



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



Document Title: HOST BRANCH DATABASE SUPPORT GUIDE

Document Reference: DES/APP/SPG/0001

Document Type: Support Guide

Release: HNG-X Release 1

Abstract: This Support Guide details information in support and maintenance of the Branch, the Branch Support and the Standby databases

Document Status: APPROVED

Andrew Aylward, HNG-X Host Development [29/10/2009]

Author & Dept: Chris Walker, HNG-X Host Development [09/09/2009]

Wing Pang, HNG-X Host Development [09/09/2009]

Tony Dolton, HNG-X Host Development [23/10/2009]

External Distribution: None

Approval Authorities:

Name	Role	Signature	Date
Mik Peach	Service Management – SSC		

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



0 Document Control

0.1 Table of Contents

0 DOCUMENT CONTROL.....	2
0.1 Table of Contents.....	2
0.2 Document History.....	9
0.3 Review Details.....	9
0.4 Associated Documents (Internal & External).....	10
0.5 Abbreviations.....	10
0.6 Glossary.....	11
0.7 Changes Expected.....	11
0.8 Accuracy.....	12
0.9 Copyright.....	12
1 INTRODUCTION.....	13
1.1 Document Overview.....	13
1.2 Scope.....	13
1.3 Assumptions.....	13
2 BRDB HOST PROCESSES.....	14
2.1 Approach used for Support Guide.....	14
2.1.1 Benefits of this Approach.....	14
2.2 Table of BRDB Host Processes.....	14
2.2.1 BRDB Environment Variables.....	16
2.3 BRDB Host Processes - Overview.....	17
2.3.1 Individual Programs.....	17
2.3.2 Interface Feeds.....	18
2.3.3 Data Aggregations.....	18
2.3.4 Support Differences.....	18
2.4 BRDB Host Processes – Support Details.....	19
2.4.1 Host Interface Feeds – additional support details.....	19
2.5 Error Logging/Notification.....	21
2.5.1 Program Return Code.....	21
2.5.2 Screen Output.....	22
2.5.3 Operational Exceptions.....	22
2.5.4 Process Control.....	22
2.5.5 Feed Data Exceptions.....	22
3 BRDB SCHEDULING.....	23
3.1 Multi-Instance Batch Jobs.....	23
3.2 Any Active Node Batch Jobs.....	24
3.3 Branch Database Jobs in other Schedules.....	24
3.4 Monitoring Jobs.....	24
3.5 Repeating/Daemon Processes.....	25
3.6 BRDB Schedules and Failover.....	25
3.7 Schedule BRDB_PAUSE_FEED3.....	25
3.7.1 Dependencies.....	25
3.7.2 Job BRDBX011_PAUSE_NPS_TT_COPY.....	25
3.7.3 Job BRDBX011_PAUSE_NPS_GREV_COPY.....	26



3.8 Schedule BRDB_STARTUP.....	26
3.8.1 Dependencies.....	26
3.8.2 Job BRDBC001.....	26
3.9 Schedule BRDB_START_FEED3.....	27
3.9.1 Dependencies.....	27
3.9.2 Job BRDBX011_START_NPS_TT_COPY.....	27
3.9.3 Job BRDBX011_START_NPS_GREV_COPY.....	27
3.10 Schedule BRDB_TT_TO_NPS3.....	27
3.10.1 Dependencies.....	27
3.10.2 Job BRDBX003_TT_TO_NPS_1..4_NOPAGE.....	28
3.11 Schedule BRDB_GREV_NPS3.....	28
3.11.1 Dependencies.....	28
3.11.2 Job BRDBX003_GREV_TO_NPS_1..4_NOPAGE.....	28
3.12 Schedule BRDB_PAUSE_FEED1.....	28
3.12.1 Dependencies.....	28
3.12.2 Job BRDBX011_PAUSE_NPS_TT_COPY.....	29
3.12.3 Job BRDBX011_PAUSE_NPS_GREV_COPY.....	29
3.13 Schedule BRDB_COMPLETE.....	29
3.13.1 Dependencies.....	29
3.13.2 Job CREATE_BRDB_COMPLETE_FLAG.....	29
3.14 Schedule SOD.....	30
3.14.1 Dependencies.....	30
3.14.2 Job (Unnamed).....	30
3.14.3 Job (Unnamed).....	30
3.15 Schedule BRDB_START_FEED1.....	30
3.15.1 Dependencies.....	30
3.15.2 Job BRDBX011_START_NPS_TT_COPY.....	30
3.15.3 Job BRDBX011_START_NPS_GREV_COPY.....	31
3.16 Schedule BRDB_START_LFS.....	31
3.16.1 Dependencies.....	31
3.16.2 Job BRDBX011_START_LFS_PCOL_COPY.....	31
3.16.3 Job BRDBX011_START_LFS_PDEL_COPY.....	32
3.17 Schedule BRDB_TT_TO_NPS1.....	32
3.17.1 Dependencies.....	32
3.17.2 Job BRDBX003_TT_TO_NPS_1..4_NOPAGE.....	32
3.18 Schedule BRDB_GREV_NPS1.....	32
3.18.1 Dependencies.....	32
3.18.2 Job BRDBX003_GREV_TO_NPS_1..4_NOPAGE.....	32
3.19 Schedule BRDB_PCL_TO_LFS.....	33
3.19.1 Dependencies.....	33
3.19.2 Job BRDBX003_PCOL_TO_LFS_1..4_NOPAGE.....	33
3.20 Schedule BRDB_PDL_TO_LFS.....	33
3.20.1 Dependencies.....	33
3.20.2 Job BRDBX003_PDEL_TO_LFS_1..4_NOPAGE.....	33
3.21 Schedule SOD.....	34
3.21.1 Dependencies.....	34
3.21.2 Job COMPLETE.....	34
3.22 Schedule BRDB_REF_DATA_SLA.....	34
3.22.1 Dependencies.....	34
3.22.2 Job BRDBX032_BRDB_REF_DATA_SLA.....	34
3.23 Schedule BRDB_ONCH_AGG.....	34
3.23.1 Dependencies.....	35
3.23.2 Job BRDBX007_ONCH_AGG_1..4.....	35
3.23.3 Job BRDBC008_CHECK_ONCH_AGG.....	35
3.24 Schedule BRDB_CSH_TO_LFS.....	35
3.24.1 Dependencies.....	35



