRDB_INPUT_DIR_NAME* and then calls BRDBC040, which performs validation on the files and then calls a separate import process to load them.

The following is a list of directories used by this job: -

Note: The list is stored as values in the following table columns for the row *"WHERE ext_interface_feed_name = 'BRDB_PAF_FROM_CD'.*

Description	Column Name	Value
PAF (REF data) share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/support/working
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/input
BRDB audit directory	AUDIT_DIR_NAME	N/A
BRDB PAF load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.53.1.2 File Retention Periods

Processed PAF files - those with an uppercase extension, e.g. *.PAF - will be retained on the BRDB file system as per the metadata defined in BRDB_FILES_TO_HOUSEKEEP.

3.53.1.3 Failure Action

Determine the root cause and notify Support teams. Possible failures could include corrupt files, or spurious data, lack of disk space or other similar problems.

3.53.1.4 Rerun Action

None. The schedule will not need to be held.

3.54 Schedule BRDB_PAFADD_LOAD

This schedule is run *every day, including Sundays* and loads a PAF file received from the Post Office, which is different from the files delivered for the full PAF load (See Section 3.79).

3.54.1 Job BRDBC038_PAF_ADD_LOAD

3.54.1.1 Implementation

For more on this process, please see Section 3.6.5

This job calls executable BRDBC038 in the following way: -

```
${BRDB_PROC}/BRDBC038 BRDB_PAF_ADD_LOAD ^BRDBBDAY^
```

FUJ00234958

BRDBC038 will attempt to find the "additional data" PAF file, of the form **`*compstd*.*.paf`**, in the *INPUTSHARE_DIR_NAME* (see table below for details), register its existence within the database, copy it to *BRDB_INPUT_DIR_NAME*, also copy it to *AUDIT_DIR_NAME* and then calls BRDBC040, which performs validation on it and then calls a separate import process to load it.

The following is a list of directories used by this job: -

Note: The list is stored as values in the following table columns for the row *"WHERE ext_interface_feed_name = 'BRDB_PAF_ADD_LOAD'*.

Description	Column Name	Value
PAF (REF data) share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/support/working
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/input
BRDB audit directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/paf
BRDB PAF load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.54.1.2 File Retention Periods

Processed PAF files - those with an uppercase extension, e.g. *.PAF - will be retained on the BRDB file system as per the metadata defined in BRDB_FILES_TO_HOUSEKEEP.

3.54.1.3 Failure Action

Determine the root cause and notify Support teams. Possible failures could include a corrupt file, or spurious data within the file, lack of disk space or other similar problems.

3.54.1.4 Rerun Action

None. The schedule will not need to be held.

3.55 Schedule BRDB_TXN_POST_D

This schedule is run once every 60 minutes from BRDB_SOD until 17:00 and will attempt to post any outstanding/onhold CFD subfile transactions on a per fad hash basis.

3.55.1 Dependencies

This schedule depends on the completion of BRDB_SOD.

3.55.2 Job BRDBX053_POST_EXT_TXNS_1…4

3.55.2.1 Implementation

This job calls $BRDB_SH/BRDBX053.sh

3.55.2.2 Rerun Action

None.

3.56 Schedule BRDB_TXN_LOAD_EX

This schedule is run daily from 17:55. The schedule registers all relevant external transaction files into BRDB.

3.56.1 Dependencies

This schedule depends on the completion of BRDB_TXN_POST_D.

FUJ00234958

3.56.2 Job BRDBC038_PS_FROM_FDG

3.56.2.1 Implementation

Invokes BRDBC038 to scan & process the input share directory (populated by PODG) for Paystation transaction files (of the form PS??????????.TP_).

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name = 'PS'*.

Description	Column Name	Value
Share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/input_share
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/externaltxns
BRDB audit directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns
BRDB load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.56.2.2 Rerun Action

None.

3.56.3 Job BRDBC038_PG_FROM_FDG

3.56.3.1 Implementation

Invokes BRDBC038 to scan & process the input share directory (populated by PODG) for Post&Go transaction files (of the form PG??????????.TP_).

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name = 'PG'*.

Description	Column Name	Value
Share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/input_share
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/externaltxns
BRDB audit directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns
BRDB load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.56.3.2 Rerun Action

None.

3.57 Schedule BRDB_STOP_TLD

This schedule is run at 20:00. The schedule stops the CFD file daemons.

3.57.1 Dependencies

This schedule depends on the completion of BRDB_TXN_POST_D.

FUJ00234958

3.57.2 Job BRDBX011_STOP_PS

3.57.2.1 Implementation

Invokes BRDBX011.sh to stop the Paystation BRDBC038 file daemon.

3.57.2.1.1 Associated BRDB System Parameter

Parameter Name	Parameter Value	Description
PS_STOP_YN	Y or N	Controls the operation of the file daemon

3.57.2.2 Rerun Action

None.

3.57.3 Job BRDBX011_STOP_PG

3.57.3.1 Implementation

Invokes BRDBX011.sh to stop the Post&Go BRDBC038 file daemon.

3.57.3.1.1 Associated BRDB System Parameter

Parameter Name	Parameter Value	Description
PG_STOP_YN	Y or N	Controls the operation of the file daemon

3.57.3.2 Rerun Action

None.

3.58 Schedule BRDB_TXN_LOAD_D

This schedule is run daily at 18:00 until 20:00 and will validate and stage external transactions.

3.58.1 Dependencies

This schedule depends on the completion of BRDB_TXN_POST_D.

3.58.2 Job CREATE_BRDB_LOAD_FLAG

3.58.2.1 Implementation

touch /opt/tws/FLAGS/BRDB_Load.FLAG if not present, keep retrying until flag is not present.

3.58.2.2 Rerun Action

None.

3.58.3 Job BRDBC051_LOAD_TXNS

3.58.3.1 Implementation

After successfully recreating the flag, executes CFD validation and staging process BRDBC051 for the current TWS date.

3.58.3.2 Rerun Action

None.

FUJ00234958

3.58.4 Job BRDB_TXN_LOAD_SLEEP

3.58.4.1 Implementation

Sleep for 60 seconds.

3.58.4.2 Rerun Action

None.

3.58.5 Job BRDB_TXN_LOAD_RESUBMIT

3.58.5.1 Implementation

Resubmits schedule BRDB_TXN_LOAD_D until 19:59.

3.58.5.2 Rerun Action

None.

3.58.6 Job RM_BRDB_LOAD_FLAG

3.58.6.1 Implementation

Removes execution lock flag.

3.58.6.2 Rerun Action

None.

3.59 Schedule BRDB_TXN_ERRORS

This schedule is run daily at 20:05 to produce any error reports produced during the CFD validation process.

3.59.1 Dependencies

At 20:05 **Opens** "/opt/tws/FLAGS/BRDB_Load.FLAG" (! -f %p).

3.59.2 Job BRDBC052_TXN_ERRORS_PS

3.59.2.1 Implementation

If the execution flag from BRDB_TXN_LOAD_D is not present then execute BRDBC052 for Paystation.

3.59.2.2 Rerun Action

None.

3.59.3 Job BRDBC052_TXN_ERRORS_PG

3.59.3.1 Implementation

If the execution flag from BRDB_TXN_LOAD_D is not present then execute BRDBC052 for Post&Go.

3.59.3.2 Rerun Action

None.

3.59.4 Job BRDBC052_TXN_ERRORS_AT

3.59.4.1 Implementation

FUJ00234958

If the execution flag from BRDB_TXN_LOAD_D is not present then execute BRDBC052 for AT.

3.59.4.2 Rerun Action

None.

3.60 Schedule BRDB_PAYSTN

This schedule is run daily at 20:05 to produce any error reports produced during the CFD validation process.

3.60.1 Dependencies

At 20:05 **Opens** "/opt/tws/FLAGS/BRDB_Load.FLAG" (! -f %p).

3.60.2 Job BRDBX003_XDATA_TXN_TO_PS_1…4

3.60.2.1 Implementation

Invokes Oracle package PKG_BRDB_XDATA_TXN_TO_PS via BRDBX003.sh. This package updates table BRDB_F_ST_APS_TRANSACTIONS, column ADDITIONAL_DATA for Paystation only transactions using table RDDS_PS_PRODUCT_MAP (joined on product ID).

3.60.2.2 Rerun Action

None.

3.60.3 Job BRDBC008_CHECK_ XDATA_TXN_TO_PS

3.60.3.1 Implementation

Checks that all fad hashes have been successfully processed by BRDBX003_XDATA_TXN_TO_PS

3.60.3.2 Rerun Action

None.

3.61 Schedule BRDB_TXN_POST

This schedule follows BRDB_PAYSTN (on all available nodes) to post validated CFD transactions to the following BRDB tables

- BRDB_F_RX_APS_TRANSACTIONS.
- BRDB_F_RX_DCS_TRANSACTIONS
- BRDB_F_RX_EPOSS_TRANSACTIONS
- BRDB_F_RX_EPOSS_EVENTS
- BRDB_F_RX_NWB_TRANSACTIONS

3.61.1 Dependencies

This schedule depends on the completion of BRDB_PAYSTN.

3.61.2 Job BRDBC054

3.61.2.1 Implementation

This module confirms all sub files in BRDB_SUB_FILE_AUDIT have been processed (ie status != staging)

FUJ00234958

3.61.2.2 Rerun Action

None.

3.62 Schedule BRDB_LTD_AGG

This schedule is run daily and updates table BRDB_STOCK_UNIT_ASSOCIATIONS column LAST_TRADING_DATE.

3.62.1 Dependencies

This schedule relies on the completion of BRDB_XFR_COMPL.

3.62.2 Job BRDBX007_LAST_TRAD_DATE_AGGR_1…4

Updates LAST_TRADING_DATE on a per fad_hash basis.

3.62.2.1 Implementation

Calls BRDBX007.sh with a parameter of LAST_TRADING_DATE

3.62.2.2 Rerun Action

*** Prompts for rerun – action? **

3.63 Schedule BRDB_EXT_REP

This schedule is run daily and invokes the Generic Reporting Mechanism to create reports associated with Client File deliveries.

3.63.1 Dependencies

This schedule relies on the completion of BRDB_XFR_COMPL and BRDB_LTD_AGG.

3.63.2 Job GENERIC_CREATE_REPORT_VIEWS

Recreates the views required for the generic reporting mechanism.

3.63.2.1 Implementation

Calls GREPX001.sh

3.63.2.2 Rerun Action

*** Prompts for rerun – action? **

3.63.3 Job GENERIC_CREATE_EXT_REPORTS

Creates the reports required for CFD.

3.63.3.1 Implementation

This job is implemented by a call to the shell script GREPX002.sh.

Outputs files of the form…

Non_Polled_Terminals*.csv

Subfiles_On_Hold*.csv

4 Royal Mail Reports

- PSE_1_1_YYYYMMDD.XM_

FUJ00234958

- PSE_2_1_YYYYMMDD.XM_
- PSE_3_1_YYYYMMDD.XM_
- PSE_4_1_YYYYMMDD.XM_

to the following directories below.

Usage	BRDBBLV1 Environment Variable
Working directory	BRDB_MSU_WORKING
BRDB reports directory	BRDB_MSU_OUTPUT

Files in the working directory are immediately cleaned up on successful completion while files within the reports directory are removed after 9 days.

3.63.3.2 Rerun Action

If the contents of the PSE* files are found to be wrong, then it may be necessary to regenerate the files after any problems have been rectified. In this case, the following procedure must be followed:

1. The underlying reason for the incorrect data must be rectified (e.g. ensure that the correct reference or transaction data is present on the BRDB).

2. Make a note of the current value in table gen_rep_report_parameters column rep_effective_date (there is only one row in the table).

3. Make sure that the report schedules are not due to execute (either BRDB_EXT_REP or BRDB_GEN_REP). If they are then limit the schedules to prevent them running.

4. Update the branch database value to the Trading Date of the day you want to re-run (replace *yyyymmdd* with the date you want to re-extract):
```
UPDATE gen_rep_report_parameters SET rep_effective_date =
TO_DATE('yyyymmdd', 'YYYYMMDD');
```

5. Log onto a brdb node as the brdb user and run the following commands:
```
/app_sw/brdb/sh/GREPX002.sh 'REP:PSE_1_1'
/app_sw/brdb/sh/GREPX002.sh 'REP:PSE_2_1'
/app_sw/brdb/sh/GREPX002.sh 'REP:PSE_3_1'
/app_sw/brdb/sh/GREPX002.sh 'REP:PSE_4_1'
```

6. Reset gen_rep_report_parameters.rep_effective_date back to what it was (replace *yyyymmdd* with the date you made a note of earlier):
```
UPDATE gen_rep_report_parameters SET rep_effective_date =
TO_DATE('yyyymmdd', 'YYYYMMDD');
```

7. If any TWS schedules were held in step 3 then release them.

NOTE: The remaining two steps should **not** be run when the relevant PODG route is available, since the output files must be renamed before PODG picks them up for processing. The PODG route is normally available between 01:00 and 06:00.

FUJ00234958

8. Run the following command to post-process the files and rename them with the correct extension (replacing *yyyymmdd* with the date you are re-running for):
   ```
   /app_sw/brdb/sh/BRDBX043.sh yyyymmdd
   ```

9. Rename the files in /app/brdb/trans/support/reportoutput to the number that you have agreed with POL. The files will have a suffix of '.XML'. E.g. if the value agreed was '2' and the date was 04/11/2014:
   ```
   mv PSE_1_1_20141104.XML PSE_1_2_20141104.XML
   mv PSE_2_1_20141104.XML PSE_2_2_20141104.XML
   mv PSE_3_1_20141104.XML PSE_3_2_20141104.XML
   mv PSE_4_1_20141104.XML PSE_4_2_20141104.XML
   ```

The files should now be available for PODG to zip and transfer.

3.63.4 Job BRDBX043

Checks all expected Royal Mail Extended Data reports are present. BRDBX043.sh adds a XML header to each file, counts the number of detail records and then adds a XML trailer containing the record count.

3.63.4.1 Dependencies

This job waits until the completion of GENERIC_CREATE_EXT_REPORTS before running.

3.63.4.2 Implementation

This job is implemented by a call to the shell script $BRDB_SH/BRDBX043.sh YYYYMMDD.

The script carries out the following actions (where YYYYMMDD is the TWS date and 'n' is a number between 1 and 4 inclusive):

- Confirms that the 4 RM reports in $BRDB_MSU_OUTPUT/PSE_n_1_[YYYYMMDD].XM_ exist
- Creates a temporary copy of each file (postfixed with .TMP) in $BRDB_MSU_OUTPUT
- Adds a XML header to each PSE_n_1_YYYYMMDD.XM_.TMP file
- Counts the number of object lines in each file
- Adds a XML trailer to each PSE_n_1_YYYYMMDD.XM_.TMP file, including the object count
- Renames each PSE_n_1_YYYYMMDD.XM_.TMP to $BRDB_MSU_OUTPUT/PSE_n_1_YYYYMMDD.XML

Description	BRDBBLV1 Environment Variable
Working directory	BRDB_MSU_WORKING
BRDB reports directory	BRDB_MSU_OUTPUT

$BRDB_MSU_OUTPUT/PSE*.XM_ files are immediately cleaned up on successful completion of the job while $BRDB_MSU_OUTPUT/PSE*.XML files are removed after 7 days. Any $BRDB_MSU_WORKING/PSE*.XM_ files are removed after 4 days.

FUJ00234958

3.63.4.3 Rerun Action

If the contents of the PSE* files are found to be wrong, then it may be necessary to regenerate the files after any problems have been rectified. In this case, follow the procedure in section 3.94.3.2 above.

3.64 Schedule BRDB_BF_TO_BLCS

This schedule runs as a daemon. It polls for new Branch-Full events once every 20 minutes and it is stopped by the BRDB_PAUSE_BF schedule.

3.64.1 Dependencies

This schedule depends on the completion of BRDB_SOD

3.64.2 Job BRDBC055_BF_TO_BLCS_1…4

These jobs (one per node) create Branch-Full files and insert Branch-Full event transactions into the BRDB_BRANCH_FULL_EVENTS table.

3.64.2.1 Implementation

These jobs call executable BRDBC055 to poll for new Branch-Full events in the BRDB_RX_NRT_TRANSACTIONS (NRT) table. The job processes all outstanding entries in the NRT transaction table and, for each entry, it will calculate the current items on-hand summed separately by carrier that are of a status 'LCIn'.

The results of the calculation are stored in the BRDB_BRANCH_FULL_EVENTS table. In addition the results are written to a Branch-Full event interface file that is passed to the BLCS via PODG for process as part of its Capacity Management suit

3.64.2.2 Rerun Action

Alert Operations on failure.

3.65 Schedule BRDB_PAUSE_BF

This schedule is run at 18:00. It will terminate the BRDB_ BF_TO_BLCS schedule Branch-Full Events daemons.

3.65.1 Dependencies

At 18:00

3.65.2 Job BRDBX011_PAUSE_BF_TO_BLCS

3.65.2.1 Implementation

Invokes BRDBX011.sh to stop the BRDBC055 Branch-Full Event daemon.

3.65.2.1.1 Associated BRDB System Parameter

Parameter Name	Parameter Value	Description
BRDB_BRANCH_FULL_STOP_YN	Y or N	Controls the operation of the Branch-Full daemon

3.65.2.2 Rerun Action

None.

FUJ00234958

3.66 Schedule BRDB_BF_TO_CRED

This schedule is run daily to create Branch-Full file from BRDB_BRANCH_FULL_EVENTS table.

3.66.1 Dependencies

This schedule depends on the completion of BRDB_BF_TO_BLCS

3.66.2 Job BRDB_BF_TO_CREDENCE

This job creates a Branch-Full file per day for delivery to Credence via PODG at 18:30.

3.66.2.1 Implementation

This job calls a executable BRDBC056 to extract Branch Full data from the BRDB_BRANCH_FULL_EVENTS table each day. This will be written to a new interface file and delivered via PODG to Credence.

The filename is in the form of: `BFCYYYYMMDDHHMIN.XML`

Where

`BFC`	Static Prefix (Branch Full Credence)
`YYYY`	Year
`MM`	Month
`DD`	Day
`HH`	Hour in 24 hour format
`MI`	Minutes
`N`	The node that executed the BRDB process

The file is initial created in output local directory specified by the system parameter 'BFCS_OUTPUT' and then moved to output share directory specified by the system parameter 'BFCS_OUTPUT_SHARE'.

3.66.2.2 Rerun Action

Alert Operations on failure.

3.67 Schedule BRDB_IOH_TO_BLCS

This schedule is run daily to create Items On hand file from BRDB_PS_BARCODES table

3.67.1 Dependencies

This schedule depends on the completion of BRDBC038_CR_LOAD1_BRDBC058

3.67.2 Job BRDB_IOH_TO_BLCS

This schedule is run daily to create Items On hand file to be delivered to BLCS via PODG at 18:00.

3.67.2.1 Implementation

This job call to the executable BRDBC057 to extract a count of items on-hand summed separately by branch and carrier that are of a status 'LCIn'.The results will be written to a new interface file that will be delivered to the BLCS via PODG.

The filename is in the form of: `IOHYYYYMMDDHHMIN.XML`

FUJ00234958

Where

IOH	Static Prefix (Item On Hand)
YYYY	Year
MM	Month
DD	Day
HH	Hour in 24 hour format
MI	Minutes
N	The node that executed the BRDB process

The file is initial created in output local directory specified by the system parameter 'BFCS_OUTPUT' and then moved to output share directory specified by the system parameter 'BFCS_OUTPUT_SHARE'.

3.67.2.2 Rerun Action

Alert Operations on failure.

3.68 Schedule BRDB_CR_DESP

This schedule runs once a day to update the status of Items on Hand for the Paystation Direct Settlement branches.

3.68.1 Dependencies

Runs at 19:30

3.68.2 Job BRDBX061_CR_DESPATCH_SIM

This job runs daily to update the status of Items on Hand for the Paystation Direct Settlement branches.

.

3.68.2.1 Implementation

Invokes BRDBX061.sh which updates the status of Items on Hand from either "MailIn" or "ByPassDesp" to "OutOfOffice" when the associated branch is a Paystation Direct Settlement branch.

3.68.2.2 Rerun Action

Alert Operations on failure.

3.69 Schedule BRDB_CR_LOAD1

This schedule starts up (CR) file daemon. It loads (CR) files into BRDB.

3.69.1 Dependencies

Runs at 19:30

FUJ00234958

3.69.2 Job BRDBC038_CR_LOAD1_BRDBC058

This daemon job runs daily as a file input daemon to populate/update items on hand and Track&Trace messages via Ingenico CR_ files.

3.69.2.1 Implementation

This daemon job invokes BRDBC038 which looks for one or more Ingenico Collect&Return (CR) files (populated via PODG):

/app/brdb/trans/externalinterface/input_share/PS??????????.CR_

Each file is registered in BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of 'CR' before being moved into the following directory:

/app/brdb/trans/externalinterface/externaltxns

Once all relevant files have been registered, BRDBC038 invokes BRDBC058 (C&R load process) which loads MAL (mail) transactions from each file into BRDB_F_ST_MAL_TRANSACTIONS. Once the MAL transactions have been validated (BRDB_F_ST_MAL_TRANSACTIONS.IS_VALIDATED='Y', errors populated into BRDB_FILE_ERRORS), BRDBC058 populates the following tables:

BRDB_RX_TT_TRANSACTIONS

BRDB_PS_BARCODES

Any transactions failing validation are output to the following location:

/app/brdb/trans/externalinterface/output_share/PS??????????.CRX

The error files (CRX) are then picked up via PODG and transmitted to POL for checking. Note that CRX files are registered in BRDB_FILE_AUDIT_TRAIL.

3.69.2.2 Rerun Action

Alert Operations on failure.

If the child process (/app_sw/brdb/c/BRDBC058) fails then resolve the root cause and invoke the load process directly via user **brdbblv*n*** (where ***n*** = node the executable is invoked on) as per the following:

$> **$BRDB_PROC/BRDBC058**

Then restart the Daemon process:

$> $BRDB_PROC/BRDBC038 CR ^BRDBBDAY^

Note that BRDBC038 will not invoke child process BRDBC058 unless there are new files to register.

3.70 Schedule BRDBC038_CR_LOAD2_BRDBC058

This schedule is similar to schedule BRDBC038_CR_LOAD1_BRDBC058 except it runs at 07:30 ..

3.70.1 Dependencies

Runs at 07:30

3.70.2 Job BRDBC038_CR_LOAD2_BRDBC058

FUJ00234958

This daemon job is run daily at 07:30 to populate/update items on hand and Track&Trace messages via Ingenico CR_ files.

3.70.2.1 Implementation

Same as 3.98.2.1

3.70.2.2 Rerun Action

Same as 3.98.2.2

.

3.71 Schedule BRDB_LVL0_BACKUP

This setup the Level 0 (Full Database Backup) file (/backup/sbrdbbackup.tmp) to 0 as input for RMAN backup on the Standby Database. It runs on Sundays and Wednesday

3.71.1 Dependencies

The backup Diskgroups are required not to be mounted on both the standby and primary node. If at all there is a reboot of the server or the cluster ware at anypoint in time, the following Diskgroups must be unmounted immediatel across all the node.(both PRIMARY and STANDBY)

3.72 Schedule BRDB_LVL1_BACKUP

This runs the Level1 (incremental backup) file [/backup/sbrdbbackup.tmp] to 1 as input for RMAN backup on the Standby Database. This runs Everyday except Sunday and Wednesday.

3.72.1 Dependencies

Same as 3.101.1

3.73 Schedule BRDB_SBRDB_BACKUP

This schedule runs daily to monitor the RMAN backup on the standby server (BDS).This reduces workload on the PRIMARY sever. The backup on standby is configured in an RMAN catalog, which is stored on the EDS server which is managed by SMG and runs via a secure channel using Oracle wallets to authenticate connection.

3.73.1.1 Implementation

The monitor script (/usr/local/bin/MonitorSBRDBBackup.sh), monitors the status of the RMAN backup on the standby which runs as a cron job on the Standby server. It renames the file /backup/sbrdbbackup.tmp to /backup/sbrdbbackup.flag as long as there isn't any file name /backup/sbrdbbackup.done or /backup/sbrdbbackup.err is present.

If the backup completes without error, the /backup/sbrdbbackup.done is then removed by the Monitor job for the next days backup. If failure occurs, i.e the file /backup/sbrdbbackup.err , check what the issue is from the backup log, [/home/oracle/SRMANBackup_SBRDB_YYYYMMDD_hhmm .log] on Standby Node 1, and fix the error and manually run the backup on the Standby server, using the following command as UNIX user "Oracle"

COMMAND:-
/usr/local/bin/SRMANBackup.sh -v -d SBRDB -l 0|1

FUJ00234958

Once backup is complete, manually remove the error file /backup/sbrdbbackup.err .

3.74 Schedule BRDB_NRT_BAP_AGT

This schedule runs daily at 06:00 as a daemon. It polls for new NRT BAP events once every 15 minutes and it is stopped by the BRDB_PAUSE_BF schedule.

3.74.1 Dependencies

This schedule runs daily at 06:00

3.74.2 Job BRDBC060_BAP_AGT_1...4

These jobs (one per node) create BAP Pre-Advice files.

3.74.2.1 Implementation

These jobs call executable BRDBC060 on each node to poll for new BAP NRT event in BRDB_RX_NRT_TRANSACTIONS (NRT) table. It processes all unprocessed BAP NRT transactions with the *Client Name* set to "PS2DBarcode" and *Client Routing Name* set to "POLBAP". The XML fields within the *NRT_Payload* CLOB is then extracted and written to Pre-Advice file in csv format .

The filename is in the form of: `POL_PreAdvice3_<n>NNNNNNNN.zip`

Where

`n`	Node id <1-4>)
`NNNNNNNN`	specific rolling file sequence number, 00000000 to 99999999

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name = 'BRDBC060'.*

Description	Column Name	Value
OUTPUT Share directory	OUTPUTSHARE_DIR_NAME	/app/brdb/trans/externalinterface/output_share
OUTPUT local directory	BRDB_OUTPUT_DIR_NAME	/app/brdb/trans/externalinterface/output

The file is initial created in output local directory , then zipped and copied to output share directory which will pick up by PODG for transmission to Royal Mail.

Polling interterval is specified in BRDB_EXT_INTERFACE_FEEDS.sleep_repeat_secs and currently set to 900 seconds (15 minutes).

Max record size is specified in BRDB_SYSTEM_PARAMETER. BAP_MAX_RECORD_NO and currently set to 24500

3.74.2.2 Failure Action

Determine the root cause and notify Support teams. Possible failures are:

1) File creation, space or other file manipulation issues.
2) Errors in reading and updating NRT Transactions table.

FUJ00234958

3) NRT_Payload greater than BAP_MAX_CLOB_LEN

4) NRT_Payload XML malformed (e.g. more than 3 Detail Supplement Records, invalid field content)

Note: (3) & (4) will not cause the program abend but instead copied the error record to brdb_host_interface_feed_excp table and mark the record as processed.All the errors are captured in BRDB_OPERATIONAL_EXCEPTIONS table and the *NRT_Payload* CLOB records are copied to BRDB_HOST_INTERFACE_FEED_EXCP table.

3.74.2.3 Rerun Action

Alert Operations on failure.

If the job abend and the root cause cannot resolved immediate. The only solution is to skip the error record by setting the following column value in BRDB_RX_NRT_TRANSACTIONS table

```
processed_yn          = 'Y',
selected_yn           = 'Y',
processed_timestamp   = SYSTIMESTAMP,
update_timestamp      = SYSTIMESTAMP
```

3.75 Schedule BRDB_PAUSE_BAP

This schedule runs at 23:15. It will terminate the BRDB_ NRT_BAP_AGT schedule NRT BAP daemons.

3.75.1 Dependencies

At 18:00

3.75.2 Job BRDBX011_PAUSE_BAP_AGT

3.75.2.1 Implementation

Invokes BRDBX011.sh to stop the BRDBC060 BAP AGT daemon.

3.75.2.1.1 Associated BRDB System Parameter

Parameter Name	Parameter Value	Description
BRDBC060_STOP_YN	Y or N	Controls the operation of the NRT BAP daemon

3.75.2.2 Rerun Action

Alert Operations on failure.

3.76 Schedule BRDB_PPK_LOAD

This schedule runs the loader that loads Pin Pad Key files into the BRDB. See SVM/SDM/OLA/1855, DES/SYM/HLD/0012 and REQ/APP/AIS/1833 for details. The loading of a set of Pin Pad Key files will occur very infrequently; this job will normally find no keys to load, and return success.

3.76.1 Dependencies

This schedule runs after BRDB_SOB, i.e. daily at 19:10

3.76.2 Job BRDBC038_PPK_FROM_KSN

This job runs daily to load Pin Pad Key files.

FUJ00234958

3.76.2.1 Implementation

This job invokes BRDBC038 which looks for Pin Pad Key files on the NAS FSA:

/app/brdb/trans/externalinterface/input_share/*bbbb_pppppp*.ekf

The "*bbbb*" denotes the hexadecimal Banking Key Identifier (BKID) of the key set, and "*pppppp*" is the decimal Pin Pad Identifier (PPID) of the Pin Pad for which the key is valid. Each file is registered as new ('N') in BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of 'BRDB_PPK_FROM_KSN' before being moved into the following directory:

/app/brdb/trans/externalinterface/input

Once all relevant files have been registered, BRDBC038 invokes BRDBC062 (the Pin Pad Key Loader process) which then validates and loads the files.

Validation involves checking that the BKID of the key files to be loaded are all the same, and greater than the key set currently loaded, which is stored (as a number) in the BRDB system parameter PPK_VERSION. As each file is validated, it is marked as validated ('V') in BRDB_FILE_AUDIT_TRAIL and copied to the following directory:

/app/brdb/trans/externalinterface/loaddir

Once all the files have been validated, they are then loaded. This involves inserting each as a new record in the table indicated by the synonym BRDB_PED_KEYS_SAV, which is one of table BRDB_PED_KEYS_A/B. The file content is inserted as a BLOB.

Once all the files have been loaded, the new set of files is made available to the BAL by recreating the the synonym BRDB_PED_KEYS to point at the table with the newly loaded keys. The system parameter PPK_TABLE_SET is updated to indicate this table (A/B). The synonym BRDB_PED_KEYS_SAV is recreated to point at the table containing the previous set of keys, which will be used during the next load. The keys themselves are marked as complete ('C') in BRDB_FILE_AUDIT_TRAIL, and the key files are renamed to '.EKF' in both the input and loaddir directories.

Any errors during validation or loading cause the loader to exit immediately, after setting the file status to error ('E') and renaming the key files to '.EKF'.

3.76.2.2 Rerun Action

Alert Operations on failure.

If either BRDBC038 or the child process BRDBC062 fails then the root cause should be investigated. As well as validating the keys as documented above, BRDBC062 is designed to exit with an error if any '.ekf' or '.EKF' files are found in the input_share directory, since that might indicate a problem during the running of BRDBC038 (e.g. an attempt to load some keys already loaded previously).

The most likely error scenario is that a problem with the key files to load is found (e.g. the key files don't all have the same BKID, or have a BKID the same as or earlier than the keys already loaded). Although BRDBC062 is designed to be rerunnable (e.g. it will check for previously validated key files and load them if appropriate), and can be run on its own without calling BRDBC038, the simplest approach will be to clear out any keys not yet loaded, and start by running BRDBC038 again. This will involve removing the following:

- Any unrequired '.ekf' files in **/app/brdb/trans/externalinterface/input_share**.
- Any '.EKF' files in **/app/brdb/trans/externalinterface/input_share**.
- Any '.ekf' or '.EKF' files related to keys not yet loaded in **/app/brdb/trans/externalinterface/input** and **/app/brdb/trans/externalinterface/loaddir**.
- Any rows in BRDB_FILE_AUDIT_TRAIL relating to keys not yet loaded. In particular, rows with status 'N' or 'V', and rows with status 'E' where the key needs to be loaded in the future.

FUJ00234958

The correct set of keys to load should then be placed in **`/app/brdb/trans/externalinterface/input_share`**.

If wishing to attempt to reload the keys immediately, run the loader process as follows:

```
$> $BRDB_PROC/BRDBC038 BRDB_PPK_FROM_KSN ^BRDBBDAY^
```

Alternatively, the keys can be left to load the next time the schedule runs.

Note that BRDBC038 will not invoke child process BRDBC062 unless there are new files to register.

3.77 Schedule BRDB_START_EUM

This schedule runs every 10 minutes from 8:02 until 20:00. The schedule registers all EUM ForgeRock files into BRDB

3.77.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule.

3.77.2 Job BRDBC038_EUM_FROM_FORGEROCK

3.77.2.1 Implementation

Invokes BRDBC038 to scan & process the input share directory (populated by PODG) for ForgeRock files (of the form FR*.XM_).

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name* = 'BRDB_EUM_FORGEROCK_LOADER'.

Description	Column Name	Value
Share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/input_share
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/externaltxns
BRDB audit directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns
BRDB load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.77.2.2 Rerun Action

None

3.78 Schedule BRDB_EUM_LOAD

This schedule runs once and calls a daemons process which polls every 10 minutes from 8:05 until 20:00. The schedule loads all the register EUM ForgeRock files into BRDB tables

3.78.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule then follows TWO_MIN_WAIT.

3.78.2 Job BRDBC066_EUM_FORGEROCK_LOAD

3.78.2.1 Implementation

FUJ00234958

This job call to the executable BRDBC066 to load the EUM ForgeRock files which are specified from BRDB_FILE_AUDIT_TRAIL (where process_name = 'BRDB_EUM_FORGEROCK_LOADER' and file_status = 'N').

BRDBC066 populates the following tables:

- BRDB_POID_USER_DETAILS
- BRDB_POID_CURRICULA

3.78.2.2 Rerun Action

Alert Operations on failure.

Resolve the root cause and rerun the job again.

3.79 Schedule BRDB_STOP_EUM

This schedule is run daily at 20.00.

3.79.1 Dependencies

This schedule depends on the BRDB_START_EUM schedule being started

3.79.2 Job BRDBX011_STOP_EUM_LOAD

3.79.3 Implementation

Invokes BRDBX011.sh to stop both the EUM ForgeRock BRDBC038 file daemon and BRDBC066 EUM ForgeRock Loader daemon.

3.79.4 Rerun Action

Alert Operations on failure

3.80 Schedule BRDB_CHK_CRED

This schedule is run daily to alert Operation if an error file returned from Credence.

3.80.1 Dependencies

This schedule depends on completion of BRDB_SOD schedule.

3.80.2 Job BRDBX065_CHECK_CREDENCE_ERRORS

This job runs daily to alert operation if error files returning from Credence.

3.80.3 Implementation

This job invokes BRDBX065.sh which looks for error files from Credence on the input share directory:

`/app/brdb/trans/externalinterface/input_share/C*.TP[X|Z]`

An operational exception will insert into BRDB_OPERATIONAL_EXCEPTION table if the error file(s) exist and the file(s) will move to the local input directory:

`/app/brdb/trans/externalinterface/input`

3.80.4 Rerun Action

Alert Operations on failure

3.81 Schedule BRDB_CREDENCE

This schedule is run daily to generate Credence files.

FUJ00234958

3.81.1 Dependencies

This schedule follows BRDB_TXN_POST schedule.

3.81.2 Job BRDBX063_CREDENCE_CREATE_1..4

These jobs (one per node) perform Credence file generation.

3.81.2.1 Implementation

These jobs call the shell script BRDBX063.sh to produce Credence TP_ files in directory /app/brdb/trans/externalinterface/output.

These jobs will produce one file per FAD Hash which means that a total of 128 files will be created across four BRDB instances and at the same time the value of, record count, total amount and total quantity for each branch is inserted into the BRDB_CREDENCE_FILE_TOTALS table.

All 128 files will be registered in BRDB_FILE_AUDIT_TRAIL table together with the total number of data records inserted into the TOTAL_RECORDS column

The filename of the Credence file will be structured as below:

C_DDDHHH.TP_

When:

C_	= the destination
DDD	= transmission day number (001 to 366)
HHH	= file number and in this case it is the Fad Hash value plus one (001-128)
TP_	= the filename extension

A reconciliation report will be performed between the sub-file-audit records in the BRDB_CREDENCE_FILE_TOTALS table and the branch database source tables to prove that all data has been delivered as a record within one of the files. This will be implemented as a generic report (specified in section GEN_REP.).The location and the report file name is as below:

/app/brdb/trans/support/reportoutput/Credence_Extract_Reconciliation_*YYYYMMDD*.csv

This file will be delivered into the corporate domain using the RDT PODG system.

3.81.2.2 Rerun Action

Alert Operations on failure

3.81.3 Job BRDBC008_CHECK_CREDENCE

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.81.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDB_TO_CREDENCE.

3.81.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.82 Schedule BRDB_ZIP_CRED

FUJ00234958

This schedule runs daily to take all the files that have been created in BRDB_CREDENCE schedule and zip them into four files.

3.82.1 Dependence

This schedule follows the completion of BRDB_CREDENCE schedule

3.82.2 Job BRDBX064_CREDENCE_ZIP

This job runs daily to zip up all the files that have been created into four files.

3.82.3 Implementation

This job is implemented by a call to the shell script BRDBX064.sh specifying the BISJDAY (ddd).

The BRDBX064.sh shell script performs the followings:

- clear down any previous attempts to create zip files i.e.

  /app/brdb/trans/externalinterface/output_share/tmp_C_${BISJDAY}_?.zip
- zip the files in app/brdb/trans/externalinterface/output/C_${ BISJDAY}*.TP_ to four zip files, i.e.

  /app/brdb/trans/externalinterface/output_share/tmp_C_${BISJDAY}_N.zip

  Where N = 1 to 4
- rename filename from 'tmp_C_${BISJDAY}_N.zip' to 'C_${BISJDAY}_N.ZIP' and register the files in BRDB_FILE_AUDIT_TRAIL

BRDBX064.sh utilise the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Tables BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. This table (BRDB_PROCESS_CONTROL) is used to enforce requirements such as ensuring that BRDBX064.sh can only be run once for a given BISJDAY

3.82.4 Rerun Action

Alert Operations on failure

3.83 Schedule BRDB_CHK_CFS

This schedule is run daily to alert Operations if an error file is returned from CFS.

3.83.1 Dependencies

This schedule depends on completion of the BRDB_SOD schedule.

3.83.2 Job BRDBX070_CHECK_CFS_ERRORS

This job runs daily to alert operation if error files are returned from CFS.

3.83.3 Implementation

This job invokes BRDBX070.sh which looks for error files from CFS in the input share directory:

`/app/brdb/trans/externalinterface/input_share/BTF*.ERR`

An operational exception will be inserted into the BRDB_OPERATIONAL_EXCEPTIONS table if one or more error files exist, and the file(s) will be moved to the local input directory:

`/app/brdb/trans/externalinterface/input`

3.83.4 Rerun Action

Alert Operations on failure

FUJ00234958

3.84 Schedule BRDB_BTF_FILES

This schedule is run daily to generate CFS interface files.

3.84.1 Dependencies

This schedule follows the BRDB_TXN_POST schedule.

3.84.2 Job BRDBX068_BTF_CREATE_1..4

These jobs (one per node) perform CFS interface file generation.

3.84.2.1 Implementation

These jobs are implemented as a call to the shell script BRDBX068.sh with file type "BTF" and the business day (YYYYMMDD) and this in turn invokes package procedure PKG_BRDB_TO_CFS.generate_cfs_files for the specified BRDB instance to create CFS interface files in the /app/brdb/trans/externalinterface/output directory.

The procedure loops through all fad hash values associated with the input instance ID and will produce one file per fad hash which means that a total of 128 files will be created per day. The CFS interface file will be registered as new ('N') in the BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of 'BRDB_TO_CFS' when first created. It will be updated to 'C' when all the data records for a particular fad hash have been successfully written to the file, and the total number of data records is written to the TOTAL_RECORDS column.

The application also checks that each branch accounting code's transactions balance to a zero sum. Any branches that don't balance are logged in BRDB_OPERATIONAL_EXCEPTIONS, although the transactions are still output and processing continues.

The PKG_BRDB_TO_CFS.generate_cfs_files procedure utilises the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that each fad hash (i.e. CFS interface file) can only be run once for a given business day.

The filename of the CFS interface file is structured as below:

BTF*YYYYMMDDnnn*.csv

Where:

BTF_	= the fixed file prefix
YYYYMMDD	= current date
nnn	= file number, which is the Fad Hash value **plus one** (001-128)
.csv	= fixed file extension, denoting comma separated values format

3.84.2.2 Rerun Action

Alert Operations on failure

3.84.3 Job BRDBC008_CHECK_BTF

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.84.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDB_TO_CFS.

FUJ00234958

3.84.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.85 Schedule BRDB_ZIP_BTF

This schedule runs daily to take all the files that have been created in the BRDB_BTF_FILES schedule and zip them into four files.

3.85.1 Dependencies

This schedule follows the completion of the BRDB_BTF_FILES schedule

3.85.2 Job BRDBX069_BTF_ZIP

This job runs daily to zip up all the files that have been created into four files.

3.85.3 Implementation

This job is implemented as a call to the shell script BRDBX069.sh specifying the file type "BTF" and the business day (YYYYMMDD).

The BRDBX069.sh shell script performs the following:

- clear down any previous attempts to create zip files i.e.

  /app/brdb/trans/externalinterface/output_share/tmp_BTF*.zip
- zip the files app/brdb/trans/externalinterface/output/BTF*YYYYMMDD**.csv to four zip files, i.e.

  /app/brdb/trans/externalinterface/output_share/tmp_BTF*YYYYMMDD_N*.zip

  Where N = 1 to 4
- rename filename from tmp_BTF*YYYYMMDD_N*.zip to BTF*YYYYMMDD_N*.zip and register the files in BRDB_FILE_AUDIT_TRAIL

BRDBX069.sh utilises the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Tables BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that BRDBX069.sh can only be run once for a given business day.

3.85.4 Rerun Action

Alert Operations on failure

3.86 Schedule BRDB_BTR_FILES

This schedule is run daily to generate CFS reconciliation files.

3.86.1 Dependencies

This schedule follows the BRDB_ZIP_BTF schedule.

3.86.2 Job BRDBX068_BTR_CREATE_1..4

These jobs (one per node) perform CFS reconciliation file generation.

3.86.2.1 Implementation

These jobs are implemented as a call to the shell script BRDBX068.sh with file type "BTR" and the business day (YYYYMMDD) and this in turn invokes package procedure PKG_BRDB_CFS_RECON.generate_recon_files for the specified BRDB instance to create CFS reconciliation files in the /app/brdb/trans/externalinterface/output directory.

FUJ00234958

The procedure loops through all fad hash values associated with the input instance ID and will produce one file per fad hash which means that a total of 128 files will be created per day. The CFS reconciliation file will be registered as new ('N') in the BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of 'BRDB_CFS_RECON' when first created. It will be updated to 'C' when all the data records for a particular fad hash have been successfully written to the file, and the total number of data records is written to the TOTAL_RECORDS column.

The PKG_BRDB_CFS_RECON.generate_recon_files procedure utilises the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that each fad hash (i.e. CFS reconciliation file) can only be run once for a given business day.

The filename of the CFS reconciliation file is structured as below:

BTR*YYYYMMDDnnn*.csv

Where:

BTR_	= the fixed file prefix
YYYYMMDD	= current date
nnn	= file number, which is the Fad Hash value **plus one** (001-128)
.csv	= fixed file extension, denoting comma separated values format

3.86.2.2 Rerun Action

Alert Operations on failure

3.86.3 Job BRDBC008_CHECK_BTR

This job checks for the successful completion of the previous job for all FAD-Hashes.

3.86.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDB_CFS_RECON.

3.86.3.2 Rerun Action

As specified in section 3.1, alert Operations if rerun fails.

3.87 Schedule BRDB_ZIP_BTR

This schedule runs daily to take all the files that have been created in the BRDB_BTR_FILES schedule and zip them into four files.

3.87.1 Dependencies

This schedule follows the completion of the BRDB_BTR_FILES schedule

3.87.2 Job BRDBX069_BTR_ZIP

This job runs daily to zip up all the files that have been created into four files.

3.87.3 Implementation

This job is implemented as a call to the shell script BRDBX069.sh specifying the file type "BTR" and the business day (YYYYMMDD).

The BRDBX069.sh shell script performs the following:

FUJ00234958

- clear down any previous attempts to create zip files i.e.

  /app/brdb/trans/externalinterface/output_share/tmp_BTR*.zip
- zip the files app/brdb/trans/externalinterface/output/BTR*YYYYMMDD**.csv to four zip files, i.e.

  /app/brdb/trans/externalinterface/output_share/tmp_BTR*YYYYMMDD_N*.zip

  Where N = 1 to 4
- rename filename from tmp_BTR*YYYYMMDD_N*.zip to BTR*YYYYMMDD_N*.zip and register the files in BRDB_FILE_AUDIT_TRAIL

BRDBX069.sh utilises the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Tables BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that BRDBX069.sh can only be run once for a given business day.

3.87.4 Rerun Action

Alert Operations on failure

3.88 Schedule BRDB_BOI_FILE

This schedule runs daily to generate Bank of Ireland Client File.

3.88.1 Dependencies

This schedule follows the completion of the BRDB_SOB schedule

3.88.2 Job BRDBX071_CREATE_BOI_FILE

This job runs daily to generate Bank of Ireland Client File.

3.88.3 Implementation

This job is implemented as a call to the shell script BRDBX071.sh specifying the business day (YYYYMMDD).

The BRDBX071.sh shell script performs the following: The filename format is as follows:

ATMR***nnnn***.CIF

There is one file produced per batch window, with each file identified by the filename itself.

Each filename contain a numeric value (represented by ***nnnn*** above) which starts at 1 and has a maximum value of 9999. The accepted design pattern is to hold this value as a numeric system parameter in BRDB_SYSTEM_PARAMETERS, this parameter to be called ATM_FILE_SEQUENCE_NO.

Each business day it produce one ATMRnnnn.CIF file, with the nnnn increasing by one each day. The new filename should be recorded in table BRDB_FILE_AUDIT_TRAIL and retained in that table for a year.

3.88.4 Rerun Action

Alert Operations on failure

3.89 Schedule BRDB_CSH_TO_CWC

This schedule runs daily to generate file for flexible cash planning CWC application via PODG. Four interface file is generated.

3.89.1 Dependencies

FUJ00234958

This schedule follows the BRDB_SOB schedule.

3.89.2 Job BRDBX072_COH_FILE_TO_CWC

This job generates Cash on hand interface file.

3.89.2.1 Implementation

This job is implemented as a call to the shell script BRDBX072.sh with file type "COH" and the business day (YYYYMMDD) as parameter and this in turn invokes package procedure PKG_BRDB_COH_TO_CWC. generate_COH_InterfaceFile for the specified BRDB instance to create cash on hand interface file in the /app/brdb/trans/externalinterface/output_share directory.

The procedure loops through all fad hash values associated with the input instance ID and will produce a single file per day. The Cash on hand interface file will be registered as new ('N') in the BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of ' BRDB_COH_TO_CWC' when first created. It will be updated to 'C'.

The PKG_BRDB_COH_TO_CWC. generate_COH_InterfaceFile procedure utilises the existing 'process audit' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_AUDIT can be queried for this information.

The filename of the Cash on hand interface file is structured as below:

HORCWCCOH9999.xml

Where "9999" is stored as a system parameter COH_FILE_SEQUENCE that is incremented after use. Files will initially be written to /app/brdb/trans/externalinterface/output and will be moved to /app/brdb/trans/externalinterface/output_share to make them available to PODG. The commit on the system parameter will only take place once the move has completed – this makes the process re-runnable should any failure occur.

3.89.2.2 Rerun Action

Alert Operations on failure

3.89.3 Job BRDBX072_CIP_FILE_TO_CWC

This job generates Cash in Pouch interface file.

3.89.3.1 Implementation

This job is implemented as a call to the shell script BRDBX072.sh with file type "CIP" and the business day (YYYYMMDD) and this in turn invokes package procedure PKG_BRDB_CIP_TO_CWC. generate_CIP_InterfaceFile for the specified BRDB instance to create cash in pouch interface file in the /app/brdb/trans/externalinterface/output_share directory.

The procedure loops through all fad hash values associated with the input instance ID and will produce a single file per day. The Cash in pouch interface file will be registered as new ('N') in the BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of ' BRDB_CIP_TO_CWC' when first created. It will be updated to 'C'.

The PKG_BRDB_CIP_TO_CWC. generate_CIP_InterfaceFile procedure utilises the existing 'process audit' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_AUDIT can be queried for this information.

The filename of the Cash in pouch interface file is structured as below:

HORCWCCIP9999.xml

FUJ00234958

Where "9999" is stored as a system parameter CIP_FILE_SEQUENCE that is incremented after use. Files will initially be written to /app/brdb/trans/externalinterface/output and will be moved to /app/brdb/trans/externalinterface/output_share to make them available to PODG. The commit on the system parameter will only take place once the move has completed – this makes the process re-runnable should any failure occur.

3.89.3.2 Rerun Action

Alert Operations on failure

3.89.4 Job BRDBX072_PAY_FILE_TO_CWC

This job generates cash out payment file.

3.89.4.1 Implementation

This job is implemented as a call to the shell script BRDBX072.sh with file type "PAY" and the business day (YYYYMMDD) and this in turn invokes package procedure PKG_BRDB_PAY_TO_CWC. generate_PAY_InterfaceFile for the specified BRDB instance to create cash out payment interface file in the /app/brdb/trans/externalinterface/output_share directory.

The procedure loops through all fad hash values associated with the input instance ID and will produce a single file per day. The Cash out payments interface file will be registered as new ('N') in the BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of ' BRDB_PAY_TO_CWC' when first created. It will be updated to 'C'.

The PKG_BRDB_PAY_TO_CWC. generate_PAY_InterfaceFile procedure utilises the existing 'process audit' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_AUDIT can be queried for this information.

The filename of the cash out payment file is structured as below:

HORCWCPAY9999.xml

Where "9999" is stored as a system parameter PAY_FILE_SEQUENCE that is incremented after use. Files will initially be written to /app/brdb/trans/externalinterface/output and will be moved to /app/brdb/trans/externalinterface/output_share to make them available to PODG. The commit on the system parameter will only take place once the move has completed – this makes the process re-runnable should any failure occur.

3.89.4.2 Rerun Action

Alert Operations on failure

3.89.5 Job BRDBX072_DEP_FILE_TO_CWC

This job generates cash in payments file.

3.89.5.1 Implementation

This job is implemented as a call to the shell script BRDBX072.sh with file type "DEP" and the business day (YYYYMMDD) and this in turn invokes package procedure PKG_BRDB_DEP_TO_CWC. generate_DEP_InterfaceFile for the specified BRDB instance to create cash in payment interface file in the /app/brdb/trans/externalinterface/output_share directory.

The procedure loops through all fad hash values associated with the input instance ID and will produce a single file per day. The Cash in payment interface file will be registered as new ('N') in the BDB table

FUJ00234958

BRDB_FILE_AUDIT_TRAIL with a process_name of ' BRDB_DEP_TO_CWC' when first created. It will be updated to 'C'.

The PKG_BRDB_DEP_TO_CWC. generate_DEP_InterfaceFile procedure utilises the existing 'process audit' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_AUDIT can be queried for this information.

The filename of the Cash in payment file is structured as below:

HORCWCDEP9999.xml

Where "9999" is stored as a system parameter DEP_FILE_SEQUENCE that is incremented after use. Files will initially be written to /app/brdb/trans/externalinterface/output and will be moved to /app/brdb/trans/externalinterface/output_share to make them available to PODG. The commit on the system parameter will only take place once the move has completed – this makes the process re-runnable should any failure occur.

3.89.5.2 Rerun Action

Alert Operations on failure

3.90 Schedule BRDB_TC_LOAD

This schedule runs daily to process Transaction Correction file from CFS application via PODG. It also generate error file if any.

3.90.1 Dependencies

This schedule follows the BRDB_SOB schedule.

3.90.2 Job BRDBC038_TC_FROM_CFS

This job register transaction correction file into BRDB.

3.90.2.1 Implementation

Invokes BRDBC038 to scan & process the input share directory (populated by PODG) for Transaction Corrections files (file name format *if<yyyymmdd><nnn>.tcn*).

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name = 'TC'.*

Description	Column Name	Value
Share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/input_share
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/externaltxns
BRDB output share directory	OUTPUTSHARE_DIR_NAME	/app/brdb/trans/externalinterface/output_share
BRDB load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir

3.90.2.2 Rerun Action

Alert Operations on failure

3.90.3 Job BRDBC073_LOAD_TC

This job process transaction correction file in BRDB.

3.90.3.1 Implementation

FUJ00234958

This job is implemented as a call to the executable BRDBX073 with file type "TC" and the business day (YYYYMMDD).

The job will look for all the unprocessed file from BRDB_FILE_AUDIT_TRAIL. For each file with a valid name, the file is pre-processed by an "awk" script.

This involves

a) checking the structure of the file is correct i.e. It contains:
   - 1 header line, with correct number of columns
   - 1 or more detail lines, with correct number of columns
     or
     no detail lines
   - 1 trailer line, with correct number of columns

b) checking the header line is correct i.e. all columns contain valid values

c) line count and checksums are correct for the preceding detail lines.

The output from this process is a single file called "TXN_CORRECTIONS" in the local directory identified by:

BRDB_EXT_INTERFACE_FEEDS.brdb_load_dir_name

containing all the "valid" Transaction Correction detail lines for all the new input files which had valid headers and trailers

Validate and Load:

This process attach the ASCII file "TXN_CORRECTIONS" as external table BRDB_F_EX_TC_Detail within the Branch Database. If there are any records where the format of the record does not match the format of the database table, then the record will be written to a BAD file. This will occur in instances where a date format is incorrect or where alpha characters are found where only numeric characters were expected. Following the file attachment, the BAD file will be read and each record in the BAD file will be written to table BRDB_FILE_ERRORS the record number in error will be taken from the BAD file.

Data will then be validated and any errors cause just the record containing the error to be rejected. Rows are written to BRDB_FILE_ERRORS to identify all the errors in the rejected record. Further these records will be written to an error file if<yyyymmdd><nnn>.err

All valid and complete rows are written to the TPS_TXN_CORRECTION_DETAILS and an audit of the transaction written to BRDB_TC_RECEIVED..

3.90.3.2 Rerun Action

Alert Operations on failure

3.91 Schedule BRDB_START_PLO

This schedule runs daily to process Planned order file from CWC application via PODG. It also generate error file if any.

3.91.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule.

3.91.2 Job BRDBC038_PLO_FROM_CWC

This job register Planned order file into BRDB.

3.91.2.1 Implementation

FUJ00234958

Invokes BRDBC038 to scan & process the input share directory (populated by PODG) for Planned Orders files (file name format Serviceorder*.xml).

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name* = 'BRDB_PLO_LOADER'.

Description	Column Name	Value
Share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/input_share
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/externaltxns
BRDB output share directory	OUTPUTSHARE_DIR_NAME	/app/brdb/trans/externalinterface/output_share
BRDB load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir
BRDB Audit Directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns

3.91.2.2 Rerun Action

None

3.92 Schedule BRDB_PLO_LOAD

This schedule runs once and calls a daemons process which polls every 2 minutes from 8:05. The schedule loads all the registered Planned Order files into BRDB tables

3.92.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule then follows TWO_MIN_WAIT.

3.92.2 Job BRDBC074_PLO_LOAD

3.92.2.1 Implementation

This job call to the executable BRDBC074 to load the Planned order files which are specified from BRDB_FILE_AUDIT_TRAIL (where process_name = 'BRDB_PLO_LOADER'' and file_status = 'N').

BRDBC074 populates the following tables:

- LFS_PLO_HEADER
- LFS_PLO_DETAILS

If the validation found errors then the Error Processor will be invoked to create a BRDB feed exception in BRDB_HOST_INTERFACE_FEED_EXCP. BRDB_FILE_AUDIT_TRAIL file_status for the XML file will be updated to 'E' for Error. An error file will be generated and error record will be written to the location specified in BRDB_OUTPUT_DIR_NAME and then moved to the location specified in OUTPUTSHARE_DIR_NAME

3.92.2.2 Rerun Action

Alert Operations on failure.

Resolve the root cause and rerun the job again.

3.93 Schedule BRDB_STOP_PLO

This schedule is run daily at 12.00.

3.93.1 Dependencies

This schedule depends on the BRDB_START_PLO schedule being started

FUJ00234958

3.93.2 Job BRDBX011_STOP_PLO_LOAD

3.93.3 Implementation

Invokes BRDBX011.sh to stop both the Planned Order BRDBC038 file daemon and BRDBC074 Planned order Loader daemon.

3.93.4 Rerun Action

Alert Operations on failure

3.94 Schedule BRDB_START_RDC

This schedule runs daily to process Replenishment Delivery Notice file from CWC application via PODG. It also generate error file if any.

3.94.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule.

3.94.2 Job BRDBC038_RDC_FROM_CWC

This job register Replenishment Delivery Notice file into BRDB.

3.94.2.1 Implementation

Invokes BRDBC038 to scan & process the input share directory (populated by PODG) for Replenishment Delivery Notice files (file name format DispatchedContainers *.xml).

The following is a list of directories used by this job: -

Note: The list is stored as values in table BRDB_EXT_INTERFACE_FEEDS for the row *"WHERE ext_interface_feed_name* = 'BRDB_RDC_LOADER'.

Description	Column Name	Value
Share directory	INPUTSHARE_DIR_NAME	/app/brdb/trans/input_share
BRDB input directory	BRDB_INPUT_DIR_NAME	/app/brdb/trans/externalinterface/externaltxns
BRDB output share directory	OUTPUTSHARE_DIR_NAME	/app/brdb/trans/externalinterface/output_share
BRDB load directory	BRDB_LOAD_DIR_NAME	/app/brdb/trans/externalinterface/loaddir
BRDB Audit Directory	AUDIT_DIR_NAME	/app/brdb/trans/audit/externalinterfaceaudit/externaltxns

3.94.2.2 Rerun Action

None

3.95 Schedule BRDB_RDC_LOAD

This schedule runs once and calls a daemons process which polls every 2 minutes from 8:05. The schedule loads all the registered Replenishment Delivery Notice files into BRDB tables

3.95.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule then follows TWO_MIN_WAIT.

3.95.2 Job BRDBC075_RDC_LOAD

3.95.2.1 Implementation

FUJ00234958

This job call to the executable BRDBC075 to load the Replenishment Delivery Notice files which are specified from BRDB_FILE_AUDIT_TRAIL (where process_name = 'BRDB_RDC_LOADER'' and file_status = 'N').

BRDBC075 populates the following tables:

- LFS_RDC_HEADER
- LFS_RDC_DETAILS

If the validation found errors then the Error Processor will be invoked to create a BRDB feed exception in BRDB_HOST_INTERFACE_FEED_EXCP. BRDB_FILE_AUDIT_TRAIL file_status for the XML file will be updated to 'E' for Error. An error file will be generated and error record will be written to the location specified in BRDB_OUTPUT_DIR_NAME and then moved to the location specified in OUTPUTSHARE_DIR_NAME

3.95.2.2 Rerun Action

Alert Operations on failure.

Resolve the root cause and rerun the job again.

3.96 Schedule BRDB_STOP_RDC

This schedule is run daily at 12.00.

3.96.1 Dependencies

This schedule depends on the BRDB_START_RDC schedule being started

3.96.2 Job BRDBX011_STOP_RDC_LOAD

3.96.3 Implementation

Invokes BRDBX011.sh to stop both the Replenishment Delivery Notice BRDBC038 file daemon and BRDBC075 Replenishment Delivery Notice Loader daemon.

3.96.4 Rerun Action

Alert Operations on failure

3.97 Schedule BRDB_PCL_TO_CWC

This schedule runs once and calls a daemons process which polls every PCL_SLEEP_INTERVAL from 20:00. The schedule generates Pouch Collection file for CWC.

3.97.1 Dependencies

This schedule depends on the completion of BRDB_SOD schedule.

3.97.2 Job BRDBC076_PCL_TO_CWC

3.97.2.1 Implementation

The process is a daemon process that generates a file and then sleeps for a period as defined in system parameter PCL_SLEEP_INTERVAL. If there are no Pouch Collection records pending then a file is not be created. Before sleeping, the process check the value of BRDB_PCL_STOP_YN and if the value is "Y" then the process will exit gracefully.

Files will get generated with the following name

POCccyymmddsss.xml

Where the red part of the filename is variable and consists of

FUJ00234958

ccyymmdd = Date passed to process via TWS

sss = Sequence number that starts at 1 each day

3.97.2.2 Rerun Action

Alert Operations on failure.

Resolve the root cause and rerun the job again.

3.98 Schedule BRDB_STOP_PCL

This schedule is run daily at 20.00.

3.98.1 Dependencies

This schedule depends on the BRDB_START_EUM schedule being started

3.98.2 Job BRDBX011_STOP_PCL

3.98.3 Implementation

Invokes BRDBX011.sh to stop both the EUM ForgeRock BRDBC038 file daemon and BRDBC066 EUM ForgeRock Loader daemon.

3.98.4 Rerun Action

Alert Operations on failure

3.99 Schedule BRDB_FROM_EMDB2

This schedule is run daily at 19:30 which will follow BRDB_SOB and the Estate Management schedule EST_BRDB_UPD (see EST HLD DES/APP/HLD/0118 for details). This schedule runs the Estate Management interface feed. It consists of a single task which will generally run on node 2; see section 3.2 above for details. Only the parent job BRDBX003_BRDATA_FROM_EMDB2 is included here however refer to LLD (DEV/APP/LLD/3850) for more details.

3.99.1 Dependencies

Schedule BRDB_FROM_EMDB2 depends on the completion of schedules BRDB_SOB and EST_BRDB_UPD.

3.99.2 Job BRDBX003_BRDATA_FROM_EMDB2

This job runs the Estate Management interface feed BRDB_EMDB2_INTERFACE.

3.99.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_EMDB2_INTERFACE.

This process references the following EMDB2 maintained tables:

Table Name	Description
EMDB2.EMDB2_POST_OFFICE	Maintained by EMDB2, contains information relevant to each individual PO branch (e.g. Retailer,suspended_distribution_flag ,etc).
EMDB2.EMDB2_BRANCH_NODES	Maintained by EMDB2, contains information relevant to each individual counter per branch - most relevantly the IP address associated with the counter.
EMDB2.EMDB2_USERS	Maintained by EMDB2, contains information relevant to each individual Branch user per branch .

FUJ00234958

EMDB2.EMDB2_STOCK_UNIT	Maintained by EMDB2, contains information relevant to each individual stock unit per branch and branch user .
EMDB2.EMDB2_ASSOCIATED_STOCK_UNIT	Maintained by EMDB2, contains information relevant to each individual node id per branch,branch user and stock unit .

The process updates the following BRDB tables.

Table Name	Description
OPS$BRBD.BRDB_BRANCH_INFO	Uses EMDB2_POST_OFFICE to set information such as the (Retailer,Branch status, etc).
OPS$BRDB.BRDB_BRANCH_NODE_INFO	Uses EMDB2_BRANCH_NODES to set information such as node id,ip_address,integrator etc.
OPS$BRDB.BRDB_FAD_HASH_OUTLET_MAPPING	New branches are inserted into this table, uses MOD(branch_code, 128) to resolve the FAD_HASH value.
OPS$BRDB.BRDB_TXN_CORR_TOOL_CTL	New branches are inserted into this table in order to allow SSC correction tools to maintain a running CURRENT_JSN value.
OPS$BRDB.BRDB_BRANCH_USERS	Uses EMDB2_USERS to set information such as branch user, stoch unit, etc. Channel Integration automatic users will be created for each node/device type of the form: $$ + device type + node ID
OPS$BRDB.BRDB_BRANCH_STOCK_UNITS	A default (DEF) stock unit is inserted for each new branch created and a stock unit of the node device type for NoAccounting branches (i.e. branch_status = 'NoAccounting').
OPS$BRDB.BRDB_STOCK_UNIT_ASSOCIATIONS	Uses EMDB2_ASSOCIATED_STOCK_UNIT to set information such as branch user,stock unit node id, etc.