3.24.2 Job BRDBX003_CASH_TO_LFS_1...4.....	35
3.24.3 Job BRDBC008_CHECK_CASH_TO_LFS.....	36
3.25 Schedule BRDB_FROM_EMDB.....	36
3.25.1 Dependencies.....	36
3.25.2 Job BRDBX003_BRDATA_FROM_EMDB.....	36
3.26 Schedule BRDB_PAUSE_LFS.....	36
3.26.1 Dependencies.....	37
3.26.2 Job BRDBX011_PAUSE_LFS_PCOL_COPY.....	37
3.26.3 Job BRDBX011_PAUSE_LFS_PDEL_COPY.....	37
3.27 Schedule BRDB_EPOS_TO_TPS.....	37
3.27.1 Dependencies.....	37
3.27.2 Job BRDBX003_EPOSS_TO_TPS_1...4.....	37
3.27.3 Job BRDBC008_CHECK_EPOSS_TO_TPS.....	38
3.28 Schedule BRDB_AP_S_TO_TPS.....	38
3.28.1 Dependencies.....	38
3.28.2 Job BRDBX003_AP_S_TO_TPS_1...4.....	38
3.28.3 Job BRDBC008_CHECK_AP_S_TO_TPS.....	39
3.29 Schedule BRDB_NWB_TO_TPS.....	39
3.29.1 Dependencies.....	39
3.29.2 Job BRDBX003_NWB_TO_TPS_1...4.....	39
3.29.3 Job BRDBC008_CHECK_NWB_TO_TPS.....	39
3.30 Schedule BRDB_DCS_TO_TPS.....	40
3.30.1 Dependencies.....	40
3.30.2 Job BRDBX003_DCS_TO_TPS_1...4.....	40
3.30.3 Job BRDBC008_CHECK_DCS_TO_TPS.....	40
3.31 Schedule BRDB_BDC_TO_TPS.....	40
3.31.1 Dependencies.....	41
3.31.2 Job BRDBX003_BUREAU_TO_TPS_1...4.....	41
3.31.3 Job BRDBC008_CHECK_BUREAU_TO_TPS.....	41
3.32 Schedule BRDB_EVT_TO_TPS.....	41
3.32.1 Dependencies.....	41
3.32.2 Job BRDBX003_EVENTS_TO_TPS_1...4.....	41
3.32.3 Job BRDBC008_CHECK_EVENTS_TO_TPS.....	42
3.33 Schedule BRDB_COFS_TO_TPS.....	42
3.33.1 Dependencies.....	42
3.33.2 Job BRDBX003_COFF_SUMM_TO_TPS_1...4.....	42
3.33.3 Job BRDBC008_CHECK_COFF_SUMM_TO_TPS.....	43
3.34 Schedule BRDB_TC_FROM_TPS.....	43
3.34.1 Dependencies.....	43
3.34.2 Job BRDBX003_TC_FROM_EMDB.....	43
3.35 Schedule BRDB_TPS_COMPL.....	43
3.35.1 Dependencies.....	43
3.35.2 Job COMPLETE.....	43
3.36 Schedule BRDB_TPS_TOTALS.....	44
3.36.1 Dependencies.....	44
3.36.2 Job BRDBX007_TPS_TXN_TOTALS_1...4.....	44
3.36.3 Job BRDBC008_CHECK_TPS_TXN_TOTALS.....	44
3.37 Schedule BRDB_TOTL_TO_TPS.....	44
3.37.1 Dependencies.....	45
3.37.2 Job BRDBX003_TXN_TOTALS_TO_TPS_1...4.....	45
3.37.3 Job BRDBC008_CHECK_TXN_TOTALS_TO_TPS.....	45
3.38 Schedule BRDB_AP_S_TOTALS.....	45
3.38.1 Dependencies.....	45
3.38.2 Job BRDBX007_AP_S_TXN_TOTALS_1...4.....	46
3.38.3 Job BRDBC008_CHECK_AP_S_TXN_TOTALS.....	46
3.39 Schedule BRDB_TOTL_TO_AP_S.....	46



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



3.39.1	Dependencies.....	46
3.39.2	Job BRDBX003_TXN_TOTALS_TO_AP_S_1..4.....	46
3.39.3	Job BRDBC008_CHECK_TXN_TOTALS_TO_AP_S.....	47
3.40	Schedule BRDB_TXNS_TO_AP_S.....	47
3.40.1	Dependencies.....	47
3.40.2	Job BRDBX003_TXNS_TO_AP_S_1..4.....	47
3.40.3	Job BRDBC008_CHECK_TXNS_TO_AP_S.....	47
3.41	Schedule BRDB_AP_S_COMPL.....	48
3.41.1	Dependencies.....	48
3.41.2	Job COMPLETE.....	48
3.42	Schedule BRDB_NWB_TO_DRS.....	48
3.42.1	Dependencies.....	48
3.42.2	Job BRDBX003_NWB_TO_DRS_1..4.....	48
3.42.3	Job BRDBC008_CHECK_NWB_TO_DRS.....	49
3.43	Schedule BRDB_DCS_TO_DRS.....	49
3.43.1	Dependencies.....	49
3.43.2	Job BRDBX003_DCS_TO_DRS_1..4.....	49
3.43.3	Job BRDBC008_CHECK_DCS_TO_DRS.....	49
3.44	Schedule BRDB_DRS_COMPL.....	50
3.44.1	Dependencies.....	50
3.44.2	Job COMPLETE.....	50
3.45	Schedule BRDB_HYD_TP_EPOS.....	50
3.45.1	Dependencies.....	50
3.45.2	Job BRDBX003_EPOSS_FROM_TPS_1..4.....	50
3.45.3	Job BRDBC008_CHECK_EPOSS_FROM_TPS.....	51
3.46	Schedule BRDB_HYD_TP_AP.....	51
3.46.1	Dependencies.....	51
3.46.2	Job BRDBX003_AP_FROM_TPS_1..4.....	51
3.46.3	Job BRDBC008_CHECK_AP_FROM_TPS.....	51
3.47	Schedule BRDB_HYD_TP_BDC.....	52
3.47.1	Dependencies.....	52
3.47.2	Job BRDBX003_BUREAU_FROM_TPS_1..4.....	52
3.47.3	Job BRDBC008_CHECK_BUREAU_FROM_TPS.....	52
3.48	Schedule BRDB_HYD_TP_NWB.....	52
3.48.1	Dependencies.....	52
3.48.2	Job BRDBX003_NWB_FROM_TPS_1..4.....	53
3.48.3	Job BRDBC008_CHECK_NWB_FROM_TPS.....	53
3.49	Schedule BRDB_HYD_TP_DCS.....	53
3.49.1	Dependencies.....	53
3.49.2	Job BRDBX003_DCS_FROM_TPS_1..4.....	53
3.49.3	Job BRDBC008_CHECK_DCS_FROM_TPS.....	54
3.50	Schedule BRDB_HYD_ONGOING.....	54
3.50.1	Dependencies.....	54
3.50.2	Job BRDBX003_RECON_FROM_TPS.....	54
3.51	Schedule BRDB_HYD_INDAY.....	54
3.51.1	Dependencies.....	54
3.51.2	Job BRDBX003_INDAY_FROM_TPS.....	55
3.52	Schedule BRDB_HYD_PREPROC.....	55
3.52.1	Dependencies.....	55
3.52.2	Job BRDBX033_BRDB_HYDRA_PROCESSING_1..4.....	55
3.52.3	Job BRDBC008_BRDBX033_PREP_RECON.....	55
3.53	Schedule BRDB_XFR_COMPL.....	56
3.53.1	Dependencies.....	56
3.53.2	Job COMPLETE.....	56
3.54	Schedule BRDB_FEED_ERRORS.....	56
3.54.1	Dependencies.....	56



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



3.54.2	Job BRDBX007_RAISE_FEED_DATA_EXCEPTIONS.....	56
3.55	Schedule BRDB_NCU_TXN_AGG.....	57
3.55.1	Dependencies.....	57
3.55.2	Job BRDBX007_NON_CUMU_TXN_TOTALS_1..4.....	57
3.55.3	Job BRDBC008_CHECK_NON_CUMU_TXN_AGGR.....	57
3.56	Schedule BRDB_CU_TXN_AGG.....	57
3.56.1	Dependencies.....	58
3.56.2	Job BRDBX007_CUMU_TXN_AGGR_1..4.....	58
3.56.3	Job BRDBC008_CHECK_CUMU_TXN_AGGR.....	58
3.57	Schedule BRDB_BBNI_MAINT.....	58
3.57.1	Dependencies.....	58
3.57.2	Job BRDBX031_JSON_USN_SSN.....	58
3.58	Schedule BRDB_SUMMARY_DTE.....	59
3.58.1	Dependencies.....	59
3.58.2	Job BRDBX011_SET_DAILY_SUMMARY_DATE.....	59
3.59	Schedule BRDB_HYDRA_PROC.....	59
3.59.1	Dependencies.....	59
3.59.2	Job BRDBX030_BRDB_HYDRA_PROCESSING_1..4.....	59
3.59.3	Job BRDBC008_BRDBX030_RECON.....	60
3.59.4	Job BRDBC008_BRDBX030_INDAY.....	60
3.60	Schedule BRDB_GEN_REPORT.....	60
3.60.1	Dependencies.....	60
3.60.2	Job GENERIC_CREATE_REPORT_VIEWS.....	60
3.60.3	Job GENERIC_CREATE_RECON_REPORTS.....	61
3.61	Schedule BRDB_TO_DWH.....	61
3.61.1	Dependencies.....	61
3.61.2	Job BRDBX020_BRDB_XFER_TO_DWH.....	61
3.62	Schedule BRDB_AGG_COMPL.....	62
3.62.1	Dependencies.....	62
3.62.2	Job COMPLETE.....	62
3.63	Schedule BRDB_FROM_RDDS.....	62
3.63.1	Dependencies.....	62
3.63.2	Job BRDBX003_REFDATA_FROM_RDDS.....	62
3.64	Schedule BRDB_FROM_TPS.....	63
3.64.1	Dependencies.....	63
3.64.2	Job BRDBX003_REFDATA_FROM_TPS.....	63
3.65	Schedule BRDB_AUD_FEED.....	63
3.65.1	Dependencies.....	63
3.65.2	Job BRDBC002_AUDIT_1..4.....	64
3.65.3	Job BRDBC008_CHECK_AUDIT_FEED.....	64
3.65.4	Job BRDBC033_AUDIT.....	64
3.66	Schedule BRDB_ORA_STATS.....	64
3.66.1	Dependencies.....	64
3.66.2	Job BRDBX005_SCHEMA.....	64
3.67	Schedule BRDB_ADMIN.....	65
3.67.1	Dependencies.....	65
3.67.2	Job BRDBC004.....	65
3.67.3	Job BRDBX006.....	65
3.67.4	Job BRDB_HKP_ORAFILES1.....	65
3.67.5	Job BRDB_HKP_ORAFILES2.....	66
3.68	Schedule BRDB_PAUSE_FEED2.....	66
3.68.1	Dependencies.....	66
3.68.2	Job BRDBX011_PAUSE_NPS_TT_COPY.....	66
3.68.3	Job BRDBX011_PAUSE_NPS_GREV_COPY.....	66
3.69	Schedule BRDB_EOD.....	67
3.69.1	Dependencies.....	67



3.69.2	Job BRDBC009.....	67
3.70	Schedule BRDB_START_FEED2.....	67
3.70.1	Dependencies.....	67
3.70.2	Job BRDBX011_START_NPS_TT_COPY.....	67
3.70.3	Job BRDBX011_START_NPS_GREV_COPY.....	68
3.71	Schedule BRDB_TT_TO_NPS2.....	68
3.71.1	Dependencies.....	68
3.71.2	Job BRDBX003_TT_TO_NPS_1..4_NOPAGE.....	68
3.72	Schedule BRDB_GREV_NPS2.....	68
3.72.1	Dependencies.....	69
3.72.2	Job BRDBX003_GREV_TO_NPS_1..4_NOPAGE.....	69
3.73	Schedule BRDB_START_BKP.....	69
3.73.1	Dependencies.....	69
3.73.2	Job COMPLETE.....	69
3.74	Schedule BRDB_BACKUP_0.....	69
3.74.1	Dependencies.....	69
3.74.2	Job BRDB_LVL0_BACKUP.....	69
3.75	Schedule BRDB_BACKUP_1.....	70
3.75.1	Dependencies.....	70
3.75.2	Job BRDB_LVL1_BACKUP.....	70
3.76	Schedule BRDB_BKP_COMPLETE.....	70
3.76.1	Dependencies.....	70
3.76.2	Job CREATE_BRDB_COMPLETE_FLAG.....	70
3.77	Schedule BRDB_MONITOR.....	71
3.77.1	Dependencies.....	71
3.77.2	Job BRDB_MON_STARTUP.....	71
3.77.3	Job BRDB_MON_PAUSE_FEED1.....	71
3.77.4	Job BRDB_MON_AUD_FEED.....	71
3.77.5	Job BRDB_MON_EOD.....	72
4	BACKUP AND RECOVERY.....	73
4.1	BRDB & BRSS Backups.....	73
4.1.1	Backup Duration.....	73
4.2	Restoring files with RMAN.....	73
4.3	Failure and Recovery.....	73
4.3.1	Escalation and Notification.....	74
4.3.2	Media Failure and Recovery.....	74
4.3.3	Instance/Node Failure and Recovery.....	74
5	GENERAL AND TROUBLESHOOTING NOTES.....	79
5.1	Database.....	79
5.1.1	Oracle Database Listeners.....	79
5.1.2	General Recommendations.....	82
5.1.3	Password Management.....	82
5.2	Backups.....	84
5.2.1	Database Backups.....	84
5.2.2	Disk Backups.....	84
5.3	Partition Management.....	84
5.3.1	Introduction.....	84
5.3.2	Assumptions.....	85
5.3.3	Overview.....	85
5.3.4	Troubleshooting.....	85
5.4	Standby Database.....	89
5.4.1	Introduction.....	89