3.99.2.2 Deadlock Scenario

There is a greater than zero chance that the interface's transaction may have contention with various concurrent Counter/BAL transactions if there is a bulk change in EMDB2 (as seen in PC0301765).

Available options for Operations:

i. Resubmit the schedule. The fewer/no Counter transactions then the interface should complete successfully.
ii. Cancel the BRDB_FROM_EMDB2 schedule, allow the Reference Data schedule to complete and then pick from one of the available sub-options
    a. Raise a call for Host to pick up the next morning for additional advice
    b. Subsequently create a one off invocation of BRDB_FROM_EMDB2 to run at around 4am (when it is less likely for Counter transactions to clash).
    c. Stop the BAL OSRs, allow BRDB_FROM_EMDB2 to complete and then finally re-enable the BAL OSRs.

3.99.2.3 Rerun Action

Alert Operations on failure.

If this schedule fails as a result of a deadlock (see McLaughlinD2617J) then consult the steps as described in Deadlock Scenario.

FUJ00234958

3.100 Schedule BRDB_BDC_TO_FRES

This schedule is run daily to generate Bureau daily Interface files

1.Transaction File BTD

2.Control Total File BCT

3.100.1 Dependencies

This schedule depends on BRDB_SOB .

3.100.2 Job BRDBX083_BDC_TO_FRES

These jobs (one per node) perform Bureau (BTD, BCT) interface file generation.

3.100.3 Implementation

The job is implemented as a call to the shell script BRDBX083.sh with the business day (YYYYMMDD) and this in turn invokes package procedure pkg_brdb_btd_to_frt.Generate_FRT_Interface_File to create Bureau interface files in the /app/brdb/trans/externalinterface/output directory.

The BTD and BCT Bureau interface files will be registered as new ('N') in the BDB table BRDB_FILE_AUDIT_TRAIL with a process_name of ' BRDB_BTD_TO_FRT' when first created. It will be updated to 'C' when all the data records have been successfully written to the file, and the total number of data records is written to the TOTAL_RECORDS column.

If File creation fails the error are is logged into BRDB_OPERATIONAL_EXCEPTIONS.

The pkg_brdb_btd_to_frt.Generate_FRT_Interface_File procedure utilises the existing 'process control' functionality – to store information on when the processes were run and whether they completed successfully etc. Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

The filename of the Bureau interface file is structured as below:

W_FILE_DAY_NOFILE_SEQUENCE.BTD

W_FILE_DAY_NOFILE_SEQUENCE.BCT

Where:

W_ = the fixed file prefix

FILE_DAY_N = SELECT LPAD(TO_CHAR(TO_DATE('${TRADING_DATE}', 'YYYYMMDD'),'DDD'),3,0) from dual

FILE_SEQUENCE = select lpad(parameter_number,3,0) from brdb_system_parameters where parameter_name = 'BRDB_FRT_FILE_SEQUENCE'

.BTD/.BCT = fixed file extension

3.100.3.1 Rerun Action

Alert Operations on failure

3.101 Schedule BRDB_TA_LOAD

This schedule runs once daily @06:05 and processes transaction acknowledgement files.

3.101.1 Dependencies

FUJ00234958

. This schedule depends on BRDB_SOB and the job BRDBC084_LOAD_TA depends on the completion of BRDBC038_TA_FROM_CREDENCE.

3.101.2 Job BRDBC084_LOAD_TA

3.101.2.1 Implementation

The job is implemented as a call to a binary BRDBC084 which checks for Transaction Acknowledgement files in the interface directory (identified by BRDB_Ext_Interface_Feeds.BRDB_Input_Dir_Name). The file names are assumed to be:

siccyymmddv999999999.tan

This will all be in lower case and is made up as follows :

si represents the system identifier (initially:

ca = Camelot,

pg = Post & Go,

ps = Paystation) but only “ca” now

ccyymmdd represents the calendar day that the TA file is first sent

v represents version of file – normally 1

999999999 represents the file number which is serially increasing for each run

Each TA file delivered on a given day needs a unique file number

.tan fixed value file suffix to indicate it is a Transaction Acknowledgment file for processing

It populated the table TPS_TXN_ACK_DETAILS with the file valid records and pupolated with BRDB_FILE_ERRORS with invalid records.

3.101.2.2 Rerun Action

Alert Operations on failure.

3.102 Schedule BRDB_AP_DIR_MAKE

This schedule runs once daily @06:05 and processes transaction acknowledgement files.

3.102.1 Dependencies

This schedule follows BRDB_SOB.

3.102.2 Job BRDBC085_BRDB_AP_DIR_MAKE

3.102.2.1 Implementation

The job is implemented as a call to a binary BRDBC085 which creates directories where they do not already exist in location defined in brdb_aps_delivery_agreements

3.102.2.2 Rerun Action

Alert Operations on failure.

3.103 Schedule BRDB_AP_VALIDATE

This schedule is run daily to do validation of APS transactions

3.103.1 Dependencies

FUJ00234958

Follows: BRDB_EXT_APS_CPY

3.103.2 Job BRDBX086_BRDB_AP_VALIDATE

The job performs APS transaction validation.

3.103.3 Implementation

The job is implemented as a call to the shell script BRDBX086.sh with the business day (YYYYMMDD) and this in turn invokes package procedure PKG_BRDB_APS_VALIDATION.do_validation to perform

validation for brdb_aps_mc_txns records for given business day.

The exception records are copied into brdb_aps_mc_txns_e.

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.103.3.1 Rerun Action

Alert Operations on failure

3.104 Schedule BRDB_CTS_FILE

This schedule is run daily to do create Client Transmission Summary

3.104.1 Dependencies

Follows: BRDB_AP_FILES

3.104.2 Job BRDBX087_BRDB_CTS_FILE

This job calls the Client TransmissionSummary Package.

3.104.3 Implementation

The job is implemented as a call to the shell script BRDBX087.sh with the business day (YYYYMMDD) and this in turn invokes package procedure pkg_brdb_APS_CTS.generate_aps_cts_files to create Client Transmission Summary for a given business day.

Files are stored in this location : /app/brdb/trans/externalinterface/output_share/APS/CTS

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.104.3.1 Rerun Action

Alert Operations on failure

3.105 Schedule BRDB_AP_RECON

This schedule is run daily to calculate the reconciliation totals and deliver the reconciliation reports.

3.105.1 Dependencies

FUJ00234958

Follows: BRDB_AP_TRNSFM

3.105.2 Job BRDBX089_BRDB_AP_RECON

This job creates reconciliation report in format: aps_reconciliation"${PROCESSING_DATE}.

3.105.3 Implementation

The job is implemented as a call to the shell script BRDBX089.sh with the business day (YYYYMMDD) -> BRDBX089.sh <trading/business date> to create reconciliation reports for a given business day.

Files are stored in this location : /app/brdb/trans/externalinterface/output_share/APS/CTS

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.105.3.1 Rerun Action

Alert Operations on failure

3.106 Schedule BRDB_AP_TRNSFM

This schedule is to perform the transformation of AP Client File from Type X to Type G.

3.106.1 Dependencies

Follows: BRDB_AP_FILES

3.106.2 Job BRDBX088_BRDB_AP_TRNSFM

This job transmits the AP Client File from Type X to Type G.

3.106.3 Implementation

The job is implemented as a call to the shell script BRDBX088.sh as BRDBX088.sh S PODG_GIRO N transcash.awk or BRDBX088.sh S PODG_GIRO Y transcash.awk. The parameters are as follows.

File Prefix, Destination Directory, Archive Flag, Awk script

Files are stored in this location ${BRDB_APS_ROOT}${BRDB_APS_FILES}/TRANSFORM/output/

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.106.3.1 Rerun Action

Alert Operations on failure

3.107 Schedule BRDB_AP_RECON

This schedule calculates the reconciled totals and deliver the reports

3.107.1 Dependencies

Follows: BRDB_AP_TRNSFM

FUJ00234958

3.107.2 Job BRDBX089_BRDB_AP_RECON

This job calculates the reconciliation totals and delivers the report.

3.107.3 Implementation

The job is implemented as a call to the shell script BRDBX089.sh with the business day (YYYYMMDD) and this in turn calculates the reconciliation totals for the given business day and the delivers the reconciliation report. The output for this script will be either 0 or 1 depends on the exceptions.

Files are stored in this location : /app/brdb/trans/externalinterface/output_share/APS/CTS

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.107.3.1 Rerun Action

Alert Operations on failure

3.108 Schedule BRDB_AP_RECON

This schedule generates the quarantine report.

3.108.1 Dependencies

Follows: NA

3.108.2 Job BRDBX090_BRDB_AP_QUARANTINE

This job generates the APS Quarantine report.

3.108.3 Implementation

The job is implemented as a call to the shell script BRDBX090.sh with the business day (YYYYMMDD) and this will generate the Quarantine report for the given business day.

Files are stored in this location : /app/brdb/trans/externalinterface/output_share/APS/CTS

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.108.3.1 Rerun Action

Alert Operations on failure

3.109 Schedule BRDB_AP_FILES

This schedule creates the BRDB APS file. the schedule runs 7 jobs in parallel but limited by a single resource "BRAPFILES" so that only one job runs at a time

3.109.1 Dependencies

Follows: BRDB_AP_VALIDATE, BRDB_AP_DIR_MAKE

FUJ00234958

3.109.2 Job BRDBX091_APS_TYPE_<Type G/X/XO/XU/XP/BT>_FILES

This job generates the specific type of BRDB APS file requested

3.109.3 Implementation

The job is implemented as a call to the shell script BRDBX091.sh with the business day (YYYYMMDD) with the parameter for the type of files that needs to be generated (G/X/XO/XU/XP/BP) and this will in turn calls pkg_brdb_aps_file.create_files and generates the specific type of APS files for the given business day.

e.g. BRDBX091.sh G 20190723

Child Location for each client account is defined in rdds_aps_delivery agreement(child directory).File location = Root location + Child directory.

Root location : /app/brdb/trans/externalinterface/output_share/APS/

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.109.3.1 Rerun Action

Alert Operations on failure

3.110 Schedule BRDB_EXT_APS_CPY

This schedule copies external APS transaction to new APS table.

3.110.1 Dependencies

Follows: BRDB_TXN_POST

3.110.2 Job BRDBX003_EXT_APS_COPY _1…4

These jobs(one per node) copy external APS transaction to new APS table

3.110.3 Implementation

The job is implemented as a call to the shell script BRDBX003.sh with the business day (YYYYMMDD) and and node id which in turn calls PKG_BRDB_EXT_APS_COPY to perform the data copy.

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.110.3.1 Rerun Action

Alert Operations on failure

3.111 Schedule BRDB_NRT_AP_CPY1..3

This schedule starts @ 8:05 and runs through the day. The daemon job is started and paused three times a day using BRDB_START_FEED1..3 and BRDB_PAUSE_FEED1..3

3.111.1 Dependencies

FUJ00234958

Follows: BRDB_START_FEED1..3

3.111.2 Job BRDBX003_APS_NRT_PROCESS

These jobs(one per node) copy APS transaction to new APS table BRDB_APS_MC_TXNS

3.111.1 Implementation

The job is implemented as a call to the shell script BRDBX003.sh with the business day (YYYYMMDD) and and node id which in turn calls PKG_BRDB_APS_NRT_PROCESS.load_data to perform the data copy.

It's a daemon process which keeps running though out the day.

Table BRDB_PROCESS_CONTROL or BRDB_PROCESS_AUDIT can be queried for this information. The table BRDB_PROCESS_CONTROL is used to enforce requirements such as ensuring that the job can only be run once for a given business day.

3.111.1.1 Rerun Action

3.112 Schedule BRDB_PDL_TO_PBI

This schedule runs once daily @09:00 Pouch Deliveries file Daemon.

3.112.1 Dependencies

This schedule follows BRDB_SOB.

3.112.2 Job BRDBC078_PDL_TO_PBI

3.112.2.1 Implementation

The job is implemented as a call to a binary BRDBC078 which does the following :

A Pouch Delivery file will be delivered to Power BI on a regular basis, 0 to 24 files will be generated daily with no empty files.

The process will be a daemon process that generates a file and then sleeps for a period as defined in system parameter PDLBI_SLEEP_INTERVAL.

If there are no Pouch Deliveries records pending then a file will not be created. Before sleeping, the process will check the value of BRDB_PDLBI_STOP_YN and if the value is "Y" then the process will exit gracefully.

The filename takes the name:

PODCCYYMMDDSSS.TXT

Where

ccyymmdd = Date passed to process via TWS

sss = Sequence number that starts at 1 each day

If the daemon process needs to be restarted for any reason then the sequence number needs to pick-up from where it was.

The sequence number will therefore be derived from the table BRDB_FILE_AUDIT_TRAIL by looking for files that have already been generated today and then selecting the next sequence number.

FUJ00234958

Files will initially be written to /app/brdb/trans/externalinterface/output and will be moved to /app/brdb/trans/externalinterface/output_share to make them available to PODG.

If there is no pending data then no file will be produced.

An example file would look as follows:

DELD|003210|310212345678|GBP|12345|07-12-2018 09:25:24

DELT|1|12345

3.112.2.2 Rerun Action

Alert Operations on failure.

3.113 Schedule BRDB_CSH_TO_PBI

This schedule runs once daily to generate Cash on Hand file which are delivered to Power BI

3.113.1 Dependencies

This schedule follows BRDB_ONCH_AGG.

3.113.2 Job BRDBX080_COH_FILE_TO_PBI

3.113.2.1 Implementation

The job is implemented as a call to a shell BRDBX080.sh to to generate Cash on Hand file which are delivered to Power BI

File name should be according to input TWS date with below name convention.

COH + TWS DATE(YYYYMMDD) + 001(Sequence No Always 001) + .txt = COHYYYYMMDD001.txt

3.113.2.2 Rerun Action

Alert Operations on failure.

3.114 Schedule BRDB_PCL_TO_PBI

This schedule runs once daily @09:00 to create a Pouch Collection file which is delivered to Power BI.

3.114.1 Dependencies

This schedule follows BRDB_SOB.

3.114.2 Job BRDBC079_PCL_TO_PBI

3.114.2.1 Implementation

The job is implemented as a call to a binary BRDBC079 which does the following.

A Pouch Collection file are delivered to Power BI on a regular basis with

0 to 24 files will be generate daily with no empty files.

The delivery of files will be each half an hour between the times 08:00 and 19:00.

The script will start on 09:00 and check if any record are available to generate the

files by joining with below 3 tables.

1)BRDB_POUCH_COLL_DETAILS

FUJ00234958

2)RDDS_POUCH_TYPES

3)BRDB_POUCH_COLL_HEADER => PBI_DELIVERED_TIMESTAMP IS NULL

If there is no record to generate the file script will sleep for 1800 seconds for the next process.

If there is any record, the script will create a file with name POCYYYYMMDD001.txt (YYYYMMDD - TWS Date) (Last3 characters are file sequence number), then write the output content of query into the file.

The file sequence number will be taken from BRDB_FILE_AUDIT_TRAIL table as per the input TWS parameter.

If Script is running first time for that TWS date and there is any record to generate the file, file sequence will be 1,for the next cycle again any record to generate the file with same TWS date script will take the next number as 2.

Insert a record into the BRDB_FILE_AUDIT_TRAIL table for that file name with input TWS date.

Update the PBI_DELIVERED_TIMESTAMP column NULL to SYSDATE in BRDB_POUCH_COLL_HEADER table.

File will be Share in Directory: /app/brdb/trans/externalinterface/output_share

Before Checking the availability of record script will check the BRDB_PCLBI_STOP_YN flag in BRDB_SYSTEM_PARAMETERS table.

If BRDB_PCLBI_STOP_YN is 'N' => Continuation Of Script

If BRDB_PCLBI_STOP_YN is 'Y' => Successful Completion Of The Script

BRDBX011.sh -n BRDB_PCLBI_STOP_YN -t T -v Y Need to be execute to successful

On completion of the script.(Flag BRDB_PCLBI_STOP_YN will be update to 'Y')

3.114.2.2 Rerun Action

Alert Operations on failure.

3.115 Schedule BRDB_PBS_UNDO

This schedule runs daily @18:00.

3.115.1 Dependencies

This schedule follows BRDB_SOD.

3.115.2 Job BRDBC092_PBS_UNDO

FUJ00234958

3.115.2.1 Implementation

The job is implemented as a call to a binary BRDBC092 which does the following :

To produce one or more Semi-colon separated files for undo payments transactions.

Maximum 99999 (PBS_UNDO_MAX_RECORDS) records must be write in a single file.If more record present script will create a new file with next serial number.

If the process needs to be restarted/rerun for any reason then the sequence number needs to pick-up from where it was.The sequence number will therefore be derived from the table BRDB_FILE_AUDIT_TRAIL by looking for files that have already been generated today and then selecting the next sequence number.

Files will initially be written to /app/brdb/trans/dcmshare/output/temp path and will be moved to /app/brdb/trans/dcmshare/output path.

If there is no record in table BRDB_RX_UNDO_TRANSACTIONS with this criteria (CLIENT_NAME = 'Ingenico' & CLIENT_ROUTING_NAME = 'DCS' & PROCESSED_YN = 'N') then no file will be produced.

The Filename Format:

batch_pay_<PBS_UNDO_CUSTOMER_ID>_<YYYYMMDD>_nnnnnn

where:

<PBS_UNDO_CUSTOMER_ID>, is the Fujitsu/Post Office customer identifier within the Ingenico.

Integration 10532

Pre-Prod 11991

Prod 12333

<YYYYMMDD>, is the system date(sysdate).

nnnnnn, is the file number, left padded with zeros.Starts at 000001 for each system day for the 1st <PBS_UNDO_MAX_RECORDS> records and then moves to 000002 for the next <PBS_UNDO_MAX_RECORDS> records.

E.g. batch_pay_12333_20201028_000001

An example file would look as follows:

1.0;PAY;1;2020-10-28T11:09:00;000001;12333

500;826;VCRDP;1001167100000001924;Test01;

3.115.2.2 Rerun Action

Alert Operations on failure.

3.116 Schedule BRDB_PBS_TO_DRS2

This schedule is run daily to to copy payment and banking transaction data to DRS2 RDS instance in AWS.

3.116.1 Dependencies

This schedule follows BRDB_SOB.

3.116.2 Job BRDBX003_SSK_TXN_TO_DRS2

3.116.2.1 Implementation

FUJ00234958

The job is implemented as a call to the shell script BRDBX003.sh with the business day (YYYYMMDD) and and node id which in turn calls PKG_BRDB_SSK_TXN_TO_DRS2.load_data to copy payment and banking transaction data to DRS2 RDS instance in AWS.

The source table is OPS$BRDB.BRDB_RX_DCS_TRANSACTIONS and the target table DRS_C12_STAGING which is present at RDS instance in AWS.All the records copy from source table to target table with column name DRS_C12_STAGING.DATA_SOURCE='DCS'.

The below query used to select the records from source table.

```
SELECT
        JOURNAL_DATE,BRANCH_ACCOUNTING_CODE,AMOUNT,NODE_ID,PROD_ID,TRADING_DATE,
        TRANSACTION_START_DATE,RESPONSE_CODE,ROUTING_GATEWAY,SCHEME_ID,
        TRANSACTION_RESULT_CODE,TRANSACTION_TYPE,RECEIPT_DATE,AUTHORISATION_CODE,
        PAN,HORIZON_TRANSACTION_ID,REVERSAL,FAD_HASH,'DCS'
FROM    OPS$BRDB.BRDB_RX_DCS_TRANSACTIONS
WHERE  TRADING_DATE = p_process_date
AND     JOURNAL_DATE BETWEEN p_process_date - 1 AND p_process_date + 2
AND     DATA_SOURCE != 'HX'
AND     RESPONSE_CODE NOT IN (SELECT RESPONSE_CODE  FROM BRDB_NON_FI_RESP_CODES)
       ORDER BY FAD_HASH;
```

3.116.2.2 Rerun Action

Alert Operations on failure.

3.116.3 Job BRDBX003_ETU_TXN_TO_DRS2

3.116.3.1 Implementation

The job is implemented as a call to the shell script BRDBX003.sh with the business day (YYYYMMDD) and and node id which in turn calls PKG_BRDB_ETU_TXN_TO_DRS2.load_data to copy payment and banking transaction data to DRS2 RDS instance in AWS.

The source table is OPS$BRDB.BRDB_RX_NWB_TRANSACTIONS and the target table DRS_C12_STAGING which is present at RDS instance in AWS.All the records copy from source table to target table with column name DRS_C12_STAGING.DATA_SOURCE=' NWB'.

The below query used to select the records from source table.

```
SELECT
        JOURNAL_DATE,BRANCH_ACCOUNTING_CODE,AMOUNT,NODE_ID,PROD_ID,TRADING_DATE,
        TRANSACTION_START_DATE,RESPONSE_CODE,ROUTING_GATEWAY,SCHEME_ID,
        TRANSACTION_RESULT_CODE,TRANSACTION_TYPE,RECEIPT_DATE,AUTHORISATION_CODE,
        PAN,HORIZON_TRANSACTION_ID,REVERSAL,FAD_HASH,'NWB'
FROM    OPS$BRDB.BRDB_RX_NWB_TRANSACTIONS
WHERE  TRADING_DATE      = p_process_date
AND     JOURNAL_DATE BETWEEN p_process_date - 1 AND p_process_date + 2
AND     APPLICATION_TYPE = 'ETA'
```

FUJ00234958


```
AND     RESPONSE_CODE NOT IN (SELECT RESPONSE_CODE FROM BRDB_NON_FI_RESP_CODES)
        ORDER BY FAD_HASH;
```

3.116.3.2 Rerun Action

Alert Operations on failure.

3.116.4 Job BRDBX003_PBS_TXN_TO_DRS2

3.116.4.1 Implementation

The job is implemented as a call to the shell script BRDBX003.sh with the business day (YYYYMMDD) and and node id which in turn calls PKG_BRDB_PBS_TXN_TO_DRS2.load_data to copy payment and banking transaction data to DRS2 RDS instance in AWS.

The source table is OPS$BRDB.BRDB_RX_PBS_TRANSACTIONS and the target table DRS_C12_STAGING which is present at RDS instance in AWS.All the records copy from source table to target table with column name DRS_C12_STAGING.DATA_SOURCE='PBS'.

The below query used to select the records from source table.

```
SELECT
        rnt.JOURNAL_DATE,rnt.BRANCH_ACCOUNTING_CODE,rnt.AMOUNT, rnt.NODE_ID, rnt.PROD_ID,
        rnt.TRADING_DATE,rnt.TRANSACTION_START_DATE,rnt.RESPONSE_CODE,rp.ROUTING_GATEWAY,
        rnt.TRANSACTION_RESULT_CODE,rnt.TRANSACTION_TYPE,rnt.RECEIPT_DATE ,
        rnt.PAN,rnt.PBS_TRANSACTION_REFERENCE,rnt.REVERSAL,rnt.FAD_HASH,'PBS'
FROM    OPS$BRDB.BRDB_RX_PBS_TRANSACTIONS rnt,OPS$BRDB.RDDS_PRODUCTS rp
WHERE   rnt.TRADING_DATE = p_process_date
AND     rnt.JOURNAL_DATE BETWEEN p_process_date - 1 AND p_process_date + 2
AND     rnt.PROD_ID      = rp.PROD_ID
AND     rnt.RESPONSE_CODE NOT IN (SELECT RESPONSE_CODE FROM BRDB_NON_FI_RESP_CODES)
        ORDER BY rnt.FAD_HASH;
```

3.116.4.2 Rerun Action

Alert Operations on failure.

4 Backup and Recovery

The Branch Database and Branch Support Database are both backed up using Oracle RMAN. The frequency of the backups, the type of backup, the backup location and retention periods are detailed in the Branch Database High Level Design (See Section 0.4). Note that RMAN backups of BRDB are actually made on SBRDB (the active standby).

4.1 BRDB & BRSS Backups

4.1.1 Backup Duration

The Oracle RMAN backups, when run, tend to do so for different durations. The factors that will affect run-time could be: -

- Activity on the node executing the backup, e.g. CPU, disk, etc.
- The type of backup being run, e.g. a full backup (incremental level 0) or an incremental level 1 backup.

FUJ00234958

- The amount of archivelogs generated since the last backup (relevant to any backup level).

It is therefore important that when backups are not run for whatever reason, that they are re-scheduled to run as soon as possible.

4.1.1.1 RMAN & Goldengate

RMAN, by default, is configured to remove any archivelogs after a successful backup. Goldengate has a direct impact on whether or not RMAN is able to remove an archivelog or not. This criterion is determined by whether the archivelog is or is not needed by the OGG Extract process.

If OGG does require the archivelog, RMAN is not "allowed" to remove it and the archivelog will remain in +BRDB_FLASH/arch. An RMAN-08137 message will be reported when this is the case. It is a warning message and not a failure.

Any subsequent backups will skip each archivelog as each one already has a successful copy in a previous backup. When attempting to drop the archivelog again, the same check is made and if OGG no longer needs the archivelog, it will be released for deletion by RMAN.

4.2 Restoring files with RMAN

DBAs in Ireland have standard support procedures for dealing with restores and recovery after differing failures, e.g. restoring SPFiles, controlfiles, archivelogs, datafiles, et cetera. These scripts and procedures will be used by the DBA Support Team in a recovery scenario in conjunction with this guide and support from technical leads and possibly vendor specialists, e.g. EMC, Oracle, et cetera.

WARNING: As with any activity relating to the physical dimension of restoring activities, keeping the high importance of these types of activities at the back of one's mind is of paramount significance! Restoring datafiles or redologs using RMAN, for instance, could cause the crash of the entire Branch Database if performed in a non-disaster scenario and without the proper authorisation!

FUJ00234958

4.3 Failure and Recovery

Failures should be detected by SMC and then escalated to the UNIX/DBA teams who, in turn where appropriate, will escalate to CS, SSC and Development.

Recovery actions will be performed by the UNIX/DBA teams with the agreement of CS, SSC and Development.

Business escalation should be handled by SMC.

4.3.1 Escalation and Notification

NB: In the event of a failure and subsequent recovery, the relevant Post Office Disaster Recovery escalation procedures need to be followed in conjunction with the relevant Business Continuity personnel and Fujitsu Support Teams.

The Business Continuity function along with the relevant management team(s) will have to consider the facts, weigh up the current threats and decide whether to authorise the failover to Standby or not.

In general, the hierarchy in which support teams are contacted is as follows: -

- SMC will typically coordinate all types of failures and will also be the first point of contact in most types of problems, application, networks, etc.; Responsible for monitoring Tivoli.
- SSC is responsible for supporting the application. DBA, UNIX and Network Support Teams are also responsible for support at this level
- Finally, the development teams would support all other teams in their respective areas of expertise.

4.3.2 Media Failure and Recovery

4.3.2.1 A Corrupt or Damaged Redolog Group

If an online redolog group has all of it's members damaged - regardless of how this came to be - the recovery solution will change depending on the 'state' of the online redolog group.

4.3.2.1.1 Scenario and Recovery Solution

Scenario: This failure scenario involves having all redologs of a particular redo log group, corrupted or damaged.

Solution: *Redolog Group is INACTIVE*

This redolog group will ***not*** be required for crash recovery.

Action → Clear the logfile group.

Redolog Group is ACTIVE

This redolog group ***is*** required for crash recovery.

Action → (i.) Issue a checkpoint and (ii.) clear the damaged redolog(s). If performing (i.) and (ii.) prove unsuccessful, then the database must be restored and recovered (incomplete recovery) to a point-in-time before the redolog(s) were damaged (to the most recent *available* group prior to damage).

Redolog Group is CURRENT

This redolog group ***is*** required for crash recovery.

Action → Clear the damaged redolog(s) (do not attempt a checkpoint). If performing (i.) is unsuccessful, then the database must be restored and recovered (incomplete

FUJ00234958

recovery) to a point-in-time before the redolog(s) were damaged (to the most recent *available* group prior to damage).

Note:

- Depending on our SLA with the customer (in terms of time-to-recover), it may be more advantageous to either complete the restore and recovery or if the corruption is localised, i.e. only present on the hardware of the current site (e.g. IRE11), then failing the Data Centre over (e.g. to IRE19) may be a faster (less troublesome) route to take.
- The Database failover from PRIMARY to STANDBY is not recommended in this scenario.

4.3.3 Instance/Node Failure and Recovery

4.3.3.1 Working Assumptions

The guidance in the following sections assumes that every effort to resolve a failure – be that failure due to software, hardware, network or failures of greater magnitude – has been taken. For hardware failures this can include checking Oracle CRS logs or Linux system logs and in the case of database instance failures, alert_BRDB[1|2|3|4].log, trace files, application and process log files, CRS logs, dump files and Grid Control alert messages. This is by no means an exhaustive list.

The recovery of an Oracle Database instance is essentially automatic as Oracle provides internal mechanisms which perform instance recovery on startup.

The recovery of a pBlade within the BRDB BladeFrame is similarly automatic, in that the BladeFrame will attempt to bring the failed pBlade back online; but if unsuccessful, a replacement of the pBlade with an operational "spare", while not automatic is fairly trouble-free

Oracle Cluster Ready Services (CRS), in normal operation will automatically restart any database instance on a node that is being restarted (for whatever reason). This will always include the grid control agent(s), the Oracle listener and the local ASM instance. However, the starting of the database instance – which is dependant on the ASM instance having started – will be ***disabled*** for all Branch Database Cluster Ready Services. That is, upon restart, all components required by the database instance will be restarted except for the instance itself.

What is important to note, is that within BRDB, database instances are closely coupled with the application (in that each branch resides in a specific FAD HASH, each FAD_HASH is accessed from a specific node when that node is available) . Therefore when an instance or node fails, its recovery will always represent a two-fold process, logically within the application and the actual node/instance itself.

4.3.3.2 Single BRDB Instance Crash

The instance will automatically be removed from **`BRDB_OPERATIONAL_INSTANCES`** by BRDBX010 which is invoked by the Fast Application Notification (FAN) mechanism at the time of the instance failure. Note that BRDBX010 is only executed by the FAN event and not by any other means.

The failed instance will need to be started manually via Grid Control or SQL*Plus. Starting the instance is an activity that needs to be thought through. The reason for this is that once the failed instance has been started manually, the cluster will once again show the full complement of instances and the listener can begin accepting connections for that instance. However the 'logical' view represented in **`BRDB_OPERATIONAL_INSTANCES`** will show that the instance in question is *not* available for requests from the Branch Access Layer (BAL). At this point, therefore, the physical database instance has been started, but the application will not be aware of that fact. This is done by stopping and starting, in a sequential manner, each Online Service Router (OSR) in turn (of which there are 20).

Please note: The instructions that follow, detail the updating of **`BRDB_OPERATIONAL_INSTANCES`** using BRDBX013 or by a manual update. It is particularly important to note that this should be done *prior* to "making the application aware", i.e. stopping and starting each OSR to reflect the change.

At the end of the online-day (after 18:00 and preferably before the overnight schedules start, but not essential), ***the recommended approach*** is to make the instances logically available, manually. This is

FUJ00234958

done by either executing BRDBX013 (BRDBX013 will check the state of each instance, whether up or down, and update **BRDB_OPERATIONAL_INSTANCES** accordingly) *or* by following the instructions in the table (Table 2) below. This is especially relevant if one wants granular control of what is represented in that table, as BRDBX013 will update all rows if necessary in order to ensure that the table represents the *actual* state of the cluster and this may not be required in every case.

BRDBX013 is executed as follows: -

```
$> cd /app_sw/brdb/sh
$> BRDBX013.sh
```

Finally, at the end of the Business day, the "End Of Day" process, namely BRDBC009, will check that all available instances are logically and correctly represented in **BRDB_OPERATIONAL_INSTANCES** and if not, will update the table to reflect the correct real-world representation. Having BRDBC009 perform this task is not necessarily the best course of action as the BAL needs to be made aware that the instance mapping has changed (this is done as detailed above). Therefore, BRDBC009 should be seen as a backup action rather than the preferred.