5.4.2 Assumptions.....	89
5.4.3 Troubleshooting.....	89
5.5 Oracle Streams.....	91
5.5.1 Introduction.....	91
5.5.2 Assumptions.....	91
5.5.3 Overview.....	91
5.5.4 Troubleshooting.....	91
5.6 BRDBX015 – Transaction Correction Tool.....	104
5.6.1 Parameters.....	104
5.6.2 Scheduling.....	104
5.6.3 Sample output.....	104
5.6.4 Diagnostics.....	105
6 APPENDIX A – STANDBY DATABASE.....	106
6.1 Oracle Data Guard Broker (DGMGR) Failover.....	106
6.2 SQL*Plus Failover.....	113
6.3 Standby Database Re-instantiation.....	115
6.4 Opening Standby Database “READ ONLY”.....	118
6.5 Standby Cluster – Software Installation.....	120
7 APPENDIX B – BRANCH SUPPORT.....	122
7.1 Cleanup and Re-instantiation of Oracle Streams.....	122
7.1.1 Assumptions.....	122
7.1.2 Cleanup and Re-instantiation Procedure.....	123
7.2 Managing Streams Lag.....	136
7.2.1 Context and Assumptions.....	136
7.2.2 Lag Evaluation and Escalation.....	136
7.2.3 Lag Assessment and Action Procedure.....	137
7.2.4 Post Lag Action Procedure.....	141
8 APPENDIX C – TRANSACTION CORRECTION TEMPLATES.....	144
8.1 Template for BRDB_RXAPS_TRANSACTIONS.....	144
8.2 Template for BRDB_RXBUREAU_TRANSACTIONS.....	146
8.3 Template for BRDB_RXCUTOFF_SUMMARIES.....	147
8.4 Template for BRDB_RXDCS_TRANSACTIONS.....	149
8.5 Template for BRDB_RXEPOSS_EVENTS.....	151
8.6 Template for BRDB_RXEPOSS_TRANSACTIONS.....	152
8.7 Template for BRDB_RXNWB_TRANSACTIONS.....	154
8.8 Template for BRDB_RXREP_EVENT_DATA.....	156
8.9 Template for BRDB_RXREP_SESSION_DATA.....	157



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



0.2 Document History

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK Reference
0.1	22 nd June 2009	Initial Version	N/A
0.2	18 th September 2009	First major update to all sections	N/A
0.3	23 rd October 2009	Updated with schedule details and other information.	N/A
0.4	29 th October 2009	Updated with general review comments and additions to Streams and Standby procedures.	N/A
0.5	29 th October 2009	Updated with Streams related information.	N/A

0.3 Review Details

Review Comments by :	2 nd October 2009
Review Comments to :	Andrew Aylward & <u>RMGADocumentManagement</u> GRO
Mandatory Review	
Role	Name
Service Management – SSC	Mik Peach
Infrastructure Design	Pat Lywood
Core Services	Andrew Gibson
Optional Review	
Role	Name
Service Support	Kirsty Gallacher
HNG-X Service Transition	Graham Welsh
Architecture	David Johns
Head of Service Management	Steve Denham
Service Support	Mike Stewart
Service Support	Mike Woolgar
Business Continuity	Adam Parker
HNG-X Host Development	Steve Goddard
HNG-X Host Development	Wing Pang
HNG-X Host Development	Andy Beardmore
HNG-X Host Development	Chris Walker
HNG-X Host Development	David Harrison
Core Services	Wayne Calvert
Core Services	Paul Simpson
Core Services	Paul Stewart
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)	2.0	16-Apr-07	Fujitsu Post Office Account HNG-X Document Template - PORTRAIT	Dimensions
DES/APP/HLD/0020	TBA		Branch Database High Level Design	Dimensions
DES/APP/LLD/0152	TBA		Branch Database Low Level Design	Dimensions
DES/APP/HLD/0021	TBA		Branch Database Scheduling High Level Design	Dimensions
DES/APP/HLD/0023	TBA		Branch Support Database High Level Design	Dimensions
DES/APP/LLD/0151	TBA		Branch Support Database Low Level Design	Dimensions
DES/APP/HLD/0025	TBA		Branch Support Database Scheduling High Level Design	Dimensions

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

0.5 Abbreviations

Abbreviation	Definition
ASM	Automatic Storage Management
BAL	Branch Access Layer
BDB	Acronym for Branch Database
BDS	Acronym for Branch Standby Database
BRDB	Branch Database Oracle SID
BRS	Acronym for Branch Support Database
CRS	Oracle Cluster Ready Services
FAN	Oracle Fast Application Notification
HLD	High Level Design
ITM	IBM Tivoli Manager
JSN	Journal Sequence Number
LCR	Logical change record (generated by the Streams capture process)
LPAN	Logical Processing Area Network
OCFS	Oracle Cluster File System
OCR	Oracle Cluster Registry
PAN	Processing Area Network
RFS	Oracle Remote File Server (a process)
RHEL	Red Hat Enterprise Linux
RMAN	Oracle Recovery Manager



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



SAN	Storage Area Network
SCN	Oracle System Change Number
SQL	Structured Query Language
SSN	Session Sequence Number
USN	User Sequence Number (in the context of the counter user)

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,
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.
Host System	The collection of host systems including TPS, APS, DRS, LFS, 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.
Node	Any device connected to a network such as a server. In the document, the term 'Node' includes the Oracle Instance.
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 Copyright

© Copyright Fujitsu Limited 2009. All rights reserved. No part of this document may be reproduced, stored or transmitted in any form without the prior written permission of Fujitsu.



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. As long as it does not begin to fall behind by 8 to 10 hours, the Branch Support Database will remain an effective querying database.

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 [DES/APP/LLD/0151].

1.3 Assumptions

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



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.