If, for whatever reason, the failed instance, once started and open, needs to be made available to the BAL and before the end of the day, then the following must be followed. Using meaningful and accurate values for the following values, e.g.: -

<FAN Event String>: **Manual recovery by <user's job title> <user's name> for fast recovery of instance due to unexpected node failure. Authorisation given by <authorisor's job title> <authorisor's name>.**

<Host Name>: **IRRELEVANT** (obtain by typing **hostname** or **uname -n** on the relevant node).

Step	Description	Server Execution
Assumptions	i. User is logged onto any node of the BRDB cluster as the ***brdb*** user. ii. It is ***imperative*** that there are no schedule related processes running when this manual operation is performed. There are many schedule related jobs which are fad-hash/branch code dependant and if these mappings are changed mid-schedule, significant problems could occur!	
1.	Logon to SQL*Plus command-line interface as OPS$BRDB, but first set the correct Oracle SID. This will connect you to the BRDB database. Double-check that you are on the right instance, noting in particular the values for *instance_name, host_name and status*.	**$> . oraenv** [now type in BRDB1 (assuming you're on node 1)] **$> sqlplus /** **SQL> SELECT * FROM v$instance;**

2.	Execute this DML to re-instate the availability of the instance in question.	`UPDATE brdb_operational_instances`
		`  SET is_available = 'Y',`
		`  fan_event = SUBSTR('<FAN Event String>', 1, 1000),`
		`  update_timestamp = SYSTIMESTAMP`
		`WHERE UPPER(host_name) = UPPER(SUBSTR('<Host Name>', -7));`

FUJ00234958

Step	Description	Server Execution
	Commit your change	`COMMIT;`

Table 2: BRDB_OPERATIONAL_INSTANCES Update Instructions

4.3.3.3 Single BRDB Node Crash and Restart

Failure notification will occur via the ITM Tivoli agent and will also be visible via Grid Control in terms of instance availability notification. FAN will update logical instance availability upon failure.

PAN Manager (BladeFrame operational software) will attempt to automatically restart the failed pServer. Once the pServer is initialised, the node has started, and with it the listener and ASM. The instance must be manually started.

See section 4.3.3.2 for more on re-instating logical instance availability.

4.3.3.4 Single BRDB Instance Crash - Fails to Start

See section 4.3.3.8.

4.3.3.5 Single BRDB Node Crash - Fails to Restart

Failure notification will occur via the ITM Tivoli agent and will also be visible via Grid Control in terms of instance availability notification. FAN will update logical instance unavailability upon failure.

If the BladeFrame cannot automatically restart the failed pServer, the PAN manager will flag an error. An attempt will be made at restarting the pServer on the spare pBlade. If unsuccessful, Support will then need to follow it up and resolve accordingly. Either solving the problem or replacing the pBlade and attempting another restart.

The BAL will not have “use” of the now unavailable instance until such time as the node's failure has been resolved and the instance is made available on the new/repaired node, by Support. As well as the instance being logically made available by either the EOD process (BRDBC009) or through manual intervention (described in section 4.3.3.2). BRDBC009 will continue to report in BRDB_OPERATIONAL_EXCEPTIONS, that the instance is unavailable.

4.3.3.6 Two or More BRDB Instances Crash

As mentioned in section 4.3.3.2, the BAL will not have “use” of the now unavailable instances until such time as each instance is available and either the EOD process (BRDBC009) has run or through manual intervention.

Each failed instance will need to be started manually via Grid Control or SQL*Plus.

If the instances restart successfully, then Support must make the instances “logically” available by the manual process specified in section 4.3.3.2, for ***each instance***.

Depending on the consensus of Support personnel, making “logically” available the newly started instances can be done at this point. The reason for either making the instances available or not is simply to do with the load on the remaining nodes and whether it is perceived that they are able to cope.

If, however, the instances are unable to restart or do restart but have further problems presenting themselves, e.g. they aren't accepting requests, there are network issues, loss of ASM diskgroups, et cetera, then the instances should be treated as ***non-restartable*** and the relevant escalation process should be followed (see Section 4.3.1).

4.3.3.7 Two or More BRDB Nodes Crash and Restart

Failure notification will occur via the ITM Tivoli agent and will also be visible via Grid Control in terms of instance availability notification. FAN will update logical instance unavailability upon failure.

The BladeFrame will attempt to automatically restart the failed pServers (on related pBlades) as defined by the LPAN configuration. Once the blades are initialised and the nodes have restarted, normal

FUJ00234958

behaviour would dictate that the related database instances are started again automatically. As with the scenario presented in section 4.3.3.3, the instances must be manually started and then made available to the BAL as the cluster will not bring them up automatically.

Depending on the consensus of Support personnel, making "logically" available the newly started instances can be done at this point. The reason for either making the instances available or not is simply to do with the load on the remaining nodes and whether it is perceived that they are able to cope.

See section 4.3.3.2 for more on re-instating logical instance availability. This applies for every instance.

4.3.3.8 Two or More BRDB Instances Crash – Fail to Restart

It must be assumed that every effort has been employed in restarting the instance(s) within the agreed SLA. If this two-or-more-instance-failure persists, then the following logic in determining an outcome should apply.

Has the problem occurred *outside core* business hours?

If ***yes***, and there are at least two RAC instance(s) in full operation, then there may be sufficient throughput available for the effective servicing of reduced business traffic. In such cases, it is often more beneficial to continue to use BRDB (the primary database), rather than initiate the failover procedure (see Section 0) which details the failing over of all users to SBRDB (the standby database) as this involves a coordinated, multi-team effort (for escalation see Section 4.3.1). In addition it will also allow more time for the resolution of the main reason for failure, be it software or hardware related.

If ***no*** or there are more than two instance failures, then the very real possibility that severe degradation in transaction throughput will present itself. At this point then the instances should be treated as ***non-restartable*** and the relevant escalation process should be followed (see Section 4.3.1).

4.3.3.9 Two or More BRDB Nodes Crash – Fail to Restart

Similar in resolution to section 4.3.3.8

It must be assumed that every effort has been employed in restarting the failed pBlades and have them correctly integrated into the cluster within the agreed SLA. If this two-or-more-node-failure persists, then the following logic in determining an outcome should apply.

Has the problem occurred *outside core* business hours?

If ***yes***, and there are at least two nodes of the RAC cluster still in full operation, then there may be sufficient throughput available for the reduced business traffic. In such cases, it is often more beneficial to continue to use the BRDB (primary database) cluster, rather than initiate the failover procedure (See Appendix A) which details the failing over of all users to the SBRDB (standby database) cluster as this involves a coordinated, multi-team effort. In addition it will also allow the resolution of the main reason for failure, be it hardware related or not.

If ***no*** or there is only a single node available, then the very real possibility that severe degradation in transaction throughput will present itself. The Business Continuity function along with the relevant management team will have to consider the facts, weigh up the current threats and decide whether to authorise the failover to the Standby cluster or not.

See section 4.3.1 for the service team/support team contact and escalation hierarchy.

Complete failover could be manually initiated and if so will need to follow the steps outlined in Section 6.

FUJ00234958

5 General and Troubleshooting Notes

5.1 Database

5.1.1 Oracle Database Listeners

The database listeners on all branch database nodes have been set up in the following way. This section provides a short explanation of how they are set up, how to interact with them and the expected status outputs.

The listeners are configured as follows: -

- The name of the listener will be of the form ***LISTENER which is controlled by ASM instance using grid user***, e.g. ***LISTENER***
- The port the listener has been configured to use is ***1529***
- ***Each database instance has a local listener configured with a local listener named LISTENER_{NODENAME}***
- The node (and in turn the IP) the listener has been configured to accept connections via the the VIP IRRELEVANT, e.g. for BDB node 1 the node name is IRRELEVANT IRRELEVANT

In terms of Oracle Net and it's configuration files, there should always be one of each on every node, namely **`sqlnet.ora`**, **`tnsnames.ora`**.(found in **`$ORACLE_HOME/network/admin`**). **`The listener.ora is configured in the GRID_HOME directory for ASM.`**(found in **`$GRID/network/admin`**)

5.1.1.1 Oracle Net Config. Files

The files have been formerly delivered during the installation of Oracle Software binaries, configuration of the ASM and database instances. and won't be need to be changed unless there is a specific problem. The following, shows a few excerpts of what the files could look like as of October 2009 (note that these values are not representative of those in the LIVE environment and are merely for reference): -

FUJ00234958

sqlnet.ora

SQLNET.INBOUND_CONNECT_TIMEOUT=15 performs the same function and behaves in the same way as the parameter configured for the listener, only waits longer.

SQLNET.EXPIRE_TIME=5 determines the number of ***minutes*** that Oracle will allow connections which are not in use, to exist, before terminating the process. This normally applies to connections which have abnormally ended.

BEQUEATH_DETACH=TRUE Stops Pro*C executables returning -1 regardless of failure or success.

```
#sqlnet.ora.lprpbdb201 Network Configuration File: /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/sqlnet.ora IRRELEVANT
# Generated by Oracle configuration tools.
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
# +--------------------------------------------------+
# | Network Configuration File: sqlnet.ora           |
# | Generated by BNGX configuration tool: -          |
# |   - brdb_edit_sqlnet.sh (using 'tag' values)     |
# |                                                  |
# +--------------------------------------------------+

SQLNET.INBOUND_CONNECT_TIMEOUT=60
SQLNET.EXPIRE_TIME=5
bequeath_detach=true
```

tnsnames.ora

The tnsnames.ora would ordinarily only have entries that are applicable to the instance(s) which exist on that node alone. However, the build process uses a single tnsnames.ora for all nodes. This is not ideal, but is how it has been delivered.

```
LISTENER_ IRRELEVANT =
 (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = IRRELEVANT  (PORT = 1529))
  )


BRDB1 =
 (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = IRRELEVANT (PORT = 1529))
   (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = BRDB1)
     (INSTANCE_NAME = BRDB1 )
    )
  )
```

listener.ora

```
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER))))
# line added by Agent
LISTENER_SCAN1=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENE
R_SCAN1))))                  # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1=ON                    # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER=ON                # line added by Agent
```

FUJ00234958

LISTENER:
LISTENER_SCAN1:
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1:
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER:

endpoints_listener.ora

```
LISTENER_LSDPBDB501=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=l
sdpbdb501-
vip)(PORT=1529))(ADDRESS=(PROTOCOL=TCP)(HOST=172.23.207.71)(PORT=1529)(IP=FIR
ST))))              # line added by Agent
```

LISTENER_LSDPBDB501: LOCAL LISTENER FOR DATABASE

INBOUND_CONNECT_TIMEOUT_LISTENER_LPRPBDB201 = 10 determines the number of ***seconds*** Oracle will wait to receive authentication from the client making the connection. Otherwise denies the request.

ADMIN_RESTRICTIONS_LISTENER_LPRPBDB201 = ON enforces the administration of the listener to an authorised user only, i.e. ***oracle***

5.1.1.2 Interaction with the Listener

Starting and stopping the listener is done via Oracle CRS as follows: -

```
lsdpbdb501:oracle:>. oraenv
ORACLE_SID = [BRDB1] ? +ASM1
The Oracle base remains unchanged with value /u01/app/oracle
lsdpbdb501:oracle:>lsnrctl status

LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 23-JUN-2014 09:11:42

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                20-JUN-2014 14:09:47
Uptime                    2 days 19 hr. 1 min. 55 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/11.2.0/grid/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/lsdpbdb501/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.23.207.71)(PORT=1529)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.23.207.73)(PORT=1529)))
Services Summary...
Service "+ASM" has 1 instance(s).
  Instance "+ASM1", status READY, has 1 handler(s) for this service...
Service "BRDB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
Service "BRDB_DGB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
Service "SYS$OPS$OGGADMIN.OGG$Q_E11BDB.BRDB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
The command completed successfully
lsdpbdb501:oracle:>srvctl stop listener -n lsdpbdb501
```

FUJ00234958


```
lsdpbdb501:oracle:>lsnrctl status

LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 23-JUN-2014 09:13:10

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
   Linux Error: 2: No such file or directory
lsdpbdb501:oracle:>srvctl start listener -n lsdpbdb501
```

Checking the status of the listener and it's services is done as follows: -

```
lsdpbdb501:oracle:>lsnrctl status

LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 23-JUN-2014 09:13:55

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                23-JUN-2014 09:13:26
Uptime                    0 days 0 hr. 0 min. 29 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/11.2.0/grid/network/admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/lsdpbdb501/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.23.207.71)(PORT=1529)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.23.207.73)(PORT=1529)))
Services Summary...
Service "+ASM" has 1 instance(s).
  Instance "+ASM1", status READY, has 1 handler(s) for this service...
Service "BRDB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
Service "BRDB_DGB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
Service "SYS$OPS$OGGADMIN.OGG$Q_E11BDB.BRDB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Executing **`lsnrctl services LISTENER`** will show a little more information for each service than the **`status`** command.

The important services used are listed as follows: -

`+ASM[1234]` This service is required for Grid Control and allows access to ASM.

`BRDB` This service is generally required for the BAL and TWS and allows those applications to connect without specifying an individual instance.

"SYSOPSOGGADMIN.OGG$Q_E11BDB.BRDB This service is required for Goldengate.

`BRDB_DGB` Oracle defined service related to Data Guard.

If any services are not created, then client connections which use those services will be unable to connect. This is similar to the status of the listener itself in that unless it is continually being monitored, the only way one will really know there is an issue, is with the inability to connect.

5.1.2 General Recommendations

FUJ00234958

5.1.2.1 Logs and Trace Files.

From time to time there will be important log files, trace files and background process dump files that will be needed for support purposes and would have been explicitly renamed and "saved" by support personnel. These files, if found in a "house kept" directory, will be removed by the housekeeping processes after the retention period has been exceeded. For quick reference those directories are: -

/u01/app/oracle/diag/rdbms/<DB>/<INSTANCE>/alert

/u01/app/oracle/diag/rdbms/<DB>/<INSTANCE>/cdump

/u01/app/oracle/diag/rdbms/<DB>/<INSTANCE>/trace

The database alert log and the listener log files are always being written to and are important files. It is highly recommended that these files are kept manageable. A good way of doing this would be to copy the files every month or fortnightly in order to keep a history and keep their sizes at a manageable level.

5.1.3 Password Management

In general all Branch Database and Branch Support Database passwords fall into one of three categories: -

- The users are locked (within the database) and even if the password is known, logging on is not a possibility.
- The passwords are managed by Microsoft Active Directory. This is possible because the users that this applies to are "externally identified" and in order to logon, one must be logged onto the server as an OS user and then log onto the database, thereby relying on OS authentication.
- The passwords are set by privileged users and known to only secure/trusted personnel. This can only apply to privileged users, e.g. SYSTEM, SYS, DBSNMP, etc. The following table shows interdependencies of database users of this type: -

User	Interdependencies	Risk If Changed
SYS (See Section 5.1.3.1)	Oracle Grid Control Oracle Data Guard RMAN BACKUP	Grid Control Agents will be unable to logon Standby Database log shipping and coordination will fail Rman backups will be unable to logon
SYSTEM	None	None
DBSNMP	Oracle Grid Control	Grid Control Agents will be unable to logon
AUDITUSER	The Audit Server	Audit Server will fail to logon
BMC_USERLV BMC_USERTR	BMC Patrol	None
BRDBRDDS	RDDS Feeds	None
BRDBRDMC	RDMC Feeds	None
DELTRUSER	Counter Training	None
EMDB_SUP	The EMDB Interface	None

FUJ00234958

User	Interdependencies	Risk If Changed
OMDBUSER	The OMDB Interface	
LVBALUSER[1-4]	Live Counter Connections	The BAL OSR will fail to startup correctly
ORAEXCPLV	BRDB Exception logging	In the event of a failure, BRDB processes will not be able to log exceptions
REP_GEN	Generic Reporting	Reports will fail to generate
TRBALUSER[1-4]	Training Counter Connections	Counter training will not be possible
TWS TWSSUP	The TWS Scheduler	All schedules will fail to run
OPS$OGGADMIN	OGG extract + datapump	Goldengate will fail to operate correctly
BDAS_QUERY BDAS_AUTH	BDAS_AUTH is dependent on BDAS_QUERY	BDAS Appliaction will fail to operate correctly

5.1.3.1 Changing the SYSDBA Password

The SYS passwords have related *sysdba* password files for both the main application instance and ASM instance on *all nodes* of any Online RAC Cluster. The significance of the password file is that the password internal to the database (for the SYS user) must match the password with which the password file was created. If either of them changes without the other, all remote logons will fail with an "Insufficient Privileges" ORA- error.

Password file(s) must be changed alongside any password change. Oracle Grid Control and Oracle Data Guard rely on being able to logon remotely as privileged users.

The instances affected on ***BDB*** are as follows: -

BRDB[1|2|3|4] and +ASM[1|2|3|4]

The instances affected on ***BDS*** are as follows: -

SBRDB[1|2|3|4] and +ASM[1|2|3|4]

The instances affected on ***BRS*** are as follows: -

BRSS[1] and +ASM[1]

Then on every node a password file will exist in `<ORACLE_HOME>/dbs` of the form `orapw<ORACLE_SID>`, for each instance above.

For example should one wish to change the 'SYS' password on BDB node 3 to 'b0bsy0uruncl3', one would perform the following tasks as the ***oracle*** user logged onto ***node 3***: -

Logon to BRDB3 and change the password:

```
$oracle > . oraenv
ORACLE_SID = [grid] ? BRDB3
$oracle> sqlplus '/as sysdba'
SQL> ALTER USER SYS IDENTIFIED BY b0bsy0uruncl3;
SQL> EXIT;
```

Recreate the password file on all nodes in the cluster:

FUJ00234958


```
$oracle> cd $ORACLE_HOME/dbs
$oracle> orapwd file=/u01/app/oracle/product/11.2.0/dbhome_3/dbs/orapwBRDB3
password=b0bsy0uruncl3 entries=5 ignorecase=y force=y
```

Note: The process for changing the ASM password is the same as that for the database instance but must be logged on as grid user. If this is changed, the password information must be relayed to MSS for monitoring to continue.

Recreate the password file on all nodes in the cluster:

Logon to BRDB node 3 as "grid" UNIX user

```
$grid > . oraenv
ORACLE_SID = [grid] ? +ASM3
$grid> cd $ORACLE_HOME/dbs
$grid> orapwd file=$ORACLE_HOME/dbs/orapw+ASM3 password=b0bsy0uruncl3
entries=5 ignorecase=y force=y
```

Run the following on **SBRDB Node 1** ONLY as UNIX user "oracle".

$oracle:> /app_sw/brdb/patch/rman_set_pwd.sh -v -u sys -d BRDB -- This will prompt for password. Use the same sys password set on BRDB database. (This changes for all nodes due to file residing on ACFS filesystem)

Stop and restart the standby databases and ensure logs are currently shipped from PRIMARY to STANDBY.

5.1.3.2 Listener Password

The database listeners (one on each node) have their access restricted by privileged users only, e.g. root or oracle. The listeners are not password protected.

5.2 Backups

5.2.1 Database Backups

See Section 4 for more detail.

5.2.2 Disk Backups

Most disks in the Primergy BX900 are protected by either being mirrored or the disks will be replicated via REC (Eternus Storage).

5.3 Partition Management

5.3.1 Introduction

This section does not detail specific functionality but is intended to provide an overview of how the use of physical partitions works and to handle the partition creation failure. The partition management describes in this section applied to both the BRDB and BRSS.

Note this section does NOT include how partitions are created and archived off though where appropriate, reference is made to interactions.

5.3.2 Assumptions

It is assumed physical partitions exist for each partitioned table for the desired processing date.

FUJ00234958

5.3.3 Overview

The creation and removed physical partitions for each partitioned table is performed by start of day job; i.e. BRDBC001 and BRSSC001.

The operation of the start of day process is defined in LLD.

5.3.3.1 Partition Metadata

The operation of the partition table is driven by the following metadata:

5.3.3.1.1 <BRDB/BRSS>_PARTITION_CREATES

This table is used to record the creation, status change and removal of partitions by the Start of Day housekeeping for support and audit purposes.

5.3.3.1.2 <BRDB/BRSS>_PARTITION_STATUS_HISTORY

This table is used to record the history of the created partition. The entry is inserted by Start of the Day process (<BRDB/BRSS>C0001).

5.3.3.1.3 <BRDB/BRSS>_SUBPARTITION_RANGES

The entry in this table will contain the next partition (range value) that will be created by Start of Day process (<BRDB/BRSS>C0001). The partition range value will be increment by 1 at the end of the process.

5.3.3.1.4 <BRDB/BRSS>_PROCESS_CONTROL

This table holds process run information and in this case it contains the partition creation information for each table. This table is used for re-run of the Start of the day process for the failure partition.

5.3.4 Troubleshooting

The Start of Day (BRDBC0001/BRSSC001) process creates physical partitions for a number of days ahead (configurable via system parameter PARTITIONS_AHEAD), therefore this process would have to fail for several days in succession and not have been corrected in order for the partitions to be missing for the current day. This most likely occurs due to insufficient space available in the corresponding tablespace for which an Operational Exception would be generated. The process can be restarted after rectifying the cause of failure.

BRDC001 can run either as Pre Release 9 with no input parameter or Extended hours with tws date as input parameter, i.e. $BRDB_PROC/BRDBC001 ^BRDBBDAY^ .

The partition creation rules for BRDBC001 are:

1. If the current system time is beyond the allowable time in hhmm specified by 'PARTITIONS_EXPIRED_TIME' System Parameter (currently set to 0500), then it does not begin to create/delete partitions for a new day. The process will, however, exit with an error if physical partitions for the next day failed to create..
2. BRDBC001 continues to create partitions for all the partitioned tables on a daily basis (one day at a time)
3. Repeat (1) until 'n' ahead partitions have been created.

Pre Release 9 mode (ie BRDBC001 with no argument) is required to run if the job was abandoned due to time exceeding the allowed period and partitions for the next business day do not exist.

FUJ00234958

Note that it is possible due to the unavoidable implicit database commit performed when adding/dropping table partitions that, in some esoteric failure scenarios, the partition metadata will be out of sync with the actual partitions. In this situation, re-running the SOD process will potentially fail.

In this scenario it will be necessary to confirm whether the metadata/partitions are inconsistent by running a script provided by development (see further sections).

If the partitions/metadata is inconsistent it will be necessary to manipulate either to remedy the situation. Given that the remedial activity will be dependent on a number of variables including whether any data has been written to the new partitions etc, a call should be raised with 4th line support.

In some situations, typically in test, it is desirable to run BRDBC001/BRSSC001 more than once in a calendar day. The default (build) value of the PROCESS_DAY_MULTIPLE_RUNS_YN flag in the <BRDB/BRSS>_PROCESSES table for the <BRDB/BRSS>C001 process is 'N' so would prevent this. Therefore the PROCESS_DAY_MULTIPLE_RUNS_YN flag should be changed to 'Y' to allow this if required.

WARNING – This should only be done in Live at the guidance of development.

The following is a checklist in the event the <BRDB/BRSS>C001 job fails (to be done before re-running the job): -

i. Check the entry in <BRDB/BRSS>_OPERATIONAL_EXCEPTONS and this will show the error(s) that cause the job to failure.

ii. Check the 'parameter value for 'BRDB SYSTEM DATE' from '<BRDB/BRSS>_SYSTEM_PARAMETERS table. It should set to (N – 1) where N is current system date.

iii. Check the column 'SYSTEM_DATE','START_DATE' and 'END_DATE' in the <BRDB/BRSS>_PROCESS_CONTROL table. This table is used to control the process for each Table-Group and table affected.

SYSTEM_DATE should equal to the '<BRDB/BRSS> SYSTEM DATE' from the <BRDB/BRSS>_SYSTEM_PARAMETER table

END_DATE should have the NULL value for the failure partition table.

iv. Check the 'RANGE_VALUE' from the <BRDB/BRSS>_SUBPARTITION_RANGES table. This value should equal to '<BRDB/BRSS> SYSTEM DATE' + 2 in the format of 'YYYYMMDD'

v. Check the table <BRDB/BRSS>_PARTITION_CREATES and <BRDB/BRSS>_PARITION_STATUS_HISTORY. The failure partition_range_value for the partition table must not exist in the above tables.

vi. Check the value of the PROCESS_DAY_MULTIPLE_RUNS_YN flag in the <BRDB/BRSS>_PROCESSES table for the <BRDB/BRSS>C001 process is 'N'.

There is another option to fix a single partition by passing the parameters to the Start of Day; i.e.

```
<BRDB/BRSS>C001 [<Table-Group> <Table-Name> <Partition-Date (YYYYMMDD)>]
<SYSTEM_DATE(YYYYMMDD)>
```

Where SYSTEM_DATE is optional when exist and this value will set in '<BRDB/BRSS> SYSTEM DATE'.

5.3.4.1 Determining Exception Information

As it is entirely possible for SOD (<BRDB|BRSS>C001) to fail during the normal day-to-day overnight run, the following query will help in diagnosing problems and give greater detail as to the reason(s) for failure. This query will show all exceptions for the last 24 hours, the last of which will be displayed first: -

```
set lines 200 pages 90
col exception_detail FOR a70
```

FUJ00234958


```
col exception_object FOR a20
col process_name FOR a20
col exception_timestamp FOR a30

SELECT exception_timestamp, exception_detail, exception_object, process_name
  FROM brdb_operational_exceptions
 WHERE exception_timestamp >= SYSDATE - 1
 ORDER BY exception_timestamp DESC;
```

5.3.4.2 Useful Queries

The below scripts reconcile differences between physical partitions and partition metadata maintained by the BRDB/BRSS application.

These scripts should not be run unless directed by Development support staff.

Updates status for records in <BRDB/BRSS>_PARTITION_CREATES table to 'ARCH' where the Status is set to 'DEL' and the partition exists in the database: -

```
UPDATE     <brdb/brss>_partition_creates bpc
SET        bpc.status = 'ARCH'
WHERE      bpc.status = 'DEL'
AND        EXISTS (SELECT 'x'
                   FROM   all_tab_partitions atp,
                          <brdb/brss>_partitioned_tables bpt
                   WHERE  atp.table_owner = 'OPS$<BRDB/BRSS>'
                   AND    atp.table_name = bpc.table_name
                   AND    atp.table_name = bpt.table_name
                   AND    atp.partition_name = bpt.partition_root_name || '_'
                          || bpc.partition_range_value);
```

Updates status for records in <***BRDB/BRSS***>_PARTITION_STATUS_HISTORY table to 'ARCH' where the Status is set to 'DEL' and the partition exists in the database: -

```
UPDATE     <brdb/brss>_partition_status_history bpsh
SET        bpsh.status = 'ARCH'
WHERE      bpsh.status = 'DEL'
AND        EXISTS (SELECT 'x'
                   FROM   all_tab_partitions atp
                   WHERE  atp.table_owner = 'OPS$<BRDB/BRSS>'
                   AND    atp.table_name = bpsh.table_name
                   AND    atp.partition_name = bpsh.partition_name) ;
```

Updates mismatched records in <***BRDB/BRSS***>_PARTITION_CREATES table to 'DEL': -

```
UPDATE     <brdb/brss>_partition_creates bpc
SET        bpc.status = 'DEL'
WHERE      bpc.status != 'DEL'
AND        NOT EXISTS (SELECT 'x'
                       FROM   all_tab_partitions atp,
                              <brdb/brss>_partitioned_tables bpt
                       WHERE  atp.table_owner = 'OPS$<BRDB/BRSS>'
                       AND    atp.table_name = bpc.table_name
                       AND    atp.table_name = bpt.table_name
                       AND    atp.partition_name = bpt.partition_root_name ||
                              '_' || bpc.partition_range_value) ;
```

FUJ00234958

Updates mismatched records in <BRDB/BRSS>_PARTITION_STATUS_HISTORY table to 'DEL'

```
UPDATE     <brdb/brss>_partition_status_history bpsh
SET        bpsh.status = 'DEL'
WHERE      bpsh.status != 'DEL'
AND        NOT EXISTS (SELECT 'x'
                       FROM   all_tab_partitions atp
                       WHERE  atp.table_owner = 'OPS$<BRDB/BRSS>'
                       AND    atp.table_name = bpsh.table_name
                       AND    atp.partition_name = bpsh.partition_name)
AND        create_date = (SELECT   MAX(bpsh1.create_date)
                          FROM     <brdb/brss>_partition_status_history bpsh1
                          WHERE    bpsh1.table_name = bpsh.table_name
                          AND     bpsh1.partition_name = bpsh.partition_name);
```

Inserts missing records into <***BRDB/BRSS***>_PARTITION_CREATES table: -

```
INSERT INTO <brdb/brss>_partition_creates
       (table_name,
        partition_range_value,
        status,
        status_date)
SELECT atp.table_name,
       substr(atp.partition_name,
       LENGTH(npt.partition_root_name) + 2) partition_range_value,
       'NEW',
       SYSDATE
  FROM   all_tab_partitions atp,
       <brdb/brss>_partitioned_tables bpt
 WHERE  atp.table_owner = 'OPS$<BRDB/BRSS>'
   AND    atp.table_name = bpt.table_name
   AND    NOT EXISTS (SELECT 'x'
                        FROM <brdb/brss>_partition_creates bpc
                       WHERE bpc.table_name = atp.table_name
                         AND bpc.partition_range_value =
SUBSTR(atp.partition_name,
                             LENGTH(bpt.partition_root_name) + 2));
```

Inserts missing records into <***BRDB/BRSS***>_PARTITION_STATUS_HISTORY table: -

```
INSERT INTO <brdb/brss>_partition_status_history (
                     table_name, partition_name,
                     create_date, status, sql_statement)
SELECT atp.table_name,
       atp.partition_name,
       SYSDATE,
       'NEW',
       'METADATA CORRECTION UTILITY FROM SUPPORT GUIDE'
FROM   all_tab_partitions atp,
       <brdb/brss>_partitioned_tables bpt
WHERE  atp.table_owner = 'OPS$<BRDB/BRSS>'
AND    atp.table_name = bpt.table_name
AND    NOT EXISTS (SELECT   'x'
                   FROM     <brdb/brss>_partition_status_history bpsh
```

FUJ00234958


```
WHERE     bpsh.table_name      = atp.table_name
AND       bpsh.partition_name = atp.partition_name);
```

Check the partition that will .be created when BRDBC001/BRSSC001 next run: -

```
SELECT table_name,
       range_value
FROM   <brdb/brss>_subpartition_ranges;
```

Check the latest partition created in the system: -

```
SELECT    table_name,
          max(partition_range_value),
FROM      <brdb/brss>_partition_creates
GROUP BY table_name
ORDER BY table_name;
```

"pt_clean.sh" shell script can be used to rebuilt the meta partition tables (<***brdb/brss***>_partition_creates,

<***brdb/brss***>_subpartition_ranges and <***brdb/brss***>_partition_status_history) from the database. This shell script can be found in /app_sw/brdb/build/schema or /app_sw/brss/build/schema.

NB. This script will set status to 'ARCH' in the <***brdb/brss***>_partition_creates and all the partitions will be deleted when the <***BRDB/BRSS***>C0001 next run.

5.4 Standby Database

5.4.1 Introduction

The build of and theory surrounding the BRDB Standby database (SBRDB) is detailed extensively in the Standby Database Low Level Design [DEV/APP/LLD/0152].

5.4.2 Assumptions

The Primary Database BRDB will be running on a 4-node cluster and the Standby Database on a 4-node cluster configuration.

The Data Guard Configuration has been successfully built and running without errors.

5.4.3 Troubleshooting

The very first thing one should consider when troubleshooting is to consider the status of the architectural components surrounding the solution, e.g. the network, the SAN, the BladeFrame, etc. (see Section 5.4.3.1)

Oracle has a number of processes on both the Primary database and the Standby database monitoring the sending, the transportation and the receiving of replicated redo from source to the destination.

It is important to note that the Data Guard Broker is key to the monitoring of the solution without which, the seamless failover to Standby from Primary would not be possible nor would the trouble free monitoring through Grid Control be possible.

5.4.3.1 Checklist

Is the database in recovery mode or is it down (all nodes)?

FUJ00234958

Is there enough storage space, e.g. check **+SBRDB_FLASH**? Do all the file systems have sufficient free space?

Is the network up?

Have you checked the Data Guard Monitor status, e.g. **dgmgrl … show configuration**? Is it showing **SUCCESS** (see Section 5.4.3.3)?

Have you checked the Data Guard logs on Standby and Primary, e.g.,

/u01/app/oracle/diag/rdbms/<DB>/<INSTANCE>/alert/drc<INSTANCE>.log?

5.4.3.2 Useful Queries

This query will help identify Data Guard problems (On BRDB or SBRDB).