2.1.1 Benefits of this Approach

The approach described above is considered to be highly preferable to that found in some previous support guides (for other applications), where large chunks of information have simply been copied from a version of the design documents to a support document. Such duplication merely creates another place to search for the required information, and doubles the amount of maintenance required when the underlying information needs to be changed. Also, where inevitable discrepancies arise, it's not clear which of the documents is up to date or correct.

The approach employed here makes good use of the existing design documents, and preserves their status as the main source of fact.

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_AP5_TXN_FROM_TPS	BRDB APS transactions from TPS feed
4	BRDBX003.sh	BRDB_AP5_TXN_TO_AP5	BRDB APS transactions to AP5 feed
5	BRDBX003.sh	BRDB_AP5_TXN_TO_TPS	BRDB APS transactions to TPS feed
6	BRDBX003.sh	BRDB_BDC_TXN_FROM_TPS	BRDB BDC transactions from TPS feed
7	BRDBX003.sh	BRDB_BDC_TXN_TO_TPS	BRDB BDC transactions to TPS feed



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



8	BRDBX003.sh	BRDB_CASH_TO_LFS	BRDB Cash Declarations to LFS feed
9	BRDBX003.sh	BRDB_CNTR_REF_FROM_RDDS	BRDB Counter Reference Data from RDDS feed
10	BRDBX003.sh	BRDB_CUTOFF_SUMM_TO_TPS	BRDB Cut Off Summaries to TPS feed
11	BRDBX003.sh	BRDB_DCS_TXN_FROM_TPS	BRDB DCS transactions from TPS feed
12	BRDBX003.sh	BRDB_DCS_TXN_TO_DRSS	BRDB DCS transactions to DRSS feed
13	BRDBX003.sh	BRDB_DCS_TXN_TO_TPS	BRDB DCS transactions to TPS feed
14	BRDBX003.sh	BRDB_EMDB_INTERFACE	BRDB Estate Management Interface feed
15	BRDBX003.sh	BRDB_EPOSS_EVNT_TO_TPS	BRDB EPOSS events to TPS feed
16	BRDBX003.sh	BRDB_EPOSS_TXN_FROM_TPS	BRDB EPOSS transactions from TPS feed
17	BRDBX003.sh	BRDB_EPOSS_TXN_TO_TPS	BRDB EPOSS transactions to TPS feed
18	BRDBX003.sh	BRDB_HOST_REF_FROM_RDDS	BRDB Host Reference Data from RDDS feed
19	BRDBX003.sh	BRDB_INDAY_XML_FROM_TPS	BRDB In-Day Migration Blob from TPS feed
20	BRDBX003.sh	BRDB_MEMOS_FROM_RDDS	BRDB Desktop Memos from RDDS feed
21	BRDBX003.sh	BRDB_NWB_TXN_FROM_TPS	BRDB NWB transactions from TPS feed
22	BRDBX003.sh	BRDB_NWB_TXN_TO_DRSS	BRDB NWB transactions to DRSS feed
23	BRDBX003.sh	BRDB_NWB_TXN_TO_TPS	BRDB NWB transactions to TPS feed
24	BRDBX003.sh	BRDB_PCOL_TO_LFS	BRDB Pouch Collections to LFS feed
25	BRDBX003.sh	BRDB_PDEL_TO_LFS	BRDB Pouch Deliveries to LFS feed
26	BRDBX003.sh	BRDB_PLO_FROM_LFS	BRDB Planned Order details from LFS feed
27	BRDBX003.sh	BRDB_RDC_FROM_LFS	BRDB Replenishment Delivery details from LFS feed
28	BRDBX003.sh	BRDB_RECON_XML_FROM_TPS	BRDB Reconciliation Blob from TPS feed
29	BRDBX003.sh	BRDB_REF_COPY_FROM_TPS	BRDB Outlets/Transaction Modes from TPS feed
30	BRDBX003.sh	BRDB_REV_TXN_TO_NPS	BRDB Reversal Records to NPS feed
31	BRDBX003.sh	BRDB_TT_TXN_TO_NPS	BRDB Track and Trace Records to NPS feed
32	BRDBX003.sh	BRDB_TXN_CORR_FROM_TPS	BRDB Transaction Corrections from TPS feed
33	BRDBX003.sh	BRDB_TXN_TOT_TO_APSS	BRDB Transaction Totals to APSS feed
34	BRDBX003.sh	BRDB_TXN_TOT_TO_TPS	BRDB Transaction Totals to TPS feed
35	BRDBC004	BRDBC004	Audit, Archive, Purge
36	BRDBX005.sh	BRDBX005.sh	Gather Optimiser Statistics
37	BRDBX006.sh	BRDBX006	File Housekeeping
38	BRDBX007.sh	BRDB_APSS_TXN_TOTALS	Data aggregation to calculate APSS transaction totals
39	BRDBX007.sh	BRDB_CUMU_TXN_AGGR	Data aggregation for daily cumulative summary
40	BRDBX007.sh	BRDB_NON_CUMU_TXN_AGGR	Data aggregation for daily summary



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



41	BRDBX007.sh	BRDB_TPS_TXN_TOTALS	Data aggregation to calculate outlet transaction totals
42	BRDBX007.sh	OVERNIGHT_CASH_ON_HAND	Data aggregation to calculate ONCH figures.
43	BRDBX007.sh	RAISE_FEED_DATA_EXCEPTIONS	Inserts into operational exceptions if Feed data exceptions
44	BRDBC008	BRDBC008	Check Job Completion
45	BRDBC009	BRDBC009	End Of Day
46	BRDBX011.sh	BRDBX011	Updates system parameters
47	BRDBX015.sh	None	Transaction correction tool
48	BRDBX020.sh*	None	File transfer for BRDB Branch Migration Status data feed
49	BRDBX021.sh	None	Pause or restart Oracle Streams propagation
50	BRDBX030.sh	BRDBX030_INDAY	Hydra XML processing (INDAY)
51	BRDBX030.sh	BRDBX030_RECON	Hydra XML processing (RECON)
52	BRDBX031.sh*	BRDBX031	Reset JSN, USN and SSN
53	BRDBX032.sh*	BRDB_REF_DATA_SLAS	Reference Data SLAs
54	BRDBC033*	BRDBC033	Transaction Correction Journal Auditing
55	GREPX001.sh*	GREPX001	Create generic views for reporting
56	GREPX002.sh*	GREPX002	Create generic reports

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).

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 *brdbb1v1*. 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/10.2.0/db_1
OMNIHOME	/opt/cool/omnibus
INPUTRC	/etc/inputrc
G_BROKEN_FILERAMES	1
ORACLE_SID	BRDB1
LANG	C
NETCOOL_LICENSE_FILE	27000@ltpbdb001
BRDB_CONNECT_STR	BRDB
LOGNAME	brdbblv1
BRDB_SH	/app_sw/brdb/sh
HISTSIZE	1000
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), although some are new for the Branch Database e.g. "Hydra XML processing" (BRDBX030). They are invoked by a direct call (from a Linux shell) to an executable.