```
SET lines 100
SET pages 45
ALTER SESSION SET NLS_DATE_FORMAT='DD-MON HH24:MI:SS';
SELECT facility,
       error_code,
       TIMESTAMP,
       message
  FROM v$dataguard_status
 ORDER BY message_num;
```

This query will help with determining if any Standby Logs are not in use when they should be (On SBRDB). It does not matter what group the standby logs belong to, but one should see 1 log for every primary instance, e.g. 1, 2, 3 and 4 in LIVE.

```
SET lines 100
SET pages 45
ALTER SESSION SET NLS_DATE_FORMAT='DD-MON HH24:MI:SS';
SELECT group#,
       thread#,
       sequence#,
       archived,
       status,
       last_time
  FROM v$standby_log WHERE status <> 'UNASSIGNED';
```

5.4.3.3 Useful Tools

Data Guard Monitor is very important for monitoring the status of the Data Guard Configuration and is not possible without the Data Guard Broker. The broker is started automatically – at instance startup - by setting the database initialisation parameter **dg_broker_start** to **TRUE**. The broker is in essence the DMON process and writes information to a log called /u01/app/oracle/diag/rdbms/<DB>/<INSTANCE>/alert/drc<INSTANCE>.log in which all status and error information can be monitored/viewed.

The Data Guard Monitor Command-line Utility or DGMGRL can be used to get useful feedback from the configuration, e.g. …

```
$> dgmgrl
DGMGRL for Linux: Version 11.2.0.4.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect /
Connected.
```

FUJ00234958


```
DGMGRL> show configuration

Configuration
  Name:                BRDB_DATAGUARD_CFG
  Enabled:             YES
  Protection Mode:     MaxPerformance
  Fast-Start Failover: DISABLED
  Databases:
    BRDB  - Primary database
    SBRDB - Physical standby database

Current status for "BRDB_DATAGUARD_CFG":
SUCCESS
```

FUJ00234958

5.5 Oracle Goldengate

5.5.1 Introduction

Goldengate configuration and activation for BRDB and BRSS are detailed extensively in the BRDB High Level Design [DES/APP/HLD/0020], BRSS High Level Design [DES/APP/HLD/0023], Low Level Design [DEV/APP/LLD/0151] and Goldengate LLD [DEV/APP/LLD/2432]

5.5.2 Assumptions

A single-source replication environment is configured and has the following characteristics:

- One Manager process, controlled via CRS, this process monitors the other OGG jobs
- One Integrated Extract/Capture process named E11BDB located in BRDB node 1
- One Data Pump/Propagation process named P11BDB located in BRDB node 1.
- One Integrated Replicat/Apply process named R11BRS located in BRSS node 1.

5.5.3 Shutting Down & Starting up Goldengate for Baseline Application

This part covers the steps required to shut Goldengate down prior to applying baselines that might affect Goldengate.

5.5.3.1 Stopping BRDB Goldengate Processes

1. Login to BDB as 'oggadmin' user
2. Invoke ggsci:

```
$OGG_HOME/ggsci
```

3. Stop extract:

```
GGSCI> stop E11BDB


Sending STOP request to EXTRACT E11BDB ...

Request processed.
```

4. Issue info all repeatedly until E11BDB is shown as STOPPED

```
GGSCI> info all


Program     Status      Group       Lag at Chkpt  Time Since Chkpt
```

FUJ00234958


```
MANAGER     RUNNING
EXTRACT     STOPPED     E11BDB      00:00:05      00:01:58
EXTRACT     RUNNING     P11BDB      00:00:00      00:00:06
```

5. Wait until all lag disappears against data pump P11BDB by issuing::

```
GGSCI> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
EXTRACT     STOPPED     E11BDB      00:00:05      00:01:58
EXTRACT     RUNNING     P11BDB      00:00:00      00:00:06
```

6. Stop datapump once lag is gone:

```
GGSCI> stop P11BDB

Sending STOP request to EXTRACT P11BDB ...
Request processed.
```

7. Ensure P11BDB is now stopped by issuing::

```
GGSCI> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
EXTRACT     STOPPED     E11BDB      00:00:05      00:05:32
EXTRACT     STOPPED     P11BDB      00:00:00      00:02:25
```

8. Stop all OGG processes via CRS (logon as 'oracle' user):

```
oracle> crsctl stop res brdb.oggadmin.oggapp
oracle> crsctl stop resource dbfs_mount
```

FUJ00234958

5.5.3.2 Stopping BRSS Goldengate Processes

1. Login to BRS as 'oggadmin' user
2. Invoke ggsci:

```
$OGG_HOME/ggsci
```

3. Ensure there is no lag associated with replicat R11BRS by issuing:

```
GGSCI> info all


Program     Status      Group       Lag at Chkpt  Time Since Chkpt


MANAGER     RUNNING

REPLICAT    RUNNING     R11BRS      00:00:00      00:00:03
```

4. Login as 'oracle' user
5. Stop all OGG processes on BRS

```
oracle> crsctl stop res brss.oggadmin.oggapp
```

5.5.3.3 Apply Baselines

Maintenance or baseline activity would take place once all Goldengate processes were down.

5.5.3.4 Starting BRDB Goldengate Processes

1. Login to BDB as 'oracle' user, start DBFS_MOUNT and OGG processes (where lxxpbdb201 is dependent on the environment – LST or LIVE)

```
oracle> crsctl start resource dbfs_mount

oracle> crsctl start res brdb.oggadmin.oggapp -n lxxpbdb201 -f
```

2. Login as 'oggadmin' invoke ggsci:

```
$OGG_HOME/ggsci
```

FUJ00234958

3. Check the OGG processes are running

```
GGSCI>        info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt


MANAGER     RUNNING

EXTRACT     RUNNING     E11BDB      00:00:00      00:00:00

EXTRACT     RUNNING     P11BDB      00:00:00      00:00:00
```

5.5.3.5 Starting BRSS Goldengate Proceeses

1. Login as 'oracle' user
2. Start all OGG processes on BRS via CRS

```
oracle> crsctl start res brss.oggadmin.oggapp
```

3. Login to BRS as 'oggadmin' user
4. Invoke ggsci:

```
$OGG_HOME/ggsci
```

5. Ensure the replicat R11BRS is running by issuing:

```
GGSCI> info all


Program     Status      Group       Lag at Chkpt  Time Since Chkpt


MANAGER     RUNNING

REPLICAT    RUNNING     R11BRS      00:00:00      00:00:03
```

5.5.4 Troubleshooting

FUJ00234958

The following is a list of tables and views that are useful, in troubleshooting OGG issues. This is basically "reference" information (more detailed information can be found in the Oracle Goldengate administration guide):

All OGG processes:

```
select process_name, instance_id, insert_timestamp, process_status
from   ops$oggadmin.brdb_brss_gg_monitoring
where  (process_name, instance_id, insert_timestamp) in (
   select process_name, instance_id, max(insert_timestamp)
   from   ops$oggadmin.brdb_brss_gg_monitoring
   group  by process_name, instance_id
);
```

Extract Process

dba_capture:	basic status, error info
gv$goldengate_capture:	detailed current status info
dba_capture_parameters:	configuration information

Replicat Process

v$gg_apply_receiver	basic status, error info
all_gg_inbound_progress	high/low apply positions for replicat

5.5.4.1 OGG Commands

OGG_Monitoring_Commands.pdf

5.5.4.2 Troubleshooting Capture Problems

The Oracle OGG Extract process utilises the Logminer process within the Branch Database to capture all DML changes to objects owned by OPS$BRDB. The Extract process may stop capturing changes, some of the useful methods describes in this section can use to diagnose the problem and resolve them.

The manager process will attempt to restart the extract a configurable number of times (see E11BDB.prm for current number of attempts).

Check capture process status:

The Capture Process captures changes only when it is ***ENABLED***. One can check whether the process is enabled, disabled, or aborted by querying the ***DBA_CAPTURE*** data dictionary view:

```
SELECT capture_name, status
FROM   dba_capture
WHERE  capture_name like '%E11BDB';
```

If the capture process is disabled, then try restarting it.

If the capture is aborted, then it needs to correct an error before restarting it. The following query shows when the capture process aborted and the error that caused it to abort:

```
SELECT status_change_time, error_message
```

FUJ00234958


```
  FROM dba_capture
 WHERE status = 'ABORTED' AND capture_name like '%E11BDB';
```

Check Capture current status: -

The state of a capture process describes what the capture process is doing currently. One can view the state of a capture process by querying the STATE column in the V$STREAMS_CAPTURE dynamic performance view.

```
SELECT state
  FROM v$streams_capture
 WHERE capture_name like '%E11BDB';
```

The following capture process states are possible: -

INITIALIZING: Starting up.

CAPTURING CHANGES: Scanning the redo log for changes that evaluate to TRUE against the capture process rule sets.

EVALUATING RULE: Evaluating a change against a capture process rule set.

CREATING LCR: Converting a change into an LCR.

ENQUEUING MESSAGE: Enqueuing an LCR that satisfies the capture process rule sets into the capture process queue.

SHUTTING DOWN: Stopping.

WAITING FOR DICTIONARY REDO: Waiting for redo log files containing the dictionary build related to the first SCN to be added to the capture process session. A capture process cannot begin to scan the redo log files until all of the log files containing the dictionary build have been added.

DICTONARY INITIALIZATION: Processing a dictionary build.

MINING (PROCESSED SCN = scn_value): Mining a dictionary build at the SCN scn_value.

LOAD (step X 0f Y): Processing information from a dictionary build and currently at step X in a process that involves Y steps, where X and Y are number.

PAUSED FOR FLOW CONTROL: Unable to enqueue LCRs either because of low memory or because propagations and apply processes are consuming messages slower than the capture process is creating them. This state indicates flow control that is used to reduce spilling of captured messages when propagation or apply has fallen behind or is unavailable.

Common capture issues: -

1. ORA-01291: missing logfile.

   A missing redo is possible when a logfile is dropped for any administrative reasons. The v$logmnr_logs can be checked to determine the missing SCN range and add the relevant redo log files

   Query the REQUIRED_CHECKPOINT_SCN column in the DBA_CAPTURE to determine the required checkpoint SCN for a captured. Then restore the redo log file that includes the required checkpoint SCN and all subsequent redo log files.

2. Capture process loops on startup.

   This may be a missing logfile which cannot be opened. All logs from the BRDB nodes (1|2|3|4) have to be present with respect to the required_checkpoint_scn.

3. Capture process is in “PAUSED FOR FLOW CONTROL” or “ENQUEUING MESSAGE” status.

FUJ00234958

- Check the source queue, as there is probably a large amount of LCRs being spilled to disk.
- Check if the destination site is down.
- Check the propagation and apply status'.

5.5.4.3 Troubleshooting Data Pump Problems

The Oracle OGG data pump process resides on the Branch Database and propagates trail files to the target database (BRSS) filesystem. The data pump process will abend if

- BRSS's OGG processes are down
- The BRSS platform is unavailable
- The BRSS file system (DBFS) is unavailable - e.g. the BRSS database is down

The manager process will attempt to restart the data pump a configurable number of times (see P11BDB.prm for current number of attempts).

5.5.4.4 Troubleshooting Replicat Problems

The Oracle OGG Replicat process resides within the Branch Support Database. The replicat reads the trail files (/u02/goldengate/dirdat/bz*) and applies them to the OPS$BRDB schema. Some of the useful methods describes in this section can use to diagnose the apply problem and resolve them.

Exceptions raised when attempting to apply changes are inserted into OPS$OGGADMIN.OGG_EXCEPTIONS. The replicat then continues to operate without abending.

Check apply process status:

An apply process applies changes only when it is enabled. Query the STATUS column in DBA_APPLY to determine the state of the apply process: -

```
SELECT  apply_name, status
  FROM  dba_apply
 WHERE  apply_name like '%R11BDB';
```

The possible values are ENABLED, DISABLED and ABORTED.

If the apply process is disabled, then try restarting it: -

```
DBMS_APPLY_ADM.START_APPLY( apply_name => 'BRSS_APPY');
```

If the apply process is aborted, then correct an error before restart the apply process. The following query shows when the apply process and the error that caused it to abort: -

```
SELECT status_change_time, error_message
  FROM dba_apply
 WHERE status     = 'ABORTED'
   AND apply_name like '%R11BDB';
```

If the apply process is enabled, but changes are not applied: -

Check that the apply process queue is receiving the messages to be applied using v$buffered_queues: -

```
SELECT queue_id, queue_name, (num_msgs - spill_msgs) mem_msgs,
       spill_msgs
  FROM v$buffered_queues
 WHERE queue_name like '%R11BDB';
```

Or using the v$streams_apply_coordinator view: -

FUJ00234958


```
SELECT total_received,
       total_applied,
       total_errors,(total_assigned - total_rollbacks + total_applied))
being_applied
  FROM v$streams_apply_coordinator
 WHERE apply_name like '%R11BDB';
```

Check the Error Table

When an apply process cannot apply a message, it

- records an exception in OPS$OGGADMIN.OGG_EXCEPTIONS
- records the original record's details in a discard file (/u02/goldengate/dirrpt/R11BRS*dsc)
- applies all other non-erroring parts of the transaction to BRSS
- continues to process other items within the trail file (i.e. replicat moves on)

Query OGG exception table OGG_EXCEPTIONS to determine if there are errors in the error queue.

```
SELECT count(*)
  FROM ops$oggadmin.ogg_exceptions
 WHERE resolved_yn = 'N';
```

5.5.4.5 Working with DML Exceptions

The OGG replicat process will record all rows of every sub-transaction that fails to apply. These failed transactions with their associated errors are available to query from OPS$OGGADMIN.OGG_EXCEPTIONS. The discard file will contain the contents of the failed operation. Errorred transactions can be fixed manually via the DB link (if it's an update or insert) once the root cause has been identified.

Once an exception has been resolved, manually update the record in OGG_EXCEPTIONS:

```
UPDATE OPS$OGGADMIN.OGG_EXCEPTIONS
SET    RESOLVED_YN = 'Y'
WHERE  logrba = :x
AND    logposition = :y;
```

FUJ00234958

5.5.4.6 Useful Queries

i. The following query displays the current status of the capture process

```
set lines 100
column capture_name            heading 'Capture|Name' format A12
column process_name            heading 'Capture|Process|Number' format A7
column sid                     heading 'Session|ID' format 999999
column serial#                 heading 'Session|Serial|Number' format 9999999
column state                   heading 'State' format A27
column total_messages_captured heading 'Redo|Entries|Evaluated|In Detail'
format 9999999
column total_messages_enqueued heading 'Total|LCRs|Enqueued' format 999999

SELECT c.capture_name,
       substr(s.program,instr(s.program,'(')+1,4) process_name,
       c.sid,
       c.serial#,
       c.state,
       c. total_messages_captured,
       c. total_messages_enqueued
  FROM v$streams_capture c, v$session s
 WHERE c.sid = s.sid
   AND c.serial# = s.serial#;
```

FUJ00234958

ii. Minimum Archive Log Necessary to Restart Capture

```
set lines 300
set pages 9999
set serveroutput on

DECLARE
  hScn number := 0;
  lScn number := 0;
  sScn number;
  ascn number;
  alog varchar2(1000);
BEGIN
  select min(start_scn), min(applied_scn) into sScn, ascn
    from dba_capture
   where capture_name = 'OGG$CAP_E11BDB';

  DBMS_OUTPUT.ENABLE(2000);

  for cr in (select distinct(a.ckpt_scn)
               from system.logmnr_restart_ckpt$ a
              where a.ckpt_scn <= ascn and a.valid = 1
                and exists (select * from system.logmnr_log$ l
                             where a.ckpt_scn between l.first_change# and
l.next_change#)
              order by a.ckpt_scn desc)
  loop
    if (hScn = 0) then
       hScn := cr.ckpt_scn;
    else
       lScn := cr.ckpt_scn;
       exit;
    end if;
  end loop;

  if lScn = 0 then
    lScn := sScn;
  end if;

 dbms_output.put_line('Capture will restart from SCN ' || lScn ||' in the
following file:');
   for cr in (select name, first_time
                from DBA_REGISTERED_ARCHIVED_LOG
               where lScn between first_scn and next_scn order by thread#)
   loop
     dbms_output.put_line(cr.name||' ('||cr.first_time||')');
   end loop;
end;
/
```

FUJ00234958

iii. Display Capture Status Error Message

```
set serveroutput on size 950000
set verify off
set feedback off
set lines 180
set pages 9999
prompt +============================================================+
prompt |Display Capture Status Error Message                        |
prompt +============================================================+

column capture_name        heading 'Capture|Process|Name' format A10
column status_change_time  heading 'Abort Time'
column error_number        heading 'Error Number' format 99999999
column error_message       heading 'Error Message' format A40 wrap

SELECT  capture_name, status_change_time  , error_number, error_message
 FROM     dba_capture
 WHERE  status='ABORTED'
  AND       capture_name = 'OGG$CAP_E11BDB' ;
```

iv. This query will help to Display Information about the Reader Server for Each Apply Process

```
column apply_name               heading 'Apply Process|Name' format A15
column apply_captured           heading 'Dequeues Captured|Messages?' format
A17
column process_name             heading 'Process|Name' format A7
column state                    heading 'State' format A17
column total_messages_dequeued  heading 'Total Messages|Dequeued' format
99999999

SELECT r.apply_name,
       ap.apply_captured,
       substr(s.program,instr(s.program,'(')+1,4) process_name,
       r.state,
       r. total_messages_dequeued
  FROM v$streams_apply_reader r, v$session s, dba_apply ap
 WHERE r.sid = s.sid
   AND r.serial# = s.serial#
   AND r.apply_name = ap.apply_name;
```

v. The following query displays information about the transactions received, applied, and being applied by the apply process:

```
column apply_name     heading 'Apply Process Name'         format A25
column total_received heading 'Total|Trans|Received'       format 99999999
column total_applied  heading 'Total|Trans|Applied'        format 99999999
column total_errors   heading 'Total|Apply|Errors'         format 9999
column being_applied  heading 'Total|Trans Being|Applied' format 99999999
column total_ignored  heading 'Total|Trans|Ignored'        format 99999999

  SELECT apply_name,
         total_received,
         total_applied,
         total_errors,
         (total_assigned  - (total_rollbacks + total_applied)) being_applied,
         total_ignored
    FROM v$streams_apply_coordinator;
```

5.6 SSC Transaction Correction Tools

5.6.1 BRDBX015 – Transaction Correction Tool

FUJ00234958

As the Transaction Correction Tool is no longer required (Ref# HNG-X CP2707 – Transaction Correction Tool – Decommissioning), the tool has been decommissioned in such a way that it cannot be used again.

5.6.2 BRDB Clear Stock Unit Lock (clear_su_lock.sh)

This tool allows members of the SSC group to unlock stock units for any given branch accounting code, locking user and stock unit. Any attempt to run the tool will be audited as well as the actual changes made and running user. The SSC user must have been granted the SSC role within the BRDB database prior to running this tool.

Validation and processing occurs in an Oracle package (OPS$SUPPORTTOOLUSER.PKG_BRDB_CLEAR_SU_LOCK) while the package is initially called by a shell script (clear_su_lock.sh) on the BRDB server.

The script is located in /app/brdb/trans/support/brdbx015/clear_su_lock.sh

See DEV/APP/LLD/0202 for more information.

5.6.2.1 Parameters

The tool must be supplied with 3 switches, each with a parameter:

Parameter	Parameter Name	Datatype	Example	Valid Input
-b	Branch Accounting Code	Number	999999	1 – 999999
-u	Lock Holder Username	STRING	USR123	A [1-15 chr]
-s	Stock Unit	STRING	DEF	0 - zzz

5.6.2.2 Executing

```
./clear_su_lock.sh -b <BRANCH_CODE> -u <LOCK_USER> -s <STOCK_UNIT>
```

5.6.2.3 Scheduling

This task is scheduled on an ad hoc basis, as and when stock units need to be unlocked.

5.6.2.4 Audit Records/Logging

Start and finish records are inserted into OPS$BRDB.BRDB_PROCESS_AUDIT with a process_name of 'BRDB_CLEAR_SU_LOCK'.

Each update to OPS$BRDB.BRDB_BRANCH_STOCK_UNITS is audited in OPS$BRDB.BRDB_TXN_CORR_TOOL_JOURNAL.

Any exceptions will be logged to OPS$BRDB.BRDB_OPERATIONAL_EXCEPTIONS with an exception code of 'BRDB_SU_LOCK' and process_name (package name) of 'PKG_BRDB_CLEAR_SU_LOCK'.

The script verbosity level is controlled by BRDB system parameter BRDB_CLEAR_SU_LOCK_DEBUG_LEVEL (parameter_number set to 1 initially), set parameter_number to 2 in order to view the SQL update statement as well as the XML string.

Log files from each run are stored in /app/brdb/trans/support/brdbx015/log.

5.6.2.5 Sample output

This is an example of the output written to standard output and the log file when the module is successful:

```
01 Dec 14:54:10 writelock...: Starting
01 Dec 14:54:10 writelock...:    Lock file /tmp/clear_su_lock.run.lock created
01 Dec 14:54:10 writelock...: Complete.
01 Dec 14:54:10
01 Dec 14:54:10 check_env...: Starting
01 Dec 14:54:10 check_env...: Complete.
01 Dec 14:54:10
01 Dec 14:54:10 Main.......: Environment OK
01 Dec 14:54:10
01 Dec 14:54:10 view_gvar...: Starting
01 Dec 14:54:10 view_gvar...:    WHOAMI.............  gseem01
01 Dec 14:54:10 view_gvar...:    PROGNAME...........  clear_su_lock.sh
```

FUJ00234958


```
01 Dec 14:54:10 view_gvar...:    SCRIPT.............  clear_su_lock
01 Dec 14:54:10 view_gvar...:    THISDIR............  /app/brdb/trans/support/brdbx015
01 Dec 14:54:10 view_gvar...:    LOG_FILE...........
/app/brdb/trans/support/brdbx015/log/clear_su_lock_20091201_145410.log
01 Dec 14:54:10 view_gvar...:    LOCK_FILE..........  /tmp/clear_su_lock.run.lock
01 Dec 14:54:10 view_gvar...:    TSTMP..............  20091201_145410
01 Dec 14:54:10 view_gvar...:    VERBOSE............  ON
01 Dec 14:54:10 view_gvar...:    APP................  BRDB
01 Dec 14:54:10 view_gvar...:    ORACLE_SID.........  BRDBA1
01 Dec 14:54:10 view_gvar...:    BRANCH_CODE........  2007
01 Dec 14:54:10 view_gvar...:    LOCK_USER..........  X
01 Dec 14:54:10 view_gvar...:    STOCK_UNIT.........  DEF
01 Dec 14:54:10 view_gvar...: Complete.
01 Dec 14:54:10
01 Dec 14:54:10 unlock.....: Starting
01 Dec 14:54:10
Enabling ssc role
Tue 01-Dec-2009 14:54:10.619 Set DEBUG LEVEL to 1
Tue 01-Dec-2009 14:54:10.619 Starting pkg_brdb_clear_su_lock.update_data
Tue 01-Dec-2009 14:54:10.619  Starting pkg_brdb_clear_su_lock.process_audit
Tue 01-Dec-2009 14:54:10.620  Completed pkg_brdb_clear_su_lock.process_audit
Tue 01-Dec-2009 14:54:10.620   INFO: Parameter p_branch_code = 2007
Tue 01-Dec-2009 14:54:10.620   INFO: Parameter p_rollover_lock_user = X
Tue 01-Dec-2009 14:54:10.620   INFO: Parameter p_stock_unit: DEF
Tue 01-Dec-2009 14:54:10.620 Starting pkg_brdb_clear_su_lock.validate_parameters
Tue 01-Dec-2009 14:54:10.621   INFO: Validating branch_accounting_code
Tue 01-Dec-2009 14:54:10.623   OK: Branch Accounting Code: 2007 is open and exists in
OPS$BRDB.BRDB_BRANCH_INFO
Tue 01-Dec-2009 14:54:10.623   OK: Branch Accounting Code: 2007 exists in
OPS$BRDB.BRDB_TXN_CORR_TOOL_CTL
Tue 01-Dec-2009 14:54:10.628   OK: Stock unit DEF is locked for branch accounting code 2007
Tue 01-Dec-2009 14:54:10.628   OK: Stock unit DEF is locked by X
Tue 01-Dec-2009 14:54:10.629   OK: OPS$SUPPORTTOOLUSER is allowed to update
BRDB_BRANCH_STOCK_UNITS
Tue 01-Dec-2009 14:54:10.629   OK: Input parameters validated successfully
Tue 01-Dec-2009 14:54:10.629 Completed pkg_brdb_clear_su_lock.validate_parameters
Tue 01-Dec-2009 14:54:10.629 Starting pkg_brdb_clear_su_lock.reset_lock
Tue 01-Dec-2009 14:54:10.629   OK: Derived FAD_HASH for branch accounting code 2007 is: 96
Tue 01-Dec-2009 14:54:10.629   OK: Updated 1 row in table OPS$BRDB.BRDB_BRANCH_STOCK_UNITS
Tue 01-Dec-2009 14:54:10.630 Completed pkg_brdb_clear_su_lock.update_data
Tue 01-Dec-2009 14:54:10.630 Starting pkg_brdb_clear_su_lock.audit_update
Tue 01-Dec-2009 14:54:10.630   INFO: BRDB INSTANCE NAME: BRDBA1
Tue 01-Dec-2009 14:54:10.630   INFO: UNIX USER: gseem01
Tue 01-Dec-2009 14:54:10.630   INFO: ORACLE USER: SUPPORTTOOLUSER
Tue 01-Dec-2009 14:54:10.630   INFO: CURRENT_JSN: 48 for branch accounting code 2007
Tue 01-Dec-2009 14:54:10.630   OK: Inserted 1 row into OPS$BRDB.BRDB_TXN_CORR_TOOL_JOURNAL
Tue 01-Dec-2009 14:54:10.630 Completed pkg_brdb_clear_su_lock.audit_update
Tue 01-Dec-2009 14:54:10.630  Starting pkg_brdb_clear_su_lock.process_audit
Tue 01-Dec-2009 14:54:10.631  Completed pkg_brdb_clear_su_lock.process_audit
Tue 01-Dec-2009 14:54:10.631 Completed pkg_brdb_clear_su_lock.update_data
01 Dec 14:54:10
01 Dec 14:54:10 unlock.....: Complete
01 Dec 14:54:10
01 Dec 14:54:10 Main.......: Unlocked stock unit DEF for branch code 2007
01 Dec 14:54:10
01 Dec 14:54:10 cleanup.....: Cleaning up ...
01 Dec 14:54:10 cleanup.....:    Lock file /tmp/clear_su_lock.run.lock freed.
01 Dec 14:54:10
01 Dec 14:54:10 cleanup.....: _________________  Processing Complete  _________________
```

5.6.2.6 Diagnostics

The module may fail for one of the following reasons

- The SSC user may not be logged in with their SSC unix login.
- The SSC user's Oracle login may not have been granted the SSC role.
- One or more of the parameters are invalid

5.6.3 BRDB Clear Rollover Lock (clear_ro_lock.sh)

FUJ00234958

This tool allows members of the SSC group to clear branch rollover locks for any given branch accounting code and locking user. Any attempt to run the tool will be audited as well as the actual changes made and running user.

Validation and processing occurs in an Oracle package (OPS$SUPPORTTOOLUSER.PKG_BRDB_CLEAR_RO_LOCK) while the package is initially called by a shell script (clear_ro_lock.sh) on the BRDB server.

The script is located in /app/brdb/trans/support/brdbx015/clear_ro_lock.sh

See DEV/APP/LLD/0203 for more information.

5.6.3.1 Parameters

The tool must be supplied with 2 switches, each with a parameter:

Parameter	Parameter Name	Script Variable Name	Datatype	Valid Input
-b	Branch Accounting Code	BRANCH_CODE	Number	1 – 999999
-u	Lock Holder Username	LOCK_USER	STRING	A [1-15 chr]

5.6.3.2

5.6.3.3 Executing

```
./clear_ro_lock.sh -b <BRANCH_CODE> -u <LOCK_USER>
```

5.6.3.4 Scheduling

This task is scheduled on an ad hoc basis, as and when branch rollover locks need to be unlocked.

5.6.3.5 Audit Records/Logging

Start and finish records are inserted into OPS$BRDB.BRDB_PROCESS_AUDIT with a process_name of 'BRDB_CLEAR_RO_LOCK'.

Each update to OPS$BRDB.BRDB_BRANCH_INFO is audited in OPS$BRDB.BRDB_TXN_CORR_TOOL_JOURNAL.

Any exceptions will be logged to OPS$BRDB.BRDB_OPERATIONAL_EXCEPTIONS with an exception code of 'BRDB_RO_LOCK' and process_name (package name) of 'PKG_BRDB_CLEAR_RO_LOCK'.

The script verbosity level is controlled by BRDB system parameter BRDB_CLEAR_RO_LOCK_DEBUG_LEVEL (parameter_number set to 1 initially), set parameter_number to 2 in order to view the SQL update statement as well as the XML string.

Log files from each run are stored in /app/brdb/trans/support/brdbx015/log.

5.6.3.6 Sample output

This is an example of the output written to standard output and the log file when the module is successful:

```
03 Dec 15:06:55 writelock...: Starting
03 Dec 15:06:55 writelock...:    Lock file
/app/brdb/trans/support/brdbx015/log/clear_ro_lock.run.lock created
03 Dec 15:06:55 writelock...: Complete.
03 Dec 15:06:55
03 Dec 15:06:55 check_env...: Starting
03 Dec 15:06:55 check_env...: Complete.
03 Dec 15:06:55
03 Dec 15:06:55 Main........: Environment OK
03 Dec 15:06:55
03 Dec 15:06:55 view_gvar...: Starting
03 Dec 15:06:55 view_gvar...:    WHOAMI.............  gseem01
03 Dec 15:06:55 view_gvar...:    PROGNAME...........  clear_ro_lock.sh
03 Dec 15:06:55 view_gvar...:    SCRIPT.............  clear_ro_lock
03 Dec 15:06:55 view_gvar...:    THISDIR............  /app/brdb/trans/support/brdbx015
03 Dec 15:06:55 view_gvar...:    LOG_FILE...........
/app/brdb/trans/support/brdbx015/log/clear_ro_lock_20091203_150655.log
03 Dec 15:06:55 view_gvar...:    LOCK_FILE..........
/app/brdb/trans/support/brdbx015/log/clear_ro_lock.run.lock
```

FUJ00234958


```
03 Dec 15:06:55 view_gvar...:    TMP_FILE.............
/app/brdb/trans/support/brdbx015/log/clear_ro_lock.tmp
03 Dec 15:06:55 view_gvar...:    TSTMP................  20091203_150655
03 Dec 15:06:55 view_gvar...:    VERBOSE..............  ON
03 Dec 15:06:55 view_gvar...:    APP..................  BRDB
03 Dec 15:06:55 view_gvar...:    ORACLE_SID...........  BRDBA1
03 Dec 15:06:55 view_gvar...:    BRANCH_CODE..........  2007
03 Dec 15:06:55 view_gvar...:    LOCK_USER............  X
03 Dec 15:06:55 view_gvar...: Complete.
03 Dec 15:06:55
03 Dec 15:06:55 unlock......: Starting
03 Dec 15:06:55
Enabling ssc role
Thu 03-Dec-2009 15:06:55.541 Set DEBUG LEVEL to 1
Thu 03-Dec-2009 15:06:55.541 Starting BRDB_CLEAR_RO_LOCK.update_data
Thu 03-Dec-2009 15:06:55.541  Starting BRDB_CLEAR_RO_LOCK.process_audit
Thu 03-Dec-2009 15:06:55.542  Completed BRDB_CLEAR_RO_LOCK.process_audit
Thu 03-Dec-2009 15:06:55.542   INFO: Parameter p_bac = 2007
Thu 03-Dec-2009 15:06:55.542   INFO: Parameter p_rollover_lock_user = X
Thu 03-Dec-2009 15:06:55.542 Starting BRDB_CLEAR_RO_LOCK.validate_parameters
Thu 03-Dec-2009 15:06:55.542   INFO: Validating branch_accounting_code
Thu 03-Dec-2009 15:06:55.546   OK: Branch Accounting Code: 2007 is open and exists in
OPS$BRDB.BRDB_BRANCH_INFO
Thu 03-Dec-2009 15:06:55.547   OK: Branch Accounting Code: 2007 exists in
OPS$BRDB.BRDB_TXN_CORR_TOOL_CTL
Thu 03-Dec-2009 15:06:55.549   OK: Branch Accounting Code 2007 is locked
Thu 03-Dec-2009 15:06:55.549   OK: Lock on Branch Accounting Code 2007 is locked by X
Thu 03-Dec-2009 15:06:55.549   OK: OPS$SUPPORTTOOLUSER is allowed to update BRDB_BRANCH_INFO
Thu 03-Dec-2009 15:06:55.549   OK: Input parameters validated successfully
Thu 03-Dec-2009 15:06:55.549 Completed BRDB_CLEAR_RO_LOCK.validate_parameters
Thu 03-Dec-2009 15:06:55.549 Starting BRDB_CLEAR_RO_LOCK.reset_lock
Thu 03-Dec-2009 15:06:55.549   OK: Derived FAD_HASH for Branch Accounting Code 2007 is: 96
Thu 03-Dec-2009 15:06:55.552   OK: Updated 1 row in table OPS$BRDB.BRDB_BRANCH_INFO
Thu 03-Dec-2009 15:06:55.552 Completed BRDB_CLEAR_RO_LOCK.update_data
Thu 03-Dec-2009 15:06:55.552 Starting BRDB_CLEAR_RO_LOCK.audit_update
Thu 03-Dec-2009 15:06:55.563   INFO: BRDB INSTANCE NAME: BRDBA1
Thu 03-Dec-2009 15:06:55.563   INFO: UNIX USER: gseem01
Thu 03-Dec-2009 15:06:55.563   INFO: ORACLE USER: SUPPORTTOOLUSER
Thu 03-Dec-2009 15:06:55.563   INFO: CURRENT_JSN: 67 for branch accounting code 2007
Thu 03-Dec-2009 15:06:55.564   OK: Inserted 1 row into OPS$BRDB.BRDB_TXN_CORR_TOOL_JOURNAL
Thu 03-Dec-2009 15:06:55.564 Completed BRDB_CLEAR_RO_LOCK.audit_update
Thu 03-Dec-2009 15:06:55.564  Starting BRDB_CLEAR_RO_LOCK.process_audit
Thu 03-Dec-2009 15:06:55.564  Completed BRDB_CLEAR_RO_LOCK.process_audit
Thu 03-Dec-2009 15:06:55.564 Completed BRDB_CLEAR_RO_LOCK.update_data
03 Dec 15:06:55
03 Dec 15:06:55 unlock......: Complete
03 Dec 15:06:55
03 Dec 15:06:55 Main........: Unlocked branch rollover for branch code 2007
03 Dec 15:06:55
03 Dec 15:06:55 cleanup.....: Cleaning up ...
03 Dec 15:06:55 cleanup.....:    Lock file
/app/brdb/trans/support/brdbx015/log/clear_ro_lock.run.lock freed.
03 Dec 15:06:55
03 Dec 15:06:55 cleanup.....: _________________  Processing Complete  _________________
```

5.6.3.7 Diagnostics

The module may fail for one of the following reasons

- The SSC user may not be logged in with their SSC unix login.
- The SSC user's Oracle login may not have been granted the SSC role.
- One or more of the parameters are invalid

5.6.4 BRDB Update Outstanding Recovery Transaction Tool (upd_rvy_txn.sh)

This tool allows members of the SSC group to mark outstanding recovery transactions as processed in table OPS$BRDB.BRDB_RX_RECOVERY_TRANSACTIONS for any given branch accounting code, node ID, transaction start date and unique sequence number (USN). Any attempt to run the tool will be audited as well as the actual changes made and running user.

FUJ00234958

Validation and processing occurs in an Oracle package (OPS$SUPPORTTOOLUSER.PKG_BRDB_UPD_RVY_TXN) while the package is initially called by a shell script (upd_rvy_txn.sh) on the BRDB server.

The script is located in /app/brdb/trans/support/brdbx015/upd_rvy_txn.sh

See DEV/APP/LLD/0204 for more information.

5.6.4.1 Parameters

The tool must be supplied with 4 switches, each with a parameter:

Parameter	Parameter Name	Script Variable Name	Datatype	Valid Input/Format
-b	Branch Accounting Code	BRANCH_CODE	NUMBER	1-999999
-n	Node ID	NODE_ID	NUMBER	1-99
-t	Transaction Start Date	TXN_STRT_DATE	STRING	DD/MM/YYYY
-u	Unique Sequence Number	SEQ_NUM	NUMBER	> 0

5.6.4.2 Executing

```
./upd_rvy_txn.sh -b <BRANCH_CODE> -n <NODE_ID> -t <DD/MM/YYYY> -u <SEQ_NUM>
```

5.6.4.3 Scheduling

This task is scheduled on an ad hoc basis, as and when recovery transactions need to be marked as processed.

5.6.4.4 Audit Records/Logging

Start and finish records are inserted into OPS$BRDB.BRDB_PROCESS_AUDIT with a process_name of 'BRDB_UPD_RVY_TXN'.

Each update to OPS$BRDB.BRDB_RX_RECOVERY_TRANSACTIONS is audited in OPS$BRDB.BRDB_TXN_CORR_TOOL_JOURNAL.

Any exceptions will be logged to OPS$BRDB.BRDB_OPERATIONAL_EXCEPTIONS with an exception code of 'UPD_RVY_TXN' and process_name (package name) of 'PKG_BRDB_UPD_RVY_TXN'.

The script verbosity level is controlled by BRDB system parameter BRDB_UPD_RVY_TXN_DEBUG_LEVEL (parameter_number set to 1 initially), set parameter_number to 2 in order to view the SQL update statement as well as the XML string.

Log files from each run are stored in /app/brdb/trans/support/brdbx015/log.

5.6.4.5 Sample output

This is an example of the output written to standard output and the log file when the module is successful:

```
02 Dec 14:30:44 writelock...: Starting
02 Dec 14:30:44 writelock...:    Lock file /tmp/upd_rvy_txn.run.lock created
02 Dec 14:30:44 writelock...: Complete.
02 Dec 14:30:44
02 Dec 14:30:44 check_env...: Starting
02 Dec 14:30:44 check_env...: Complete.
02 Dec 14:30:44
02 Dec 14:30:44 Main.......: Environment OK
02 Dec 14:30:44
02 Dec 14:30:44 view_gvar...: Starting
02 Dec 14:30:44 view_gvar...:    WHOAMI.............  gseem01
02 Dec 14:30:44 view_gvar...:    PROGNAME...........  upd_rvy_txn.sh
02 Dec 14:30:44 view_gvar...:    SCRIPT.............  upd_rvy_txn
02 Dec 14:30:44 view_gvar...:    THISDIR............  /app/brdb/trans/support/brdbx015
02 Dec 14:30:44 view_gvar...:    LOG_FILE...........
/app/brdb/trans/support/brdbx015/log/upd_rvy_txn_20091202_143044.log
02 Dec 14:30:44 view_gvar...:    LOCK_FILE..........  /tmp/upd_rvy_txn.run.lock
02 Dec 14:30:44 view_gvar...:    TSTMP..............  20091202_143044
02 Dec 14:30:44 view_gvar...:    VERBOSE............  ON
02 Dec 14:30:44 view_gvar...:    APP................  BRDB
```

FUJ00234958


```
02 Dec 14:30:44 view_gvar...:    ORACLE_SID............  BRDBA1
02 Dec 14:30:44 view_gvar...:    BRANCH_CODE...........  2007
02 Dec 14:30:44 view_gvar...:    NODE_ID...............  1
02 Dec 14:30:44 view_gvar...:    TXN_STRT_DATE.........  06/10/2009
02 Dec 14:30:44 view_gvar...:    USN...................  123
02 Dec 14:30:44 view_gvar...: Complete.
02 Dec 14:30:44
02 Dec 14:30:44 unlock......: Starting
02 Dec 14:30:44
Enabling ssc role
Wed 02-Dec-2009 14:30:44.809 Set DEBUG LEVEL to 1
Wed 02-Dec-2009 14:30:44.809 Starting pkg_brdb_upd_rvy_txn.update_data
Wed 02-Dec-2009 14:30:44.809  Starting pkg_brdb_clear_su_lock.process_audit
Wed 02-Dec-2009 14:30:44.810  Completed pkg_brdb_clear_su_lock.process_audit
Wed 02-Dec-2009 14:30:44.810   INFO: Parameter p_bac = 2007
Wed 02-Dec-2009 14:30:44.810   INFO: Parameter p_node_id = 1
Wed 02-Dec-2009 14:30:44.810   INFO: Parameter p_txn_strt_date: 06-OCT-2009
Wed 02-Dec-2009 14:30:44.810   INFO: Parameter p_usn: 123
Wed 02-Dec-2009 14:30:44.810 Starting pkg_brdb_upd_rvy_txn.validate_parameters
Wed 02-Dec-2009 14:30:44.810   INFO: Validating branch_accounting_code
Wed 02-Dec-2009 14:30:44.812   OK: Branch Accounting Code: 2007 is open and exists in
OPS$BRDB.BRDB_BRANCH_INFO
Wed 02-Dec-2009 14:30:44.813   OK: Branch Accounting Code: 2007 exists in
OPS$BRDB.BRDB_TXN_CORR_TOOL_CTL
Wed 02-Dec-2009 14:30:44.813   OK: USN 123 is outstanding for branch accounting code 2007
Wed 02-Dec-2009 14:30:44.813   OK: OPS$SUPPORTTOOLUSER is allowed to update
BRDB_RX_RECOVERY_TRANSACTIONS
Wed 02-Dec-2009 14:30:44.813   OK: Input parameters validated successfully
Wed 02-Dec-2009 14:30:44.813 Completed pkg_brdb_upd_rvy_txn.validate_parameters
Wed 02-Dec-2009 14:30:44.813 Starting pkg_brdb_upd_rvy_txn.reset_outstanding
Wed 02-Dec-2009 14:30:44.813   OK: Derived FAD_HASH for branch accounting code 2007 is: 96
Wed 02-Dec-2009 14:30:44.814   OK: Updated 1 row in table OPS$BRDB.BRDB_RX_RECOVERY_TRANSACTIONS
Wed 02-Dec-2009 14:30:44.814 Completed pkg_brdb_rvy_txn.update_data
Wed 02-Dec-2009 14:30:44.814 Starting pkg_brdb_clear_su_lock.audit_update
Wed 02-Dec-2009 14:30:44.814   INFO: BRDB INSTANCE NAME: BRDBA1
Wed 02-Dec-2009 14:30:44.814   INFO: UNIX USER: gseem01
Wed 02-Dec-2009 14:30:44.814   INFO: ORACLE USER: SUPPORTTOOLUSER
Wed 02-Dec-2009 14:30:44.814   INFO: CURRENT_JSN: 54 for branch accounting code 2007
Wed 02-Dec-2009 14:30:44.815   OK: Inserted 1 row into OPS$BRDB.BRDB_TXN_CORR_TOOL_JOURNAL
Wed 02-Dec-2009 14:30:44.815 Completed pkg_brdb_clear_su_lock.audit_update
Wed 02-Dec-2009 14:30:44.815  Starting pkg_brdb_clear_su_lock.process_audit
Wed 02-Dec-2009 14:30:44.815  Completed pkg_brdb_clear_su_lock.process_audit
Wed 02-Dec-2009 14:30:44.815 Completed pkg_brdb_clear_su_lock.update_data
02 Dec 14:30:44
02 Dec 14:30:44 unlock......: Complete
02 Dec 14:30:44
02 Dec 14:30:44 Main........: Unlocked stock unit  for branch code 2007
02 Dec 14:30:44
02 Dec 14:30:44 cleanup.....: Cleaning up ...
02 Dec 14:30:44 cleanup.....:    Lock file /tmp/upd_rvy_txn.run.lock freed.
02 Dec 14:30:44
02 Dec 14:30:44 cleanup.....: _________________  Processing Complete  _________________
```

5.6.4.6 Diagnostics

The module may fail for one of the following reasons

- The SSC user may not be logged in with their SSC unix login.
- The SSC user's Oracle login may not have been granted the SSC role.
- One or more of the parameters are invalid

5.6.5 BRDB Branch & Stock Unit Financial Year Update (upd_ro_fad_fyr.sql)

This SQL script allows members of the SSC group to update current and next financial year for a given fad in tables OPS$BRDB.BRDB_BRANCH_INFO and OPS$BRDB.BRDB_BRANCH_STOCK_UNITS.

The script is located in /app/brdb/trans/support/brdbx015/ upd_ro_fad_fyr.sql

5.6.5.1 Parameters

FUJ00234958

The SQL script interactively prompts for the following data items:

Parameter Name	Script Variable Name	Datatype
Branch Accounting Code	ABAC	NUMBER
Financial Year	AFYR	NUMBER

5.6.5.2 To Execute

- Login to a BDB node
- Invoke sqlplus
  - sqlplus /
- Invoke the update script
  - @/app/brdb/trans/support/brdbx015/upd_ro_fad_fyr.sql
- Enter the desired branch accounting code & financial year when prompted

5.6.5.3 Scheduling

This task is scheduled on an ad hoc basis, as and when a branch requires processing.

5.6.5.4 Audit Records/Logging

Invocation of the update script will be controlled by MSC.

5.6.5.5 Sample output

This is an example of the output written to standard output:

```
SQL> @upd_ro_fad_fyr.sql
Starting script upd_ro_fad_fyr.sql...
Enter Branch accounting code  ==> 100140
Enter required financial year ==> 2013
+******************************+
|BRDB_BRANCH_INFO before UPDATE|
+******************************+

OUTLET_NAME                    CURR_TP_FYR NEXT_TP_FYR    CURR_TP STATUS ROLLOVER
------------------------------ ----------- ----------- ---------- ------ --------
                                      2008        2009        101 Open   N

+************************************+
|BRDB_BRANCH_STOCK_UNITS before UPDATE|
**************************************+

SU  CURR_TP_FYR    CURR_TP    CURR_BP ROLLOVER IS_INACTIVE IS_DELETED
--- ----------- ---------- ---------- -------- ----------- ----------
AA         2008          6          1 N        N           N
BB         2008          6          1 N        N           N
BDC        2008          6          1 N        N           N
BM         2008          6          1 N        N           N
```

FUJ00234958


```
CC          2008          6          2 N        N           N
DEF         2008          6          1 N        Y           N
EE          2008          6          1 N        N           N
FF          2008          6          1 N        N           N
GG          2008          6          1 N        N           N
HH          2008          6          1 N        N           N
JJ          2008          6          1 N        N           N

+*****************+
|Execution Output|
+*****************+
Updating BRDB_BRANCH_INFO for branch 100140
Updated 1 rows in BRDB_BRANCH_INFO
Updated 11 rows in BRDB_BRANCH_STOCK_UNITS

**!!  Execution Complete !!**

+*******************************+
|BRDB_BRANCH_INFO after UPDATE|
+*******************************+

OUTLET_NAME                    CURR_TP_FYR NEXT_TP_FYR    CURR_TP STATUS ROLLOVER
------------------------------ ----------- ----------- ---------- ------ --------
                                       2013        2014        101 Open   N

+************************************+
|BRDB_BRANCH_STOCK_UNITS after UPDATE|
+************************************+

SU  CURR_TP_FYR    CURR_TP    CURR_BP ROLLOVER IS_INACTIVE IS_DELETED
--- ----------- ---------- ---------- -------- ----------- ----------
AA          2013          6          1 N        N           N
BB          2013          6          1 N        N           N
BDC         2013          6          1 N        N           N
BM          2013          6          1 N        N           N
CC          2013          6          2 N        N           N
DEF         2013          6          1 N        Y           N
EE          2013          6          1 N        N           N
FF          2013          6          1 N        N           N
GG          2013          6          1 N        N           N
HH          2013          6          1 N        N           N
JJ          2013          6          1 N        N           N

COMMIT complete.
```

FUJ00234958

5.6.5.6 Diagnostics

The module may fail for one of the following reasons

- The SSC user may not be logged in with their SSC unix login.
- The SSC user's Oracle login may not have been granted the SSC role.

5.7 BRDBC004 Archival/Purge Logic

The replication of DELETE SQL statements to BRSS is controlled by a flag named 'ALLOW_REPLICATION', a column in the table BRDB_ARCHIVED_TABLES. A value of 'N' against a particular table indicates that DELETES against that table will not get replicated across to BRSS by Oracle OGG and a 'Y' indicates otherwise.

BRDBC004 uses this flag to allow or block the replication of DELETEs against a particular table, accordingly.

Most archive/purge metadata records are set to a default value of 'N' (there are some tables which have this flag set to 'Y').

This change to Branch Database archive metadata was made, firstly because local maintenance of purging OPS$BRDB tables in BRSS was required. Hence, archive metadata for all OPS$BRDB tables that were not already managed locally in BRSS were added to BRSS_ARCHIVED_TABLES in order to enable BRSSC004 to purge the respective local tables based on corresponding retention periods. Secondly, making the necessary changes to the archive processes on both BRDB and BRSS became critical as the large volume of transaction records being purged overnight in BRDB caused load stress on Oracle Streams (not necessarily relevant to OGG).

An associated benefit of making this fix is that all data records in BRDB, which are replicated across to BRSS can be retained locally in BRSS with differing retention periods to that of BRDB without having to manually create OGG mechanisims for every transaction table in BRDB that needed a higher retention period in BRSS.

As a result of this enhancement, any new table introduced into the Branch Database, must have the requisite 'archive metadata' added to *both* BRDB_ARCHIVED_TABLES (in BRDB) and BRSS_ARCHIVED_TABLES (in BRSS) in order for BRDBC004 and BRSSC004 to perform their respective purge functions effectively.

FUJ00234958

5.8 BRDB Software Updates/Installation

If a total service outage is possible due to the application of software to BRDB (whether that software is an Oracle patch, proprietary code for BRDB, etc.) then the following should be observed:

- Ensure the delivery handover notes clearly state a system outage is required
- CS should communicate the date/time of the planned service outage to POL (and hence the branches)
- Access to the BRDB database should be controlled by disabling/re-enabling access via the ACE.
- The OSR instances may need to be restarted if there are changes that have a direct impact on the OSRs (for example a change to BAL SQL statements)
- Examine whether any changes affect the various daemon type processes. Any impacts may result in relevant schedules being stopped early or held until after the application of the change.

5.9 Querying/Updating BRDB/BRSS during the online day

Any database query that could be considered to be 'large' should, in general, be kept outside the accepted online day operating hours.

The following is a guide to which queries (SELECTs, UPDATEs, DELETEs) might turn out to be 'large' or over-utilise resource unnecessarily (and should therefore *not* be executed): -

- The query involves more than one date partition (or does not even have a date restriction in the WHERE clause) as per those tables present in BRDB_PARTITIONED_TABLES
- The query features a function around the partitioned key column in the WHERE clause - preventing Oracle from utilising partition pruning
- Transactions that run for more than 5 minutes or consist of more than 500,000 rows may stress the OGG implementation, with the result that OGG Replicat fails to keep BRSS up to date
- Any query which that does not utilise the localisation of data to the instance from which the query is executed. In other words, if a set of data relating to a branch whose natural/defined instance is BRDB2 (for example according to the defined fad_hash-mappings) should not be queried from BRDB3. The localisation of every query should always be a consideration!

FUJ00234958

5.10 BRSS_GEN_REP/GREPX00[1|2] Empty File Recovery

This section details the recovery steps involved in recreating the necessary files created by the BRSS_GEN_REP group of TWS jobs. These jobs create csv files which are used for reporting purposes. The BRSS_GEN_REP job consists of the following sub tasks, executed in the following order: -

GREPX001
SLT_TO_5MIN_STATS
SETTLEMENT_TO_5MIN_STATS
NRT_TO_5MIN_STATS
5MIN_TO_HOURLY_STATS
HOURLY_TO_DAILY_STATS
GREPX002

It is important to know this order as it is the order in which the scripts are to be run. The "hourly" and "daily" jobs aggregate the "5 min" data and so therefore must follow them. The final job creates files, based on the aggregated data.

The embedded script that follows is a script which was used in a mumber of MSC's (System Change Request) in LIVE in order to generate the required files. The instructions which follow are summarisations of the steps followed within the script and are detailed here for purposes of providing an overview of the tasks/steps.

MSC - LIVE 043J0319240 (Generate CapMngmnt Reporting Data).sql

Step 1: Create the following temporary tables (schema: OPS$BRSS)

```
temp_hngx_raw_slt_stats
temp_capmgmt_5min_stats
temp_capmgmt_hourly_stats
temp_capmgmt_daily_stats
```

Step 2: Insert relevant reporting data into temporary tables *in the following type-order*: -

SLT_TO_5MIN_STATS
SETTLEMENT_TO_5MIN_STATS
NRT_TO_5MIN_STATS
5MIN_TO_HOURLY_STATS
HOURLY_TO_DAILY_STATS

Step 3: Generate new CSV files (into directory **`/app/brss/trans/support/reportoutput`**), based on inserted and aggregated data: -

5_MIN: `CapMgmt_5Min_Stats_msc043J0319240.csv`
HOURLY: `CapMgmt_Hourly_Stats_msc043J0319240.csv`
DAILY: `CapMgmt_Daily_Stats_msc043J0319240.csv`

Step 4: Rename the files generated in Step 3. One would need to use the *reporting_date + 1* when renaming the files; so if the date used in Step 1 (see embedded script) is 20120116, then use '0117: -

```
chown brssbth1:pathway *msc043J0319240.csv
mv CapMgmt_5Min_Stats_msc_043J0319240.csv CapMgmt_5Min_Stats_20120117.csv
mv CapMgmt_5Min_Stats_msc043J0319240.csv CapMgmt_5Min_Stats_20120117.csv
mv CapMgmt_Hourly_Stats_msc043J0319240.csv CapMgmt_Hourly_Stats_20120117.csv
mv CapMgmt_Daily_Stats_msc043J0319240.csv
CapMgmt_Daily_Stats_msc_20120117.csv
```

Step 5 (regression): There is a set of straightforward regression instructions (within the embedded script) and are in essence simply just commands for *dropping* the following tables: -

```
temp_hngx_raw_slt_stats
temp_capmgmt_5min_stats
temp_capmgmt_hourly_stats
```

FUJ00234958

temp_capmgmt_daily_stats

FUJ00234958

6 Appendix A – Standby Database

The build of and theory surrounding the BRDB Standby database (SBRDB) is detailed extensively in the Standby Database Low Level Design [DEV/APP/LLD/0152]. For added clarity, Section 6.7 has been added to aid in support activities relating to Standby Re-instantiation from *BDB-to-BDS* as originally configured. Note that Section 6.7 differs fundamentally from Section 6.4 in that it is a re-instantiation of the *original* configuration and not an *initial* instantiation of a failed-over BDS configuration, i.e. (BDS-to_BDB),

This section details the failover procedures in changing the role of a database, in our case BRDB or SBRDB. The method described in sections 6.2 and 6.3, is known as *complete failover* and must be executed as described in order to ensure no data loss.

It is very important to note – as detailed in the Branch Database High Level Design [DES/APP/HLD/0020] – that the changing of roles of the Standby to Primary is utterly *irreversible*! The term "switchover", which is a temporary role change is *not* supported. Section 6.5 therefore, details the temporary opening of the Standby Database for read-only purposes.

Without the broker, you perform role transitions by first determining if a role transition is necessary and then issuing a series of SQL statements (as described later in this section). After failover to a physical standby database, the original primary database must be re-enabled to act as a standby database for the new primary database.

Note: The procedure described in section 6.1 is the recommended course of action. Section 6.3 has been provided for, in the event that the Data Guard Broker is unavailable.

6.1 Shutdown Goldengate

Shutdown Goldengate as per the following

Step	Description	Server Execution
1	Login to BDB as 'oggadmin' user	
2	Invoke ggsci	`$OGG_HOME/ggsci`

3	Stop extract	`GGSCI> stop E11BDB`
		
		`Sending STOP request to EXTRACT E11BDB ...`
		`Request processed.`

4	Issue info all repeatedly until E11BDB is shown as STOPPED	`GGSCI> info all`
		
		`Program Status Group Lag at Chkpt Time Since Chkpt`
		
		`MANAGER RUNNING`
		`EXTRACT STOPPED E11BDB 00:00:05 00:01:58`

FUJ00234958

		EXTRACT RUNNING P11BDB 00:00:00 00:00:06
5	Wait until all lag disappears against data pump P11BDB by issuing:	GGSCI> info all Program Status Group Lag at Chkpt Time Since Chkpt MANAGER RUNNING EXTRACT STOPPED E11BDB 00:00:05 00:01:58 EXTRACT RUNNING P11BDB 00:00:00 00:00:06
6	Stop datapump once lag is gone	GGSCI> stop P11BDB Sending STOP request to EXTRACT P11BDB ... Request processed.
7	Ensure data pump P11BDB is stopped by issuing:	GGSCI> info all Program Status Group Lag at Chkpt Time Since Chkpt MANAGER RUNNING EXTRACT STOPPED E11BDB 00:00:05 00:05:32 EXTRACT STOPPED P11BDB 00:00:00 00:02:25
8	Stop all OGG processes via CRS (logon as 'oracle' user):	oracle> **crsctl stop res brdb.oggadmin.oggapp** oracle> **crsctl stop resource dbfs_mount**
9	Login to BRS as 'oggadmin' user	
10	Invoke ggsci	$OGG_HOME/ggsci

FUJ00234958

11	Ensure there is no lag associated with replicat R11BRS by issuing	GGSCI> info all Program Status Group Lag at Chkpt Time Since Chkpt MANAGER RUNNING REPLICAT RUNNING R11BRS 00:00:00 00:00:03
12	Wait until lag is zero before continuing with the rest of the failover procedure	

6.2 Oracle Data Guard Broker (DGMGRL) Failover

The broker simplifies failovers by allowing you to invoke them using a single command in the DGMGRL command-line interface, e.g. a manual failover. The method described in this manual procedure is known as complete failover and must be executed as described in order to ensure no data loss.

Step	Description	Server Execution
Assumptions	i. User is logged onto the Standby Database Server as ***oracle***. . ii. After determining that there is no possibility of recovering the primary database in a timely manner, *ensure that the primary database is shut down* (if not already) and then begin the failover operation.	
1.	*[Who: DBA]* Logon to DGMGRL command-line interface. <sys password> is always required as this is a "sysdba" connection. This will connect you via the Data Guard Broker to the Standby Database.	**$> . oraenv** [now type in SBRDB1] **$> dgmgrl** **DGMGRL> CONNECT sys/<sys password>**
2.	*[Who: DBA]* On the target standby database, issue the FAILOVER command to invoke a complete failover, specifying the name of the standby database that you want to change into the primary role.	**DGMGRL> FAILOVER TO 'SBRDB';**

FUJ00234958

Step	Description	Server Execution
	How the Broker Performs a Complete Failover Operation Once you start a complete failover, the broker: i. Checks to see if the primary database is still available and, if so, issues a warning message asking whether you want to continue with the failover operation. ii. Verifies that the target standby database is enabled. If the database is not enabled, you will not be able to perform a failover to this database. The broker shuts down all RAC instances except the apply instance assuming they are up. This is unlikely in Branch Standby Database as only one node is configured to be active at any one time. iii. Waits for the target standby database to finish applying any remaining archived redo logs before stopping Redo Apply or SQL Apply. iv. Transitions the target standby database into the primary database role by opening the new primary database SBRDB, in read/write mode.	

3.	*[Who: DBA]*
	Issue the SHOW CONFIGURATION command to verify the failover.
		`DGMGRL> SHOW CONFIGURATION;`
		
		You should see …
		
		`Configuration - BRDB_DATAGUARD_CFG`
		
		`Protection Mode: MaxPerformance`
		`Databases:`
		`SBRDB - Primary database`
		`BRDB - Physical standby database (disabled)`
		`ORA-16661: the standby database needs to be reinstated`
		
		`Fast-Start Failover: DISABLED`
		
		`Configuration Status:`
		`SUCCESS`

4.	*[Who: DBA]* Issue the SHOW DATABASE command to see that the former (failed) primary database was disabled by the broker as a consequence of the failover. Remember, it must be re-enabled.	`DGMGRL> SHOW DATABASE 'BRDB';` *Database - BRDB* *Role: PHYSICAL STANDBY* *Intended State: APPLY-ON* *Transport Lag: (unknown)* *Apply Lag: (unknown)* *Apply Rate: (unknown)* *Real Time Query: OFF* *Instance(s):* *BRDB1* *BRDB2* *Database Status:* *ORA-16661: the standby database needs to be reinstated*

5.	*[Who: DBA]*
	Check that all the indexes – database wide – are available for use.
		`SQL> SELECT owner, index_name`
		`FROM dba_indexes`
		`WHERE status = 'UNUSABLE';`
		
		`SQL> ALTER INDEX <OWNER>.<index> REBUILD ONLINE [ PARALLEL <# CPU's> ];`

FUJ00234958

Step	Description	Server Execution
	If any indexes are marked as 'UNUSABLE' they need to be rebuilt. See example to the right of this cell.	

6a.	*[Who: DBA]*
	Depending on the timing of the failover to Standby, the other SBRDB instances (nodes 2-4) will be started in nomount mode..
	• Ensure orapwd file is consistent on all BDS servers Once you're able to log on as ***oracle***, bring up the remaining database instances starting with SBRDB2, e.g.
	`oracle:> . oraenv`
	ORACLE_SID = [SBRDB2] ?
	`oracle:> sqlplus / as sysdba`
	`SQL:> alter database mount;`
	`SQL:> alter database open;`
	`SQL:exit`	

6b.	*[Who: DBA]*
	The following SBRDB database initialisation parameters need to be checked and if not correct, need to be set correctly after the Standby database (SBRDB) has been successfully transitioned from Standby to it's new role as Primary.
	
	This information can be double-checked by comparing the initialisation parameters from Primary with those of Standby. The comparison can be done against pfiles generated from both nodes. Follow Step [6b.] to accomplish this.
	
	The following parameters should be checked and the values shown below should be reflected in SBRDB for all new instances. This is done by executing a statement of the form "`ALTER SYSTEM SET <parameter>=<value> SCOPE=<scope> SID='*';`".
	
	Parameter / Future Value / Likely Current Value
	`audit_trail` — `DB` — `NONE`
	`cluster_database_instances` — `4` — `1`
	`control_file_record_keep_time` — `21` — *`NULL`*
	`instance_number` — `[1] to [4]` — `<See action 1 below>`
	`instance_name` — *`NULL`* — `<See action 2 below>`
	`local_listener` — `LISTENER_<node>` — `<See action 3 below>`
	`log_archive_dest_3` — *`NULL`* — `'LOCATION=/archredo/<DB> OPTIONAL'`
	`log_archive_dest_state_3` — *`NULL`* — `'ENABLE'`
	`sessions` — `2205` — `610`
	`thread` — `[1] to [4]` — `<See action 4 below>`
	
	[1] An "`ALTER SYSTEM ... SID='SBRDB2'`" statement required on ***each*** instance, e.g. `instance_number=2` for node 2, `3` for node 3, et cetera.
	
	[2] An "`ALTER SYSTEM ... SID='SBRDB2'`" statement required on ***each*** instance, e.g. `instance_name='SBRDB2'` for node 2, `'SBRDB3'` for node 3, et cetera.
	
	[3] An "`ALTER SYSTEM ... SID='SBRDB2'`" statement required on ***each*** instance, e.g. `local_listener='LISTENER_<node002>'` for node 2, `'LISTENER_<node003>'` for node 3, etc.
	
	[4] An "`ALTER SYSTEM ... SID='SBRDB2'`" statement required on ***each*** instance, e.g. `thread=2` for node 2, `3` for node 3, et cetera.	

FUJ00234958

Step	Description	Server Execution
6b cont..	*[Who: DBA]* In the same way as the parameters were added above, the following parameters delivered in various releases from *Release 0108 up to and including Release 0500*, should also be added: - Parameter / Future Value / Likely Current Value *"Red Alert" parameter changes* `_kghdsidx_count` 2 NULL `_library_cache_advice` FALSE NULL `_object_statistics` FALSE NULL `db_cache_advice` OFF ON `event='14532 trace name context forever, level 1'` Currently unset, simply set the value on any instance. `pga_aggregate_target` 4294967296 5368709120 *Oracle Resource Manager parameter changes* `resource_limit` TRUE NULL `resource_manager_plan` HNGX_PLAN NULL `_low_server_threshold` 16 7 or NULL `_high_server_threshold` 32 12 or NULL `shared_pool_size` 4311744512 2256M `parallel_max_servers` 64 NULL *PAF parameter changes* `sga_target` 24534581248 21474836480 `db_keep_cache_size` 5637144576 NULL *Other Oracle Bug parameter changes* `memory_broker_stat_interval` 60 NULL [	

6c.	*[Who: DBA]*
	Create a text file "copy" of the current spfile (server parameter file) on both the Primary (BRDB) and the Standby SBRDB nodes.
	
	Copy the files to a location where they can be compared and compare them either by using the UNIX diff command or a Windows compare tool, e.g.
		`. oraenv`
		[Now type BRDB1 (on node1)]
		`sqlplus '/as sysdba'`
		`SQL> CREATE PFILE='<some_dir>/pfile<DATABASE>.ora' FROM SPFILE;`
		[Now do the same for SBRDB on the Standby node.]
		`diff pfileBRDB.ora pfileSBRDB.ora`

FUJ00234958

Step	Description	Server Execution

7.	*[Who: DBA]*
	After failover, the "new" Primary database cluster (lprpbds20201 - 4) and database, SBRDB, must accept connections from all applications without changing any application connection properties. Therefore, in order to accomplish this, a new database service must be created for BRDB.
	
	On the ***first*** node: -
	The service should already be enabled, so all that needs to be done is to start the service.
	If starting the service is unsuccessful for some reason, then try enabling the service.
	Once again, after enabling the service, try starting the service again.
	With the service having been correctly created, check the CRS status to see the state of the services as well as the listener control utility.
		Syntax
		`srvctl add service`
		`-d <db_unique_name>`
		`-s <service_name>`
		`-r <preferred_list>`
		
		Command
		`srvctl add service -d SBRDB -s BRDB -r SBRDB1,SBRDB2,SBRDB3,SBRDB4`
		
		`srvctl start service -d SBRDB -s BRDB`
		
		`srvctl enable service -d SBRDB -s BRDB`
		
		[A.] `srvctl status database -d SBRDB`
		[B.] . oraenv
		ORACLE_SID = [SBRDB1] ? +ASM1
		`lsnrctl status`

The correct output seen, should be similar to the following: -

[A.]