2.3.2 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 currently over 30 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, line 17 of the Table of BRDB Host Processes above shows that the Feed of EPOSS transactions from BRDB to TPS, is performed by a call to BRDBX003.sh with a first parameter of "BRDB_EPOSS_TXN_TO_TPS".

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

PKG_<Feed name> e.g. PKG_BRDB_EPOSS_TXN_TO_TPS

2.3.3 Data Aggregations

These are also new for the Branch Database, and are similar to the Interface Feeds in that different processes (currently numbering six) 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 such as 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_EPOSS_TXN_TO_TPS COMPILE;

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.

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

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. Table BRDB_PROCESS_CONTROL 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.



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). 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). 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 (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 automatically re-runs the main Feed processes (that are scheduled to run on all Nodes) in order to process any FAD Hashes that have been re-allocated from a failed Node.

Thus, despite a potentially serious event such as a Node failure, all of the multi-Node processes should complete without requiring intervention from the support team.

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 acts as an alert to the operators, 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, and is currently set to 1000.



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 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.

2.5 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.

In addition, table BRDB_HYDRA_EXCEPTIONS is used to log errors associated with Hydra processing. Modules BRDBX030 and BRDBX033 log errors which results in the branch not being processed in the



BRDB_OPERATIONAL_EXCEPTIONS table, whilst errors where processing continues for the branch are logged in the BRDB_HYDRA_EXCEPTIONS table.

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.

2.5.5 Feed Data Exceptions

See section 2.4.1.5 (Data Exceptions) for details.



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 3.6 describe features of the schedule as a whole, and sections 3.7 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.

In the schedule listings from sections 3.7 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 follows:

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	Follows Job-Instance-1...4	On success or failure continue
<i>Job to be run on an active node</i>		
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	



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 <i>Job to be run on an active node</i>	Follows Job-Instance-1...4-rerun	On failure Alert Operations
Schedule-complete	Follows Check-DB-Job, Check-DB-Job-rerun	

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 3.7 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 deleted.

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, these are called from other applications such as LFS and RDDS. This section lists the schedules concerned.

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

LFS: Scheduling HLD is DES/APP/HLD/0088

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

Currently there are four BRDB Host Interface Feeds that are run as 'daemon' processes within the daily schedule: The 'Track and Trace' and 'Reversals' Feeds to NPS, and the 'Pouch Collections' and 'Pouch Deliveries' Feeds to LFS.

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_REV_TXN_TO_NPS_STOP_YN
BRDB_TT_TXN_TO_NPS_STOP_YN
BRDB_PDEL_TO_LFS_STOP_YN
BRDB_PCOL_TO_LFS_STOP_YN

When this parameter is set to 'Y' (from within the schedule using BRDBX011.sh) 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.

There may be a need to do this manually, and running BRDBX011.sh with the appropriate parameters will accomplish this without a problem.

3.6 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.

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

3.7 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

3.7.1 Dependencies

Schedule BRDB_PAUSE_FEED3 depends on the completion of schedule BRDB_BKP_COMPL.

3.7.2 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).

3.7.2.1 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".

3.7.2.2 Rerun Action

Alert Operations on failure.



3.7.3 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).

3.7.3.1 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".

3.7.3.2 Rerun Action

Alert Operations on failure.

3.8 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.

3.8.1 Dependencies

Schedule BRDB_STARTUP depends on the completion of schedule BRDB_PAUSE_FEED3.

3.8.2 Job BRDBC001

This job runs the BRDB start of day utility.

3.8.2.1 Implementation

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

3.8.2.2 Rerun Action

Alert Operations on failure??.

3.9 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:

BRDBX011_START_NPS_TT_COPY
BRDBX011_START_NPS_GREV_COPY

3.9.1 Dependencies

Schedule BRDB_START_FEED3 depends on the completion of schedule BRDB_STARTUP.



3.9.2 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).

3.9.2.1 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".

3.9.2.2 Rerun Action

Alert Operations on failure.

3.9.3 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).

3.9.3.1 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".

3.9.3.2 Rerun Action

Alert Operations on failure.

3.10 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.

3.10.1 Dependencies

Schedule BRDB_TT_TO_NPS3 depends on the completion of schedule BRDB_START_FEED3.

3.10.2 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.

3.10.1.1 Implementation

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

3.10.1.2 Rerun Action

Rerun on failure.



3.11 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.

3.11.1 Dependencies

Schedule BRDB_GREV_NPS3 depends on the completion of schedule BRDB_START_FEED3.

3.11.2 Job BRDBX003_GREV_TO_NPS_1...4_NOPAGE

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

3.11.2.1 Implementation

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

3.11.2.2 Rerun Action

Rerun on failure.

3.12 Schedule BRDB_PAUSE_FEED1

This schedule is run daily at 07:50. It stops the two NPS copy processes prior to the start of day processing. 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

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.

3.12.1 Dependencies

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

3.12.2 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).

3.12.2.1 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".

3.12.2.2 Rerun Action

Alert Operations on failure.



3.12.3 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).

3.12.3.1 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".

3.12.3.2 Rerun Action

Alert Operations on failure.

3.13 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.

3.13.1 Dependencies

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

3.13.2 Job CREATE_BRDB_COMPLETE_FLAG

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

3.13.2.1 Implementation

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

3.13.2.2 Rerun Action

***** Prompts for rerun – action? ****

3.14 Schedule SOD

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

3.14.1 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.

3.14.2 Job (Unnamed)

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



3.14.2.1 Implementation

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

3.14.2.2 Rerun Action

Alert Operations on failure?

3.14.3 Job (Unnamed)

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

3.14.3.1 Implementation

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

3.14.3.2 Rerun Action

Alert Operations on failure?

3.15 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

3.15.1 Dependencies

Schedule BRDB_START_FEED1 depends on the completion of schedule BRDB_SOD.

3.15.2 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).

3.15.2.1 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”.

3.15.2.2 Rerun Action

Alert Operations on failure.

3.15.3 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).



3.15.3.1 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".

3.15.3.2 Rerun Action

Alert Operations on failure.

3.16 Schedule BRDB_START_LFS

This schedule is run daily at 08:02. It prepares for the running of the two LFS copy processes by reversing the changes that stop them from running. 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_LFS_PCOL_COPY
BRDBX011_START_LFS_PDEL_COPY

3.16.1 Dependencies

Schedule BRDB_START_LFS depends on the completion of schedule BRDB_SOD.