```
Instance SBRDB1 is running on node lsdpbds501
Instance SBRDB2 is running on node lsdpbds502
```

[B.]

```
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 30-JUL-2014 19:41:01

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.4.0 - Production
Start Date                18-JUL-2014 07:21:25
Uptime                    12 days 12 hr. 19 min. 36 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/11.2.0/grid/network/admin/listener.ora
```

FUJ00234958

Step	Description	Server Execution
		Listener Log File /u01/app/11.2.0/grid/log/diag/tnslsnr/lsdpbds501/listener/alert/log.xml Listening Endpoints Summary... (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER))) (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.23.207.91)(PORT=1529))) (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.23.207.93)(PORT=1529))) Services Summary... Service "+ASM" has 1 instance(s). Instance "+ASM1", status READY, has 1 handler(s) for this service... Service "BRDB" has 1 instance(s). Instance "SBRDB1", status READY, has 1 handler(s) for this service... Service "SBRDB" has 1 instance(s). Instance "SBRDB1", status READY, has 1 handler(s) for this service... Service "SBRDBXDB" has 1 instance(s). Instance "SBRDB1", status READY, has 1 handler(s) for this service... Service "SBRDB_DGB" has 1 instance(s). Instance "SBRDB1", status READY, has 1 handler(s) for this service... The command completed successfully …
8.	• Stop and restart the dbfs resource. Login as oracle user oracle:> crsctl stop resource dbfs_mount oracle:> crsctl start resource dbfs_mount • Start Goldengate services: • Login as Unix user oggadmin. oggadmin:>. oraenv ORACLE_SID =SBRDB1 • $oggadmin> $OGG_HOME/ggsci • GGSCI> DELETE CREDENTIALSTORE • GGSCI> ADD CREDENTIALSTORE • GGSCI> INFO CREDENTIALSTORE • GGSCI> ALTER CREDENTIALSTORE ADD USER ops$oggadmin , PASSWORD some-secure-password, ALIAS LOGINALIAS • GGSCI> INFO CREDENTIALSTORE • GGSCI> exit • $oggadmin:> sqlplus / as sysdba • SQL> ALTER USER ops$oggadmin identified by some-secure-password; **$oggadmin:>**$OGG_HOME/ggsci GGSCI (lsdpbds201) 1> info all GGSCI (lsdpbds201) 1> start MANAGER GGSCI (lsdpbds201) 1> start E11BDB GGSCI (lsdpbds201) 1> start P11BDB	

FUJ00234958

Step	Description	Server Execution

```
GGSCI (lsdpbds201) 1> info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     E11BD       00:00:06      00:00:07
EXTRACT     RUNNING     P11BD       00:00:00      00:00:09
```

FUJ00234958

Step	Description	Server Execution

8.	*[Who: UNIX ADMIN]*
	The Primary database cluster (lprpbdb20201 – 4) after failover will be the former Standby database cluster (lprpbds20201 - 4), so as a result of the BRDB failover to the BDS cluster, it will be necessary to re-configure DNS to seamlessly make this change, thereby allowing all applications that reference the Primary database cluster to instead reference the Standby database cluster.
	In order to accomplish this, the following should be followed: -
	[1.] Update ACD001 to change the PBDB00X-VIP alias to point to associated BDS servers, e.g. (lprpbds201 - 4)
	[2.] Flush DNS cache on all Linux DNS servers (DNP and DNS)
	`/usr/sbin/rndc flush`
	[3.] Clear the DNS cache on all servers that address BDB on VIP alias
	`/usr/sbin/nscd --invalidate=hosts`
	[4.] Once the DNS switch is complete perform a set of 'ping' sanity checks to ensure that client applications (DAT, BAL/OSR, etc) are referencing the "new" Primary server IP addresses.
	[5.] In addition to [4.] above, perform a quick test to ensure that one is connecting to the correct database and that the newly created service (Step 7. above) is accepting connections.
	`sqlplus lvbaluser1/<lvbaluser1 password>@BRDB`
	[6.] To allow TWS to access and run schedules on the new Primary nodes: -
	Update `TWS.cpu` to point AGBRDB[1234] to PBDS20[1234]
	Update DNS to point PBDB20[1234] to LPRPBDS20[1234]
	
	WARNING
	Any subsequent DNS deliveries may reset the IP addresses back to the original BRDB1..4 servers. It may be necessary to raise an OCP along with a DNS delivery to set the IP addresses back to the fail-over servers.
	
	** Disable Housekeeping and RMAN backup jobs, if running TWS schedule on BDS servers after Failover.	

9.	*[Who: DBA or UNIX ADMIN]* WARNING On the new Primary server, e.g. the BDS Cluster (***on each node, e.g. 1 – 4***), the cron jobs which run on these nodes in the absence of any TWS schedules need to be stopped. Edit the crontab. Once the crontab has loaded (output should reflect schedule shown below). Use ***vi*** commands to add a "#" in front of every line where one does not exist. Then save and quit the file. Note: *The crontab may change over time and may not need editing. The principle remains that the new primary site, should only have the official scheduled backups being run against it.*	As the ***oracle*** and grid user … `$> crontab -e`

FUJ00234958

Step	Description	Server Execution

```
# HouseKeeping
#20 9 * * * /usr/local/bin/HousekeepOrafiles.sh -d SBRDB cron.1.std.out 2>&1
#
# RMANBackup
#5 1 *  * * /usr/local/bin/SBRDBBackup.sh
```

lxxxbds201 oracle user

 # The following crons only apply during standby failover and should be commented out in normal running

0 3 * * 1,4 /usr/local/bin/RMANBackup.sh -v -d SBRDB -l 0

0 3 * * 0,2,3,5,6 /usr/local/bin/RMANBackup.sh -v -d SBRDB -l 1

0 6 * * * /usr/local/bin/HousekeepWrapper.sh SBRDB >cron.1.std.out 2>&1

lxxxbds201 grid user

15 6 * * * /usr/local/bin/HousekeepWrapper.sh +ASM > cron.std.out 2>&1

lxxxbds202-204 oracle user

0 6 * * * /usr/local/bin/HousekeepWrapper.sh SBRDB >cron.1.std.out 2>&1

lxxxbds202-204 grid user

15 6 * * * /usr/local/bin/HousekeepWrapper.sh +ASM > cron.std.out 2>&1

lxxxbdb201 oracle user

The following crons only apply during standby failover and should be commented out in normal running

0 6 * * * /usr/local/bin/HousekeepWrapper.sh BRDB r >cron.1.std.out 2>&1

lxxxbdb201 grid user

The following crons only apply during standby failover and should be commented out in normal running

15 6 * * * /usr/local/bin/HousekeepWrapper.sh +ASM > cron.std.out 2>&1

lxxxbdb202-204 oracle user

The following crons only apply during standby failover and should be commented out in normal running

0 6 * * * /usr/local/bin/HousekeepWrapper.sh BRDB >cron.1.std.out 2>&1

lxxxbdb202-204 grid user

The following crons only apply during standby failover and should be commented out in normal running

15 6 * * * /usr/local/bin/HousekeepWrapper.sh +ASM > cron.std.out 2>&1

FUJ00234958

Step	Description	Server Execution
10	Reinstall Oracle `FAN Event handler for all` the BDS Cluster (***on each node, e.g. 1 – 4***), As the ***root*** user … **$> mv -f /app_sw/brdb/sh/fan_event_handler.ksh /u01/app/11.2.0/grid/racg/usrco** **$> chown grid:oinstall /u01/app/11.2.0/grid/racg/usrco/fan_event_handler.ksh** **$> chmod 550 /u01/app/11.2.0/grid/racg/usrco/fan_event_handler.ksh** Remove Oracle `FAN Event handler for all` the BDB Cluster (***on each node, e.g. 1 – 4***), As the ***root*** user … **$> mv -f /u01/app/11.2.0/grid/racg/usrco/fan_event_handler.ksh /app_sw/_brdb/sh/**	
11.	Edit tnsnames.ora on BDS: `Entries with BRDB1,BRDB2,BRDB3 and BRDB4, change instance names to SBRDB{n}` **BRDB{n}** = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = pbds201-vip) (PORT = 1529)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = BRDB) (INSTANCE_NAME = **SBRDB{n}**)))	
12.	To maintain a viable disaster-recovery solution in the event of another disaster you must reinstate the original primary database to act as a standby database in the new configuration. This can be accomplished by following the notes in Section 6.4, as one must re-create the primary database from a copy of the new primary database.	
	Manual *Complete* Failover through **DGMGRL** is complete.	

Table 3: Data Guard Failover Procedure.

FUJ00234958

6.3 SQL*Plus Failover

Perform role transitions by first determining if a role transition is necessary and then issuing the following series of SQL statements. The method described in this procedure is also known as complete failover and must be executed as described in order to ensure no data loss.

Step	Description	Server Execution
Assumptions	i. User is logged onto the Standby Database Server as ***oracle***. ii. After determining that there is no possibility of recovering the primary database in a timely manner, ensure that the primary database is shut down (if not already) and any other standby database instances that may be started, then begin the failover operation.	

1.	*[Who: DBA]*
	Logon to SQL*Plus command-line interface as SYSDBA, but first set the correct Oracle SID.
	
	This will connect you to the Standby Database.
	
	Double-check that you are on the right instance, noting in particular the values for *instance_name*, *host_name* and *status*.
		`$> . oraenv`
		
		[now type in SBRDB1]
		
		`$> sqlplus '/as sysdba'`
		
		`SQL> SELECT * FROM v$instance;`

2.	*[Who: DBA]* Initiate the failover by issuing the following. *Note: Include the **FORCE** keyword to ensure that the RFS processes on the standby database will fail over without waiting for the network connections to time out through normal TCP timeout processing before shutting down.*	`SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH FORCE;`
3.	*[Who: DBA]* Convert the physical standby database to the production role. *Note: Don't get confused by the word "switchover" as this command is part of a complete manual primary failover and not a role switch as may be interpreted by this word.*	`SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;`

FUJ00234958

Step	Description	Server Execution
4a.	*[Who: DBA]* Open the new production (primary) database by issuing the following statement.	`SQL> ALTER DATABASE OPEN;`

4b.	*[Who: DBA]*
	Only complete the following if the condition below is met! ***Otherwise do not.*** This can be verified by checking for this value. Run the following SQL to do so.
	
	Condition:
	If the physical standby database has been opened in read-only mode since the last time it was started, shut down the standby database (now primary database) and restart it.
		`SQL> SELECT value`
		`  FROM v$dataguard_stats`
		`  WHERE name = 'standby has been open';`
		
		`SQL> SHUTDOWN IMMEDIATE;`
		`SQL> STARTUP;`

5.	*[Who: DBA]* Check that all the indexes – database wide – are available for use.	`See Step [5.] of Section 6.1`
6.	*[Who: DBA]* The database initialisation parameters need to be checked and if not correct, need to be set correctly after the Standby database has been successfully transitioned from Standby to it's new role as Primary.	`See Step [6.] of Section 6.1`
7.	*[Who: DBA]* After failover, the "new" Primary database cluster (lprpbds20201 - 4) and database, SBRDB, must accept connections from all applications without changing any application connection properties. Therefore, in order to accomplish this, [i.] a new database service must be created for BRDB and [ii.] the DNS settings for both servers need to be reconfigured.	`See Steps [7.] and [8.] of Section 6.1`
8	*[Who:DBA]* *Reinstall Oracle Fan Event Handler on all nodes*	`See Steps [10.] of Section 6.1`
9.	To maintain a viable disaster-recovery solution in the event of another disaster you must reinstate the original primary database to act as a standby database in the new configuration.	

FUJ00234958

Step	Description	Server Execution
	This can be accomplished by following the original Standby Database deployment handover notes as one must re-create the primary database from a copy of the new primary database.	
	Manual *Complete* Failover through **SQL*Plus** is complete.	

Table 4: SQL*Plus Failover Procedure.

FUJ00234958

6.4 Standby Database Re-instantiation (BDS-to-BDB)

As explained and demonstrated in the preceding sections of this chapter, the original primary database, namely BRDB, would have now failed over to the standby database, namely SBRDB. Therefore in order to ensure a viable and highly available configuration once again, the old primary must be re-instated as the new standby.

The database is setup correctly. All that is required is getting a duplicate of the new primary database back onto the server in order to start the new standby in managed recovery mode. This is that process.

Step	Description	Server Execution
Assumptions	i. User is logged onto the Standby Database Server as ***oracle***. ii. This procedure is only applicable after having completed a failover of Primary (BRDB) to Standby (SBRDB) as detailed in sections 5.1 and 5.3. iii. Only one node should be used as the new standby database node.	

1.	*[Who: DBA]*
	New Prim Server
	
	Backup the new primary (SBRDB) database using RMAN. Ensure there is sufficient space on the device you specify as <RMAN DIR>.
	
	Logon to RMAN.
	
	Execute the backup commands as they appear, e.g. `run { … }`
	
	Exit RMAN and change directory to the <RMAN DIR> and make sure the backup is as you expect. This can be confirmed by listing the backup in RMAN, e.g. `list backup summary;`
		`$> . oraenv`
		
		[now type in SBRDB1]
		
		`$> $ORACLE_HOME/bin/rman NOCATALOG TARGET /`
		
		`RMAN> run`
		`{`
		` backup`
		`  current controlfile`
		`  for standby`
		`  format '<RMAN DIR>/%d_%U';`
		` backup`
		`  format '<RMAN DIR>/%d_%U'`
		`  database;`
		` backup`
		`  format '<RMAN DIR>/%d_%U'`
		`  archivelog all`
		`  not backed up 1 times;`
		`}`
		
		`$> cd <RMAN DIR>`
		`$> ls -l`

FUJ00234958

Step	Description	Server Execution
2.	*[Who: DBA]* New Prim Server Ensure the entire backup, which will consist of a number of files is copied across from this server to the new standby server. **Note:** The backup must exist in the same location on both servers!	`$> scp <RMAN DIR>/* pbdb201<RMAN DIR>`

3.	*[Who: DBA]*
	Old Prim Server (node 1)
	
	Cleanup the old archive directory as it would be full of files that are no longer needed. Type `YES`, if prompted.
	
	Log as "grid" UNIX user
	
	Note: The "rm -r" is extremely destructive! Make sure you're on the correct server and that BRDB most definitely has been failed over.
		`$> . oraenv`
		
		[now type in +ASM1]
		
		`$> asmcmd -p`
		
		`ASMCMD [+] > cd BRDB_FLASH/arch`
		
		`ASMCMD [+BRDB_FLASH/arch] > rm -r brdb*.arc`

4.	*[Who: DBA]*
	Old Prim Server (node 1)
	
	Remove old DBFS files before reinstanciation.
	
	Note: The "rm -r" is extremely destructive! Make sure you're on the correct server and that BRDB most definitely has been failed over.
		`su - grid`
		`grid:>  . oraenv`
		`ORACLE_SID = [grid] ? +ASM1`
		`grid :> asmcmd -p`
		`ASMCMD [+] > cd DBFS_01/BRDB/DATAFILE`
		`ASMCMD [+DBFS_01/BRDB/DATAFILE] > rm -rf DBFS_CHK*`
		`ASMCMD [+DBFS_01/BRDB/DATAFILE] > rm -rf DBFS_DAT*`
		`ASMCMD [+] > exit`

5.	*[Who: DBA]*
	Old Prim Server (node 1)
	
	Set the environment for the new standby database.
	
	Log onto RMAN and execute the restore of the new primary as the new standby.
		`$> . oraenv`
		
		[now type in BRDB1]
		
		`$> $ORACLE_HOME/bin/rman TARGET=sys/<SYS_PASSWD>@sbrdb AUXILIARY /`
		
		`RMAN> duplicate target database for standby;`

FUJ00234958

Step	Description	Server Execution
	Ensure there are no errors in this restore. Otherwise, fix the errors and run again.	
6.	*[Who: DBA]* Old Prim Server (node 1) The standby database ***should already be mounted***, but if not, mount the new standby database.	`SQL> ALTER DATABASE MOUNT STANDBY DATABASE;`

7.	*[Who: DBA]*
	Old Prim Server (node 1)
	Start the standby database in managed recovery mode.
	This must have completed successfully. To check that it has, query `v$dataguard_status`.
	Also, check that the application of logs is performing well, query `v$dataguard_stats`.
		`SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION PARALLEL 8;`
		`SQL> SELECT * FROM v$dataguard_status ORDER BY message_num DESC;`
		`SQL> SELECT * FROM v$dataguard_stats;`

8.	*[Who: DBA]* New Prim Server Ensure that the tnsnames.ora has an entry for the new primary.	`$> cd /app_sw/sbrdb/standby` `$> SBRDB_edit_tnsnames.sh -v -s lprpbds201`

9.	**Note:** The following files should already be available and configured correctly from the previous installation of the old Primary database. If for whatever reason, they are not, configure accordingly: -
	`$ORACLE_HOME/dbs/orapwBRDB`
	`$ORACLE_HOME/network/admin/tnsnames.ora`
	`$ORACLE_HOME/network/admin/listener.ora`
	`/u02/oradata/BRDB/spfileBRDB.ora`
	`/u02/oradata/BRDB/dr1BRDB.dat`
	`/u02/oradata/BRDB/dr2BRDB.dat`	

.		
	Manual re-instantiation of Standby Database Complete.	

Table 5: Primary Re-instatiation Procedure.

6.4.1 Tripwire Configuration

The Tripwire targeting on the EMS platform needs to be edited to:

- Comment out the BDB platforms

FUJ00234958

- Uncomment the BDS platforms

FUJ00234958

6.5 Opening Standby Database "READ ONLY"

This section is not applicable. The standby database is currently configured as an Active Dataguard Physical Standby which implies Database is currently in 'READ ONLY' mode.

Table 6: Opening Standby Database Read-only

6.6 Standby Cluster – Software Installation

The Standby Database BladeFrame has been configured (for the first release) to make use of a single active pServer and 3 inactive (i.e. 1 pBlade plugged in and active with the remaining pBlades utilized elsewhere). This setup effectively makes the cluster run as single-node RAC cluster, but at the point where a failover is required, the remaining pBlades are activated allowing the cluster the full compliment of nodes.

Having this configuration is sufficient for running in an environment where there is no need for software updates. However, software installations, UNIX patches, database software upgrades, database patches, etc. is an ongoing required activity.

Therefore the following describes a means of accomplishing a software update across all standby nodes in order to keep them functionally in sync with ***lprpbds201*** (node 1).

There are two possibilities, both of which will require a period of downtime, so ideally this would be after working hours each day or on the weekend. The first, "Alternative A", will allow the software update to be accomplished fairly quickly but renders the primary cluster without throughput, which may be considered a problem if batch schedules run at the same time. The second possibility will be accomplished a lot slower, but leaves the primary cluster with the ability to carry most of the operational workload.

Both possibilities are in essence the same set of steps, just executed in differing combinations of pBlades.

Alternative	Implementation Description
NOTE These alternatives are presented at a high level and the level of detail required, is beyond the scope of this document. The steps mentioned below will need to be coordinated by more than one team; At first glance, those teams would likely be UNIX Support, DBA Support and cooperation from Tivoli/Schedule Support (SMC).	

FUJ00234958

Alternative	Implementation Description
Alternative A.	Primary Cluster i. SMC: Give the go-ahead that all schedules are held for the affected node(s) . Standby Cluster iv. UNIX: Check the pServers which correspond to nodes 2, 3 and 4, e.g. ***lprpbds202 – 004*** v. DBA/UNIX/3rd Party: Perform the required change, installation, patch, etc. Once the required changes are complete, reverse the process of implementation and restore the pBlades to their original BladeFrame, thereby returning the Primary Cluster to it's former, fully operational state, including all Grid Control blackouts and notification to SMC. There must be no unresolved Grid Control alerts or exceptions in BRDB_OPERATIONAL_EXCEPTIONS at the end of this process. The BAL OSR's need to also be recycled at this point. Finally, if the "End of Day" process is not, for whatever reason, going to be run by the time all nodes will be required, then one needs to use the process defined in Section 4.3.3 to logically bring the nodes/instances back into operation. NOTE When restoring (whether in a failover scenario or general build maintenance) the Standby nodes ***lprpbds202 – 004*** Oracle CRS will not know that the instances SBRDB2 - 4 are standby instances and will therefore behave as configured and attempt to start them. This behaviour is correct and must not be changed. Because of the way this is configured, this should always be a manual task, i.e. make sure that the *apply instance* (SBRDB1) has been started and mounted "as standby"; this will "kick off" the recovery process. Even though the other instances are not up, they will be in "nomount" mode, so bring and keep them down.
Alternative B	No viable alternatives have currently been agreed upon.

Table 7: Alternatives for Managing Software Installations on BDS nodes.

FUJ00234958

6.7 Standby Database – Manual Re-instantiation Procedure

Table 8 below, details the *manual* re-instantiation procedure for the Standby Database as it was originally configured in the Branch Database build. The procedure lists the steps required in stopping the Data Guard configuration, removing the BDS database, backuping up the BDB database and recreating the entire BDS Standby Database Data Guard configuration.

Once again, the original configuration which is BDB-to-BDS is the focus of this section.

6.7.1 AUDIT Files Prior to Failback

Step	Description	Server Execution
	Prior to failing back, certain manual steps need to be carried out in order to transfer TWS logs from SBRDB to BRDB	
1.	*UNIX Admin on Standby* Login into the BDS server Cancel managed recovery.	`Stop TWS jobs being scheduled on all 4 BDS servers`
2.	*On BDS*	`Rename all arc files in /opt/tws/TWS/MAEARC with V002 extensions instead of V001 on all 4 BDS servers`
3.	*On BDS*	`Copy arc files to a safe area (a suitable NAS share available to BDB & BDS`
4.	*On BDS*	`Two days after the failback, run job /opt/tws/TWS/sh/audit_stdlist.sh to tar up the stdlists for the current day on all 4 BDS servers.`

6.7.2 Database

FUJ00234958

Step	Description	Server Execution
	Before beginning, open up a session, logging in as oracle, on both the primary and the standby servers. At the time fof writing, this procedure is recommended only for running in LST or LIVE.	

1.	*DBA on Standby*
	
	Login into the database (as oracle)
	
	Cancel managed recovery.
		`. oraenv`
		`[now type in SBRDB1]`
		
		`sqlplus '/as sysdba'`
		
		`SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;`

2.	*DBA on Standby*
	
	Login to the Dataguard broker (still as oracle)
	
	Confirm the configuration.
	
	Stop and remove the configuration.
		`dgmgrl`
		`connect sys/<password>@brdb`
		
		`DGMGRL> show configuration`
		
		`DGMGRL> disable configuration`
		`DGMGRL> remove configuration`

3.	*DBA on Primary*
	
	Login into the database (as oracle)
	
	Stop the broker.
	
	Exit SQL*Plus
	
	Create a backup of the SPFILE.
		`. oraenv`
		`[now type in BRDB1]`
		
		`sqlplus '/as sysdba'`
		
		`SQL> ALTER SYSTEM SET dg_broker_start=FALSE SCOPE=both SID='*';`
		
		`cd /u02/oradata/BRDB/`
		`cp spfileBRDB.ora spfileBRDB.ora.bck`

4.	*DBA on Standby* Stop the SBRDB database.	`srvctl stop database -d SBRDB`

FUJ00234958

5.	*DBA on Primary*
	
	RESET some of the Dataguard-related parameters (you should already be in the database).
		`ALTER SYSTEM RESET log_archive_config SCOPE=spfile SID='*';`
		
		`ALTER SYSTEM RESET log_archive_dest_2 SCOPE=spfile SID='*';`
		
		`ALTER SYSTEM RESET log_archive_dest_state_2 SCOPE=spfile SID='*';`
		
		`ALTER SYSTEM RESET fal_server SCOPE=spfile SID='*';`
		
		`ALTER SYSTEM RESET fal_client SCOPE=spfile SID='*';`
		
		`ALTER SYSTEM RESET archive_lag_target SCOPE=spfile SID='*';`

6.	*DBA on Primary* Remove the Dataguard configuration files.	`rm /u02/oradata/BRDB/dr*BRDB.dat`

7.	*DBA on Primary*
	
	Login to RMAN.
	
	Execute the following RMAN configuration changes before executing the Primary instantiation scripts.
		`$ORACLE_HOME/bin/rman nocatalog target /`
		
		`CONFIGURE CONTROLFILE AUTOBACKUP OFF;`
		`CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1;`
		`CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1;`

8.	*DBA on Primary* Clear out old lock files **`/app_sw/brdb/standby/tmp`**	`cd /app_sw/brdb/standby/tmp` `rm *`
9.	*DBA on Standby* At this point, some cleanup is required. If you want to be sure the SBRDB database is completely cleared out, then do so by running [9a - c.] If you'd prefer to just run the re-instantiation procedure as fast as possible, allowing RMAN to *overwrite* the database files that exist, then *skip* [9a.] and [9b.]	

FUJ00234958

9a.	*DBA on Standby*
	Complete this step, should you wish to, clear out the Standby database by removing the database files and/or archivelogs from ASM.
	Make sure you're happy with the diskgroup names, by listing and checking them.
	
	Now remove the files.
	Now remove the old archivelogs.
		`. oraenv`
		`[now type in +ASM1]`
		
		`asmcmd -p`
		`ASMCMD [+] > lsdg`
		
		`ASMCMD [+] > rm -f SBRDB*/*brdb*`
		`ASMCMD [+] > rm -f SBRDB_FLASH/arch/*.arc`

9b.	*DBA on Standby* Should you wish to, remove the standby database from the cluster configuration.	`srvctl remove database -d SBRDB -f`

9c.	*DBA on Standby*
	
	Clear out old lock files from **`/app_sw/sbrdb/standby/tmp`**
	
	Clear out the old backup files previously copied from primary during first installation, if the still exist.
		`cd /app_sw/sbrdb/standby/tmp`
		`rm *`
		
		`cd /app_sw/rman_backup`
		`rm dbf* arc*`

10.	*DBA on Primary* Stop and restart the BRDB database.	`srvctl stop database -d BRDB` `srvctl start database -d BRDB`
	Cleanup all done. *Re-instantiation follows …*	
	Note: Running the Standby Instantiation Scripts on primary will shutdown all four instances for the backup and restart only instance 1. Make sure you restart the rest before carrying on with the standby scripts. Ideally, the Standby Database Instantiation baselines could be executed, however as this is a *manual* support procedure, the scripts can *manually be executed* as follows from step 11 onwards. At the time fof writing, this procedure is recommended only for running in LST or LIVE. Script Hierarchy and Dependency BDB 0. BRDBConfig.sh — Config script, not to be executed. BDB 1. BRDBInitialisePrimary.sh — Needs [0] BDS 2. SBRDBInitialiseStandby.sh — Needs [0]; requires [1] to have run. BDS 3. SBRDBAddStandbyLogs.sh — Needs [0]; requires [1,2] BDB 4. BRDBCementPrimary.sh — Needs [0]; requires [1,2,3]	
11a.	*DBA on Primary* Execute the BRDB Database Standby Instantiation preparation script.	`BRDBInitialisePrimary.sh -v -s <standby_node>`

FUJ00234958

11b.	*DBA on Primary*
	
	Copy the following files to the <standby_node>
	
	Note the from and to directories; these must be as they are in this example.
		`scp /app_sw/brdb/standby/tmp/initSBRDB.ora l<env>bds201:/app_sw/sbrdb/standby`
		`scp /app_sw/rman_backup/stby_ctl_* l<env>bds201:/app_sw/rman_backup`
		`scp /app_sw/rman_backup/dbf_* l<env>bds201:/app_sw/rman_backup`
		`scp /app_sw/rman_backup/arc_* l<env>bds201:/app_sw/rman_backup`

12.	*DBA on Standby* Execute the SBRDB Database Standby Instantiation preparation script after copying the files identified in [11b.] Note: This will take a while as RMAN unacks and creates/overwrites each file of the SBRDB database.	`SBRDBInitialiseStandby.sh -v -s <primary_node>`
13.	*DBA on Standby* Execute the SBRDB Database Standby Redolog Creation Script.	`SBRDBAddStandbyLogs.sh -v -s <primary_node>`

14.	*DBA on Primary and Standby*
	
	Ensure there aren't any untoward errors and that the alert logs show archivelogs and standby redologs "ticking over" regularly without warnings or errors.
		`less /u01/admin/BRDB/bdump/alert_BRDB1.log`
		
		`less /u01/admin/SBRDB/bdump/alert_SBRDB1.log`

	Re-instantiation done.	

Table 8: BDB-to-BDS Manual Re-instantiation Procedure

6.7.3 AUDIT Files After Failback

Step	Description	Server Execution
	After failing back, manual steps need to be carried out in order to transfer TWS logs from SBRDB to BRDB	
1.	*On BDS*	**Move V002 files into the relevant BDB directory /opt/tws/MAEARC once the failback is complete (see 6.6.1)**

6.7.4 Reinstall Oracle FAN_EVENT ON BDB

ON BDB NODE1-4

```
$> mv -f /app_sw/brdb/sh/fan_event_handler.ksh /u01/app/11.2.0/grid/racg/usrco
$> chown grid:oinstall /u01/app/11.2.0/grid/racg/usrco/fan_event_handler.ksh
$> chmod 550 /u01/app/11.2.0/grid/racg/usrco/fan_event_handler.ksh
```

FUJ00234958

Remove Oracle `FAN Event handler for all` the BDS Cluster (***on each node, e.g. 1 – 4***),

As the ***`root`*** user …