3.16.2 Job BRDBX011_START_LFS_PCOL_COPY

This job prepares for the starting of the copying of Pouch Collections to LFS, 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_PCOL_TO_LFS_STOP_YN and value "N".

3.16.2.2 Rerun Action

Alert Operations on failure.

3.16.3 Job BRDBX011_START_LFS_PDEL_COPY

This job prepares for the starting of the copying of Pouch Deliveries to LFS, by setting a system parameter (see section 3.5).

3.16.1.1 Implementation

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

3.16.1.2 Rerun Action

Alert Operations on failure.



3.17 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.

3.17.1 Dependencies

Schedule BRDB_TT_TO_NPS1 depends on the completion of schedule BRDB_START_FEED1.

3.17.2 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.

3.17.2.1 Implementation

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

3.17.2.2 Rerun Action

Rerun on failure.

3.18 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.

3.18.1 Dependencies

Schedule BRDB_GREV_NPS1 depends on the completion of schedule BRDB_START_FEED1.

3.18.2 Job BRDBX003_GREV_TO_NPS_1...4_NOPAGE

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

3.18.2.1 Implementation

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

3.18.2.2 Rerun Action

Rerun on failure.

3.19 Schedule BRDB_PCL_TO_LFS

This schedule is run daily at 08:05 to start the Pouch Collection to LFS data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_PCOL_TO_LFS_1...4_NOPAGE.



3.19.1 Dependencies

Schedule BRDB_PCL_TO_LFS depends on the completion of schedule BRDB_START_LFS.

3.19.2 Job BRDBX003_PCOL_TO_LFS_1...4_NOPAGE

These jobs (one per node) start the feed that copies the Pouch Collections to LFS.

3.19.2.1 Implementation

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

3.19.2.2 Rerun Action

Rerun on failure

3.20 Schedule BRDB_PDL_TO_LFS

This schedule is run daily at 08:05 to start the Pouch Deliveries to LFS data feed. It consists of a single task which is run on each active node by jobs named BRDBX003_PDEL_TO_LFS_1...4_NOPAGE.

3.20.1 Dependencies

Schedule BRDB_PDL_TO_LFS depends on the completion of schedule BRDB_START_LFS.

3.20.2 Job BRDBX003_PDEL_TO_LFS_1...4_NOPAGE

These jobs (one per node) start the feed that copies the Pouch Deliveries to LFS.

3.20.2.1 Implementation

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

3.20.2.2 Rerun Action

Rerun on failure

3.21 Schedule SOD

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

3.21.1 Dependencies

None.

3.21.2 Job COMPLETE

This job simply echoes a message before exiting.



3.21.2.1 Implementation

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

3.21.2.2 Rerun Action

None.

3.22 Schedule BRDB_REF_DATA_SLA

This schedule is run daily. It runs the BRDB utility to generate Reference Data SLAs. 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 BRDBX032_BRDB_REF_DATA_SLA is included here.

3.22.1 Dependencies

Schedule BRDB_REF_DATA_SLA depends on the completion of schedule BRDB_SOBI.

3.22.2 Job BRDBX032_BRDB_REF_DATA_SLA

This job runs the BRDB utility that generates Reference Data SLAs.

3.22.2.1 Implementation

This job is implemented by a call to the shell script BRDBX032.sh.

3.22.2.2 Rerun Action

Alert Operations on failure.

3.23 Schedule BRDB_ONCH_AGG

This schedule is run daily. It aggregates the overnight cash on hand (ONCH) figures. 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.

3.23.1 Dependencies

Schedule BRDB_ONCH_AGG depends on the completion of schedule BRDB_SOBI.

Job BRDBC008_CHECK_ONCH_AGG depends on jobs BRDBX007_ONCH_AGG_1...4.

3.23.2 Job BRDBX007_ONCH_AGG_1...4

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

3.23.2.1 Implementation

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



3.23.2.2 Rerun Action

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

3.23.3 Job BRDBC008_CHECK_ONCH_AGG

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

3.23.3.1 Implementation

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

3.23.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.24 Schedule BRDB_CSH_TO_LFS

This schedule is run daily. It runs the Cash Declarations to LFS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_CASH_TO_LFS_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_CASH_TO_LFS is included here.

3.24.1 Dependencies

Schedule BRDB_CSH_TO_LFS depends on the completion of schedule BRDB_ONCH_AGG.

Job BRDBC008_CHECK_CASH_TO_LFS depends on jobs BRDBX003_CASH_TO_LFS_1...4.

3.24.2 Job BRDBX003_CASH_TO_LFS_1...4

These jobs (one per node) run the feed that copies the Cash Declarations to LFS.

3.24.2.1 Implementation

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

3.24.2.2 Rerun Action

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

3.24.3 Job BRDBC008_CHECK_CASH_TO_LFS

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

3.24.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_CASH_TO_LFS.



3.24.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.25 Schedule BRDB_FROM_EMDB

This schedule is run daily at 19:30. It runs the Estate Management interface 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_BRDATA_FROM_EMDB is included here.

3.25.1 Dependencies

Schedule BRDB_FROM_EMDB depends on the completion of schedules BRDB_SOBI and EST_BRDB_UPD.

3.25.2 Job BRDBX003_BRDATA_FROM_EMDB

This job runs the Estate Management interface feed.

3.25.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_EMDB_INTERFACE.

3.25.2.2 Rerun Action

Alert Operations on failure??

3.26 Schedule BRDB_PAUSE_LFS

This schedule is run daily at 20:00. It stops the two LFS feed processes, to allow the LFS batch jobs to run overnight without activity occurring in the relevant tables. 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_LFS_PCOL_COPY

BRDBX011_PAUSE_LFS_PDEL_COPY

3.26.1 Dependencies

Schedule BRDB_PAUSE_LFS depends on the completion of schedule BRDB_SOBI.

3.26.2 Job BRDBX011_PAUSE_LFS_PCOL_COPY

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

3.26.2.1 Implementation

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



3.26.2.2 Rerun Action

Alert Operations on failure.

3.26.3 Job BRDBX011_PAUSE_LFS_PDEL_COPY

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

3.26.3.1 Implementation

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

3.26.3.2 Rerun Action

Alert Operations on failure.

3.27 Schedule BRDB_EPOS_TO_TPS

This schedule is run daily. It runs the EPOSS transactions to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_EPOSS_TO_TPS_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_EPOSS_TO_TPS is included here.

3.27.1 Dependencies

Schedule BRDB_EPOS_TO_TPS depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

Job BRDBC008_CHECK_EPOSS_TO_TPS depends on jobs BRDBX003_EPOSS_TO_TPS_1...4.