```
$> mv -f /u01/app/11.2.0/grid/racg/usrco/fan_event_handler.ksh /app_sw/brdb/sh/
```

6.7.5 Stopping Goldengate on BDS prior to failback

Step	Description	Server Execution
1	Login to BDS as 'oggadmin' user	
2	Invoke ggsci	`$OGG_HOME/ggsci`

3	Stop extract	`GGSCI> stop E11BDB`
		
		`Sending STOP request to EXTRACT E11BDB ...`
		`Request processed.`

4	Issue info all repeatedly until E11BDB is shown as STOPPED	`GGSCI> info all`
		
		`Program Status Group Lag at Chkpt Time Since Chkpt`
		
		`MANAGER RUNNING`
		`EXTRACT STOPPED E11BDB 00:00:05 00:01:58`
		`EXTRACT RUNNING P11BDB 00:00:00 00:00:06`

5	Wait until all lag disappears against data pump P11BDB by issuing:	`GGSCI> info all`
		
		`Program Status Group Lag at Chkpt Time Since Chkpt`
		
		`MANAGER RUNNING`
		`EXTRACT STOPPED E11BDB 00:00:05 00:01:58`

FUJ00234958

		EXTRACT RUNNING P11BDB 00:00:00 00:00:06
6	Stop datapump once lag is gone	GGSCI> stop P11BDB Sending STOP request to EXTRACT P11BDB ... Request processed.
7	Ensure data pump P11BDB is stopped by issuing:	GGSCI> info all Program Status Group Lag at Chkpt Time Since Chkpt MANAGER RUNNING EXTRACT STOPPED E11BDB 00:00:05 00:05:32 EXTRACT STOPPED P11BDB 00:00:00 00:02:25
8	Stop all OGG processes via GGSCI (logon as 'oggadmin' user)	GGSCI> **stop *** GGSCI> stop mgr ! GGSCI> info all
9	Stop OGG resource via CRS (logon as 'oracle' user)	oracle> **crsctl stop resource dbfs_mount**
10	Login to BRS as 'oggadmin' user	
11	Invoke ggsci	$OGG_HOME/ggsci
12	Ensure there is no lag associated with replicat R11BRS by issuing	GGSCI> info all Program Status Group Lag at Chkpt Time Since Chkpt MANAGER RUNNING REPLICAT RUNNING R11BRS 00:00:00 00:00:03

FUJ00234958

12	Wait until lag is zero before continuing with the rest of the failback procedure	

6.7.6 Starting Goldengate on BDB after failback

1	• Start the dbfs resource on BDB node 1 Login as oracle user to BDB node 1 oracle:> crsctl stop resource dbfs_mount oracle:> crsctl start resource dbfs_mount
2	• Login as Unix user oggadmin on BDB node 1 • $oggadmin> $OGG_HOME/ggsci • GGSCI> DELETE CREDENTIALSTORE • GGSCI> ADD CREDENTIALSTORE • GGSCI> INFO CREDENTIALSTORE • GGSCI> ALTER CREDENTIALSTORE ADD USER ops$oggadmin , PASSWORD some-secure-password, ALIAS LOGINALIAS • GGSCI> INFO CREDENTIALSTORE • GGSCI> exit • $oggadmin:> sqlplus / as sysdba • SQL> ALTER USER ops$oggadmin identified by some-secure-password;
3	• Login to BDB as 'oracle' user, start OGG processes (where lxxpbdb201 is dependent on the environment – LST or LIVE) oracle> crsctl start res brdb.oggadmin.oggapp -n lxxpbdb201 -f

6.7.7 RMAN CATALOG RESYNC

Resync RMAN catalog once the BDS servers are back into operation as "Physical Standby" and "Active Data Guard" is running using the following commands. On the Standby Database.

Login as "oracle" UNIX user

```
oracle:> .oraenv
ORACLE_SID = [] ? SBRDB1
oracle:> export ORADATA_DIR="/u02/oradata"
oracle:> export ORA_HOME=/home/oracle
oracle:> export WALLET_HOME=${ORADATA_DIR}/rman/wallet
oracle:> export TNS_ADMIN=${WALLET_HOME}/tnsadmin
```

FUJ00234958


```
oracle:> export WALLET_MARKER_FILE=${WALLET_HOME}/wallet_marker.dat
oracle:> $ORACLE_HOME/bin/rman  target /@SBRDB1 catalog /@RMANCAT
RMAN:> RESYNC CATALOG;
RMAN:> exit
oracle:> $ORACLE_HOME/bin/rman  target /@BRDB1 catalog /@RMANCAT
RMAN:> CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
RMAN:> exit
```

6.7.8 CONTROLFILE SNAPSHOT RESYNC:

After re-instantication of BDS, do the following on BDB to re-create SNAPSHOT CONTROLFILE location.

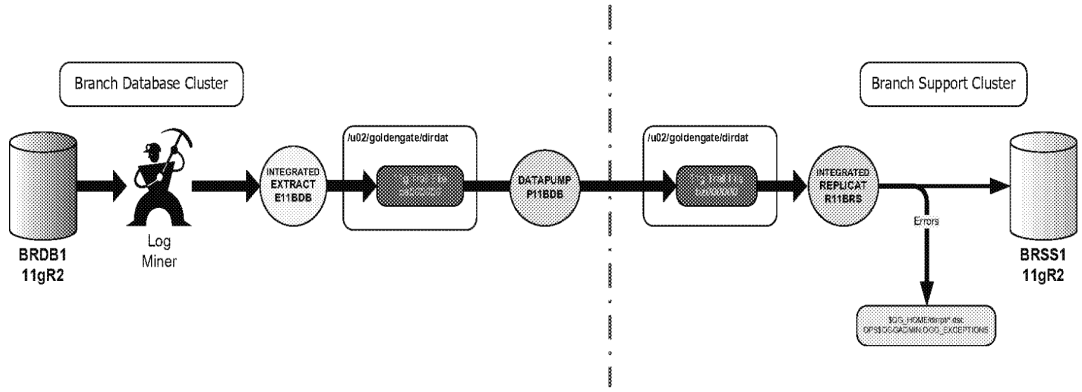
Login as “oracle” UNIX user

```
oracle:> .oraenv
ORACLE_SID = [] ? BRDB1
oracle:> rman target /
RMAN:> CONFIGURE SNAPSHOT CONTROLFILE NAME TO 'DEFAULT';
RMAN:> CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u02/oradata/BRDB/snapcf_BRDB.f';
RMAN:> exit
```

7 Appendix B – Branch Support

The Branch Support Database is a database used in supporting the main BRDB application by providing access to all data found in the main database but without having access to it. The means by which the data is replicated from BRDB to BRSS is via Oracle Goldengate. OGG is inherently complex and therefore has multiple facets to consider when supporting it day-to-day and troubleshooting any problems that arise.

The following procedures detail the rather destructive process of cleaning out all the Streams queue tables, queues, rules and configuration and then re-creating it. This is ***extremely destructive and cannot be recovered*** without restoring both the Branch Database and the Support Database, unless this is an intended action (see Section 7.1.2).



FUJ00234958

7.1 Managing Goldengate Lag

7.1.1 Context and Assumptions

Oracle Goldengate is in essence, a set of components which capture changes at a source database, propagate those changes and then apply them on a target database.

7.1.2 Lag Evaluation and Escalation

Our recommendation is that at the following periods the appropriate action is performed, bearing in mind that as the solution matures, the responses might change or even the periods at which escalation/investigation begins, might change: -

Lag Period	Action
4 hrs.	DBA Support notified in order to understand the transactions responsible and continue to monitor apply progress. SSC made aware if lag is occurring during or just before core hours.
8 hrs.	DBA Support notified. Are the original problems reoccurring? Is it the same or a similar transaction?
12 hrs	DBA Support notified. Are the original problems reoccurring? 4th-Line Support notified of the cause and progress.
16 – 20 hrs.	DBA Support notified. Are the original problems reoccurring?

24 hrs.	DBA Support notified.
	4th-Line Support notified.
	Appropriate business owner notified.
	At this point, there are ***x*** number of days (currently 4) which remain in which to continue the investigation or to put in place a fix and prepare for OGG re-instantiation, should the decision be made to do so.
	Note that ***x is defined as the lowest number of days for data retention of any table on BRDB.*** The following query shows the result:
	`SELECT MIN(retention_period)`
	`  FROM brdb_archived_tables`
	`  WHERE retention_period <> 0`
	`  AND additional_criteria IS NULL;`

48 hrs.	Re-evaluate the situation and prepare for re-instantiation providing all the approvals have been received.

Table 10: Lag Evaluation Actions

7.2 Goldengate DML Behaviour on OPS$BRDB Tables

FUJ00234958

The diagram in Section 7 shows an overview of the Oracle Goldengate technology by which the data is replicated from BRDB to BRSS, the processes involved and the action performed by each.

The table below is for informational purposes and has been included to aid in determining why certain problems with data in BRSS *might occur*, e.g. data seems to have "disappeared" or is the cause of Streams errors theoretically based on an assumption by a user that the deletion of data in BRDB succeeded therefore it must also have succeeded in BRSS. This will not be the case for tables which Streams is configured to discard deletes for (see below).

The majority of DELETE operations carried out during the BRDB purging process (BRDBC004) are not replicated to BRSS. The purpose of this is to allow differential retention periods between BRDB and BRSS. The list of tables which have their DELETEs discarded can be identified by querying BRDB_ARCHIVED_TABLES where column ALLOW_REPLICATION = 'N'. The list of tables is attached below:

discarded.deletes.xlsx

7.3 Data Aggregations

Host Data Aggregation modules in the Branch Database have been cloned and implemented in BRSS as part of HNG-X Release 5 CP0639 – Capacity Management Reporting.

Except for minor customisations done to localise the modules in BRSS Database, the data aggregation related database objects, LINUX shell scripts and TWS schedule job definitions will almost entirely resemble their counterparts in BRDB. It has to be noted that the Data Aggregation processes in BRSS will not perform Instance ID/Fad Hash based processing as it is not applicable to BRSS.

The following tables have been created in BRSS Database to contain aggregation metadata and report statistics for Capacity Management Reporting:

- BRSS_HOST_AGGREGATIONS
- BRSS_HOST_AGGREGATION_CTL
- BRSS_CAPMGMT_5MIN_STATS
- BRSS_CAPMGMT_HOURLY_STATS
- BRSS_CAPMGMT_DAILY_STATS

7.4 Table of BRSS Host Processes

The following table lists the current BRSS Host processes, a brief description of each and the names of the executables used to run them. The process name corresponds to the name that is registered in table BRSS_PROCESSES and, where applicable, the name that is used to control processing via table BRSS_PROCESS_CONTROL.

No.	Executable	BRSS Process Name	Description
1	BRSSC001	BRSSC001	Start of Day
2	BRSSC004	BRSSC004	Audit, Archive, Purge
3	BRSSX002.sh	BRSSX002	BRSS Message Journal Auditing
4	BRSSX005.sh	BRSSX005.sh	Gather Optimiser Statistics
5	BRSSX006.sh	BRSSX006	File Housekeeping

FUJ00234958

No.	Executable	BRSS Process Name	Description
6	BRSSX007.sh	SLT_TO_5MIN_STATS	Data aggregation for Cap Mgmt Reporting - Peak 5-Minute Stats for HNG-X RAW SLT STATS
7	BRSSX007.sh	SETTLEMENT_TO_5MIN_STATS	Data aggregation for Cap Mgmt Reporting - Peak 5-Minute Stats for Settlement transactions
8	BRSSX007.sh	NRT_TO_5MIN_STATS	Data aggregation for Cap Mgmt Reporting - Peak 5-Minute Stats for NRT transactions
9	BRSSX007.sh	5MIN_TO_HOURLY_STATS	Data aggregation for Cap Mgmt Reporting - Peak Hourly Stats
10	BRSSX007.sh	HOURLY_TO_DAILY_STATS	Data aggregation for Cap Mgmt Reporting - Peak Daily Stats
11	BRSSX021.sh	BRSSX021	Streams Pause, Start
12	BRSSX022.sh	BRSSX022	Daily copy of DBA_HIST tables from BRDB into BRSS
13	BRSSX023.sh	BRSSX023	Pre-processor job for GREPX001
14	GREPX001.sh	GREPX001	Generic Reporting Mechanism - view creation
15	GREPX002.sh	GREPX002	Generic Reporting Mechanism - report extraction
16	BRSSX037.sh	BRSS_CLR_BRANCH_DATA	BRSS Branch closure clear down
17	BRSSX011.sh	BRSSX011_BRSS_SANITISE_START	BRSS Sanitisation START & STOP
18	BRSSC093	BRSSC093_APSCR BRSSC093_APSADC BRSSC093_NWB BRSSC093_DCS BRSSC093_GREV BRSSC093_JRNL	BRSS Sanitisation

Table 13: BRSS Host Processes

7.5 BRSS Scheduling

7.5.1 Schedule BRSS_TRACE_STOP1

This schedule is run daily (07:30 a.m.).

7.5.1.1 Dependencies

None.

7.5.1.2 Job BRSSX011_TRACE_PAUSE_1

Updates the BRSS_SYSTEM_PARAMETERS table, sets parameter BRSS_C002_STOP_YN flag to 'Y'.

7.5.1.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameter.

7.5.1.2.2 Rerun Action

*** Prompts for rerun – action? **

FUJ00234958

7.5.2 Schedule BRSS_SOD

This schedule is run daily (08:00 a.m.).

7.5.2.1 Dependencies

Flag in " /opt/tws/FLAGS/BRSS_COMPLETE.flag" present.

7.5.2.2 Job BRSS_RM_COMPLETE_FLAG

Removes BRSS_COMPLETE.flag.

7.5.2.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameter.

7.5.2.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.3 Schedule BRSS_CLR_BRANCH

This schedule runs from 9pm but only after BRSS_SOD and BRDB_FROM_EMDB2 complete but will not start if the time is past 01:05. The called job archives and then deletes transactions for all closed branches. This schedule is run on 1 instance at any one time.

7.5.3.1 Dependencies

Schedule BRSS_CLR_BRANCH depends on the completion of schedules BRSS_SOD and BRDB_FROM_EMDB2. This job will not start after 01:05.

7.5.3.2 Job BRSSX037_CLEAR_BRDATA

This job runs the BRSS automated closure process (BRSSX037.sh).

7.5.3.2.1 Implementation

This job is implemented by a call to the shell script BRSSX037.sh, along with the TWS business date and instance number.

The process identifies all branches to be cleared by the following query

```
SELECT branch_accounting_code
FROM   OPS$BRDB.brdb_cleared_closure_data
WHERE  brss_cleared_date IS NULL
```

All transactions for those closed branches in a number of tables (identified in column BRDB_CLEARED_CONTROL_DATA.source_table) are loaded into archive tables (identified in column BRDB_CLEARED_CONTROL_DATA.target_table) and then deleted from the original tables.

Closed, cleared and archived branches are recorded in table BRDB_CLEARED_CLOSURE_DATA, with column brss_cleared_date identifying when the branch was cleared on BRSS.

7.5.3.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.4 Schedule BRSS_TRACE_STRT1

This schedule is run daily (at 8:10). Allows BRSSC002 to restart by resetting the start/stop flag.

7.5.4.1 Dependencies

FUJ00234958

Schedule BRSS_TRACE_STRT1 depends on the completion of schedule BRSS_SOD.

7.5.4.2 Job BRSSX011_TRACE_RESUME

Updates the BRSS_SYSTEM_PARAMETERS table, sets parameter BRSS_C002_STOP_YN flag to 'N'.

7.5.4.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameter.

7.5.4.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.5 Schedule BRSS_JRNL_TRACE1

This schedule is run daily.

7.5.5.1 Dependencies

Schedule BRSS_JRNL_TRACE1 depends on the completion of schedule BRSS_TRACE_STRT1.

7.5.5.2 Job BRSSC002_JRNL_TRACE1

The message journal tracing process (BRSSC002) will generate text files for a given day's journalised messages by reading records from the message journal table (BRDB_RX_MESSAGE_JOURNAL). The process will run throughout the day as a Unix daemon. This process is essentially a clone of BRDBC002 without the check that sequence numbers are a dense set.

7.5.5.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date.

Outputs files to the following directory below.

Usage	Environment Variable
BRSS output directory	BRSS_COUNTER_AUDIT_OUTPUT

7.5.5.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.6 ~~Schedule BRSS_DXC~~

~~This schedule is run daily (??:??).~~

7.5.6.1 ~~Dependencies~~

~~Schedule BRSS_DXC depends on the completion of schedule DW_EOD.~~

7.5.6.2 ~~Job BRSS_DXC_RUN~~

~~This job is used to transfer Reporting information from the BRSS environment (specifically a NAS share named, `/app/brss/trans/support/sltreports`) to the "Corporate" environment. This is accomplished by executing a DXC java client which invokes a "transfer plan", allowing the contents of the above directory to be copied to "Corporate" via the DXC.~~

~~Please note that this job is not owned by Host development.~~

7.5.6.2.1 ~~Implementation~~

~~This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and input parameters as shown: `/app_sw/dxc/executedxc.sh upload BRSSMSUOUTPUT`~~

FUJ00234958

7.5.6.2.2 ~~Rerun Action~~

~~*** Prompts for rerun – action? **~~

7.5.7 Schedule BRSS_GEN_REP

This schedule is run daily. Every 5 hours until 0700 hrs.

IN THE EVENT OF FAILURE: See Section 5.10 for recovery tasks.

7.5.7.1 Dependencies

Schedule BRSS_GEN_REP depends on the completion of schedule BRSS_SOD.

7.5.7.2 Job GENERIC_CREATE_REPORT_VIEWS

Calls shell script BRSSX023.sh with the TWS business date.

7.5.7.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date. The shell script BRSSX023.sh will in turn, call shell script GREPX001.sh (and subsequent aggregation jobs) depending upon the outcome of the validation performed between BRSS_C002_JOURNAL_DATE and REP_EFFECTIVE_DATE. This validation ensures that if the date values of these two parameters are equal, that the chain iof dependent jobs is executed, otherwise BRSSX023.sh does not run..

7.5.7.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.7.3 Job BRSSX007_SLT_TO_5MIN_STATS

Calls shell script BRSSX007.sh with aggregation name 'SLT_TO_5MIN_STATS' and the TWS business date. Data aggregation performed by this job will be used for Capacity Management Reporting requirements of Customer Services.

7.5.7.3.1 Dependencies

Job BRSSX007_SLT_TO_5MIN_STATS depends on the completion of job GENERIC_CREATE_REPORT_VIEWS.

7.5.7.3.2 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name, aggregation name and date.

7.5.7.3.3 Rerun Action

*** Prompts for rerun – action? **

7.5.7.4 Job BRSSX007_SETTLEMENT_TO_5MIN_STATS

Calls shell script BRSSX007.sh with aggregation name 'SETTLEMENT_TO_5MIN_STATS' and the TWS business date. Data aggregation performed by this job will be used for Capacity Management Reporting requirements of Customer Services.

7.5.7.4.1 Dependencies

Job BRSSX007_SETTLEMENT_TO_5MIN_STATS depends on the completion of job BRSSX007_SLT_TO_5MIN_STATS.

7.5.7.4.2 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name, aggregation name and date.

7.5.7.4.3 Rerun Action

FUJ00234958

*** Prompts for rerun – action? **

7.5.7.5 Job BRSSX007_NRT_TO_5MIN_STATS

Calls shell script BRSSX007.sh with aggregation name 'NRT_TO_5MIN_STATS' and the TWS business date. Data aggregation performed by this job will be used for Capacity Management Reporting requirements of Customer Services.

7.5.7.5.1 Dependencies

Job BRSSX007_NRT_TO_5MIN_STATS depends on the completion of job BRSSX007_SETTLEMENT_TO_5MIN_STATS.

7.5.7.5.2 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name, aggregation name and date.

7.5.7.5.3 Rerun Action

*** Prompts for rerun – action? **

7.5.7.6 Job BRSSX007_5MIN_TO_HOURLY_STATS

Calls shell script BRSSX007.sh with aggregation name '5MIN_TO_HOURLY_STATS' and the TWS business date. Data aggregation performed by this job will be used for Capacity Management Reporting requirements of Customer Services.

7.5.7.6.1 Dependencies

Job BRSSX007_5MIN_TO_HOURLY_STATS depends on the completion of job BRSSX007_NRT_TO_5MIN_STATS.

7.5.7.6.2 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name, aggregation name and date.

7.5.7.6.3 Rerun Action

*** Prompts for rerun – action? **

7.5.7.7 Job BRSSX007_HOURLY_TO_DAILY_STATS

Calls shell script BRSSX007.sh with aggregation name 'HOURLY_TO_DAILY_STATS' and the TWS business date. Data aggregation performed by this job will be used for Capacity Management Reporting requirements of Customer Services.

7.5.7.7.1 Dependencies

Job BRSSX007_HOURLY_TO_DAILY_STATS depends on the completion of job BRSSX007_5MIN_TO_HOURLY_STATS.

7.5.7.7.2 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name, aggregation name and date.

7.5.7.7.3 Rerun Action

*** Prompts for rerun – action? **

7.5.7.8 Job GENERIC_CREATE_REPORTS

Calls shell script grepx002.sh with the system name (BRSS), outputs text based report files.

Outputs files to the following directories below.

Usage	BRDBBLV1 Environment Variable
Working directory	BRSS_MSU_WORKING

FUJ00234958

BRSS reports directory	BRSS_MSU_OUTPUT

7.5.7.8.1 Dependencies

Job GENERIC_CREATE_REPORTS depends on the completion of job BRSSX007_HOURLY_TO_DAILY_STATS.

7.5.7.8.2 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameter.

7.5.7.8.3 Rerun Action

*** Prompts for rerun – action? **

7.5.8 Schedule BRSS_ORA_STATS

This schedule is run daily (01:05).

7.5.8.1 Dependencies

Schedule BRSS_ORA_STATS depends on the completion of schedule BRSS_SOD.

7.5.8.2 Job BRSSX005_SCHEMA

Gathers statistics on all objects within the OPS$BRSS and OPS$BRDB schemas.

7.5.8.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date.

7.5.8.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.9 Schedule BRSS_ADMIN

This schedule is run daily (01:15).

7.5.9.1 Dependencies

Schedule BRSS_ADMIN depends on the completion of schedule BRSS_SOD.

7.5.9.2 Job BRSSC004

Calls binary BRSSC004 to housekeep BRSS.

7.5.9.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date.

7.5.9.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.9.3 Job BRSSX022

Calls shell script BRSSX022.sh to copy AWR statistics from BRDB into BRSS tables with names starting "HIST_BRDB_". BRDBX022.sh then calls the procedure **`ops$brss.hist_brdb_refresh`**, which copies the tables in the order specified below: -

FUJ00234958


```
[ 1] HIST_BRDB_SYS_TIME_MODEL
[ 2] HIST_BRDB_SYSSTAT
[ 3] HIST_BRDB_SYSTEM_EVENT
[ 4] HIST_BRDB_SQLSTAT
[ 6] HIST_BRDB_SNAPSHOT
[ 5] HIST_BRDB_SQLTEXT
[ 7] HIST_BRDB_ACTIVE_SESS_HISTORY
[ 8] HIST_BRDB_SGASTAT
[ 9] HIST_BRDB_SQL_PLAN
[10] HIST_BRDB_OPTSTAT_HSTHEAD_HST
[11] HIST_BRDB_OPTSTAT_TAB_HISTORY
[12] HIST_BRDB_OPTSTAT_IND_HISTORY
[13] HIST_BRDB_OPTSTAT_HISTGRM_HST
```

The stats have the potential to be copies the tables in the order specified below: -

7.5.9.3.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date.

7.5.9.3.2 Rerun Action

A re-run is not required, nor recommended. Mark job complete. Work will complete next time job is run.

7.5.9.4 Job BRSSX006

Calls binary BRSSX006 to housekeep BRSS directories.

7.5.9.4.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date.

7.5.9.4.2 Rerun Action

*** Prompts for rerun – action? **

7.5.9.5 Job BRSS_HkP_Orafiles1

Calls script HousekeepOrafiles.sh to housekeep Oracle files.

7.5.9.5.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameter.

7.5.9.5.2 Rerun Action

*** Prompts for rerun – action? **

7.5.9.6 Job BRSS_HkP_Orafiles2

Calls script HousekeepOrafiles.sh to housekeep Oracle ASM files.

7.5.9.6.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameter.

7.5.9.6.2 Rerun Action

*** Prompts for rerun – action? **

7.5.10 Schedule BRSS_START_BKP

FUJ00234958

This schedule is run daily (with an alert if not started by 04:00).

7.5.10.1 Dependencies

Schedule BRSS_START_BKP depends on the completion of schedule BRSS_ADMIN.

7.5.10.2 Job MARKER

Writes marker.

7.5.10.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and date.

7.5.10.2.2 Rerun Action

CONTINUE

7.5.11 Schedule BRSS_BACKUP_0

This schedule is run every 4th Sunday.

7.5.11.1 Dependencies

Schedule BRSS_BACKUP_0 depends on the completion of schedule BRSS_START_BKP.

7.5.11.2 Job BRSS_LVL0_BACKUP

Carries out level 0 RMAN backup.

7.5.11.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameters.

7.5.11.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.12 Schedule BRSS_BACKUP_1

This schedule is run daily except 4th Sunday.

7.5.12.1 Dependencies

Schedule BRSS_BACKUP_1 depends on the completion of schedule BRSS_START_BKP.

7.5.12.2 Job BRSS_LVL1_BACKUP

Carries out level 1 RMAN backup.

7.5.12.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameters.

7.5.12.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.13 Schedule BRSS_STARTUP

This schedule is run daily (raises alert if not started by 06:00).

7.5.13.1 Dependencies

FUJ00234958

Schedule BRSS_STARTUP depends on the completion of schedule BRSS_BACKUP_0 or BRSS_BACKUP_1.

7.5.13.2 Job BRSSC001

Calls start of day process BRSSC001 to generate the next day's partitions.

7.5.13.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameters.

7.5.13.2.2 Rerun Action

*** Prompts for rerun – action? **

7.5.14 Schedule BRSS_COMPLETE

This schedule is run daily.

7.5.14.1 Dependencies

Schedule BRSS_COMPLETE depends on the completion of schedules BRSS_STARTUP, BRSS_TRACE_STOP1 and BRSS_GEN_REP.

7.5.14.2 Job BRSS_COMPLETE_FLAG

Creates complete flag.

7.5.14.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameters.

7.5.14.2.2 Job Dependency

This job is dependent on job BRSSC001.

7.5.14.2.3 Rerun Action

*** Prompts for rerun – action? **

7.5.15 Schedule BRSS_MONITOR

This schedule is run daily.

7.5.15.1 Dependencies

None

7.5.15.2 Job BRSS_MON_STARTUP

Calls maestro script monitor_schedule.sh

7.5.15.2.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameters.

7.5.15.2.2 Rerun Action

CONTINUE

7.5.15.3 Job BRSS_MON_BKP

Calls maestro script monitor_schedule.sh

FUJ00234958

7.5.15.3.1 Implementation

This job is implemented by a call to the Maestro monitor schedule command with the relevant job name and parameters.

7.5.15.3.2 Rerun Action

CONTINUE

7.5.16 Schedule BRSS_SAN_STRT

This schedule is run daily @ 18.00 to set the BRDB system parameter (BRSS_SANITISE_STOP_YN) flag to 'N' ,such that jobs in schedule BRSS_SANITISE are allow to execute.

7.5.16.1 Dependencies

BRSS_SOD

7.5.16.2 Job BRSSX011_BRSS_SANITISE_START

Calls script BRSSX011.sh

$BRSS_SH/BRSSX011.sh -n "BRSS_SANITISE_STOP_YN" -t "T" -v "N"

7.5.16.2.1 Implementation

This job is implemented by a call to the schedule command with the relevant job name (BRSSX011.sh) and parameters.

7.5.16.2.2 Rerun Action

CONTINUE

7.5.17 Schedule BRSS_SANITISE

This schedule is run daily after schedule job BRSS_SAN_STRT.This job contains 7 jobs, Each of these jobs is the same BRSS process (BRSSC093) that is called with different parameters like (APSCR,APSADC,NWB,DCS,GREV,JRNL and RECOV) to cause it to truncate PCI data from a number of different BRSS tables.

7.5.17.1 Dependencies

BRSS_SAN_STRT

7.5.17.2 Job BRSSC093_APSCR

Calls script BRSSC093

$BRSS_PROC/BRSSC093 APSCR ^BRSSBDAY^

7.5.17.2.1 Implementation

The job is implemented as a call to the executable BRSSC093 with first parameter as "APSCR" and the second parameter as the business day (YYYYMMDD).

7.5.17.2.2 Rerun Action

CONTINUE

7.5.17.3 Job BRSSC093_ APSADC

Calls script BRSSC093

$BRSS_PROC/BRSSC093 APSADC ^BRSSBDAY^

7.5.17.3.1 Implementation

FUJ00234958

The job is implemented as a call to the executable BRSSC093 with first parameter as "APSADC" and the second parameter as the business day (YYYYMMDD).

7.5.17.3.2 Rerun Action

CONTINUE

7.5.17.4 Job BRSSC093_ NWB

Calls script BRSSC093

$BRSS_PROC/BRSSC093 NWB ^BRSSBDAY^

7.5.17.4.1 Implementation

The job is implemented as a call to the executable BRSSC093 with first parameter as "NWB" and the second parameter as the business day (YYYYMMDD).

7.5.17.4.2 Rerun Action

CONTINUE

7.5.17.5 Job BRSSC093_ DCS

Calls script BRSSC093

$BRSS_PROC/BRSSC093 DCS ^BRSSBDAY^

7.5.17.5.1 Implementation

The job is implemented as a call to the executable BRSSC093 with first parameter as "DCS" and the second parameter as the business day (YYYYMMDD).

7.5.17.5.2 Rerun Action

CONTINUE

7.5.17.6 Job BRSSC093_ GREV

Calls script BRSSC093

$BRSS_PROC/BRSSC093 GREV ^BRSSBDAY^

7.5.17.6.1 Implementation

The job is implemented as a call to the executable BRSSC093 with first parameter as "GREV" and the second parameter as the business day (YYYYMMDD).

7.5.17.6.2 Rerun Action

CONTINUE

7.5.17.7 Job BRSSC093_ JRNL

Calls script BRSSC093

$BRSS_PROC/BRSSC093 JRNL ^BRSSBDAY^

7.5.17.7.1 Implementation

The job is implemented as a call to the executable BRSSC093 with first parameter as "JRNL" and the second parameter as the business day (YYYYMMDD).

FUJ00234958

7.5.17.7.2 Rerun Action

CONTINUE

7.5.17.8 Job BRSSC093_ RECOV

Calls script BRSSC093

$BRSS_PROC/BRSSC093 RECOV ^BRSSBDAY^

7.5.17.8.1 Implementation

The job is implemented as a call to the executable BRSSC093 with first parameter as "RECOV" and the second parameter as the business day (YYYYMMDD).

7.5.17.8.2 Rerun Action

CONTINUE

7.5.18 Schedule BRSS_SAN_STOP

This schedule is run daily @ 01.00 to set the BRDB system parameter (BRSS_SANITISE_STOP_YN) flag to 'Y',such that jobs in schedule BRSS_SANITISE those are still running that shut-down with a success result.

7.5.18.1 Dependencies

BRSS_SAN_STRT

7.5.18.2 Job BRSSX011_BRSS_SANITISE_STOP

Calls script BRSSX011.sh

$BRSS_SH/BRSSX011.sh -n "BRSS_SANITISE_STOP_YN" -t "T" -v "Y"

7.5.18.2.1 Implementation

This job is implemented by a call to the schedule command with the relevant job name (BRSSX011.sh) and parameters.

7.5.18.2.2 Rerun Action

CONTINUE

FUJ00234958

8 Appendix C – Transaction Correction Templates

As the Transaction Correction Tool is no longer required (Ref# HNG-X CP2707 - Transaction Correction Tool - Decommissioning), the tool has been decommissioned in such a way that it cannot be used again.

<End of document>