3.27.2 Job BRDBX003_EPOSS_TO_TPS_1...4

These jobs (one per node) run the feed that copies the EPOSS transactions to TPS.

3.27.2.1 Implementation

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

3.27.2.2 Rerun Action

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

3.27.3 Job BRDBC008_CHECK_EPOSS_TO_TPS

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

3.27.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_EPOSS_TXN_TO_TPS.



3.27.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.28 Schedule BRDB_APS_TO_TPS

This schedule is run daily. It runs the APS transactions to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_APS_TO_TPS_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_APS_TO_TPS is included here.

3.28.1 Dependencies

Schedule BRDB_APS_TO_TPS depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

Job BRDBC008_CHECK_APS_TO_TPS depends on jobs BRDBX003_APS_TO_TPS_1...4.

3.28.2 Job BRDBX003_APS_TO_TPS_1...4

These jobs (one per node) run the feed that copies the APS transactions to TPS.

3.28.2.1 Implementation

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

3.28.2.2 Rerun Action

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

3.28.3 Job BRDBC008_CHECK_APS_TO_TPS

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

3.28.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_APS_TXN_TO_TPS.

3.28.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.29 Schedule BRDB_NWB_TO_TPS

This schedule is run daily. It runs the NWB transactions to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_NWB_TO_TPS_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_NWB_TO_TPS is included here.



3.29.1 Dependencies

Schedule BRDB_NWB_TO_TPS depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

Job BRDBC008_CHECK_NWB_TO_TPS depends on jobs BRDBX003_NWB_TO_TPS_1...4.

3.29.2 Job BRDBX003_NWB_TO_TPS_1...4

These jobs (one per node) run the feed that copies the NWB transactions to TPS.

3.29.2.1 Implementation

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

3.29.2.2 Rerun Action

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

3.29.3 Job BRDBC008_CHECK_NWB_TO_TPS

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

3.29.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_NWB_TXN_TO_TPS.

3.29.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.30 Schedule BRDB_DCS_TO_TPS

This schedule is run daily. It runs the DCS transactions to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_DCS_TO_TPS_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_DCS_TO_TPS is included here.

3.30.1 Dependencies

Schedule BRDB_DCS_TO_TPS depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

Job BRDBC008_CHECK_DCS_TO_TPS depends on jobs BRDBX003_DCS_TO_TPS_1...4.

3.30.2 Job BRDBX003_DCS_TO_TPS_1...4

These jobs (one per node) run the feed that copies the DCS transactions to TPS.



3.30.2.1 Implementation

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

3.30.2.2 Rerun Action

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

3.30.3 Job BRDBC008_CHECK_DCS_TO_TPS

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

3.30.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_DCS_TXN_TO_TPS.

3.30.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.31 Schedule BRDB_BDC_TO_TPS

This schedule is run daily. It runs the BDC transactions to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_BUREAU_TO_TPS_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_BUREAU_TO_TPS is included here.

3.31.1 Dependencies

Schedule BRDB_BDC_TO_TPS depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

Job BRDBC008_CHECK_BUREAU_TO_TPS depends on jobs BRDBX003_BUREAU_TO_TPS_1...4.

3.31.2 Job BRDBX003_BUREAU_TO_TPS_1...4

These jobs (one per node) run the feed that copies the BDC transactions to TPS.

3.31.2.1 Implementation

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

3.31.2.2 Rerun Action

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



3.31.3 Job BRDBC008_CHECK_BUREAU_TO_TPS

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

3.31.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_BDC_TXN_TO_TPS.

3.31.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.32 Schedule BRDB_EVT_TO_TPS

This schedule is run daily. It runs the EPOSS events to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_EVENTS_TO_TPS_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_EVENTS_TO_TPS is included here.

3.32.1 Dependencies

Schedule BRDB_EVT_TO_TPS depends on the completion of schedules BRDB_SOB and TPS_BULK_HARV.

Job BRDBC008_CHECK_EVENTS_TO_TPS depends on jobs BRDBX003_EVENTS_TO_TPS_1...4.

3.32.2 Job BRDBX003_EVENTS_TO_TPS_1...4

These jobs (one per node) run the feed that copies the EPOSS events to TPS.

3.32.2.1 Implementation

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

3.32.2.2 Rerun Action

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

3.32.3 Job BRDBC008_CHECK_EVENTS_TO_TPS

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

3.32.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_EPOSS_EVNT_TO_TPS.



3.32.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.33 Schedule BRDB_COFS_TO_TPS

This schedule is run daily. It runs the Cut Off Summaries to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_COFF_SUMM_TO_TPS_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_COFF_SUMM_TO_TPS is included here.

3.33.1 Dependencies

Schedule BRDB_COFS_TO_TPS depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

Job BRDBC008_CHECK_COFF_SUMM_TO_TPS depends on jobs BRDBX003_COFF_SUMM_TO_TPS_1...4.

3.33.2 Job BRDBX003_COFF_SUMM_TO_TPS_1...4

These jobs (one per node) run the feed that copies the Cut Off Summaries to TPS.

3.33.2.1 Implementation

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

3.33.2.2 Rerun Action

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

3.33.3 Job BRDBC008_CHECK_COFF_SUMM_TO_TPS

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 feed name BRDB_CUTOFF_SUMM_TO_TPS.

3.33.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.34 Schedule BRDB_TC_FROM_TPS

This schedule is run daily. It runs the Transaction Corrections from TPS interface 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_TC_FROM_EMDB is included here.



3.34.1 Dependencies

Schedule BRDB_TC_FROM_TPS depends on the completion of schedules BRDB_SOB and TPS_TC.

3.34.2 Job BRDBX003_TC_FROM_EMDB

This job runs the Transaction Corrections from TPS interface feed.

3.34.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_TXN_CORR_FROM_TPS.

3.34.2.2 Rerun Action

Alert Operations on failure.

3.35 Schedule BRDB_TPS_COMPL

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

3.35.1 Dependencies

Schedule BRDB_TPS_COMPL depends on the completion of schedules BRDB_EPOS_TO_TPS, BRDBAPS_TO_TPS, BRDB_NWB_TO_TPS, BRDB_DCS_TO_TPS, BRDB_BDC_TO_TPS, BRDB_EVT_TO_TPS and BRDB_COFS_TO_TPS.

3.35.2 Job COMPLETE

This job simply echoes a message before exiting.

3.35.2.1 Implementation

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

3.35.2.2 Rerun Action

None.

3.36 Schedule BRDB_TPS_TOTALS

This schedule is run daily. It aggregates the outlet transaction totals. 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_TPS_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_TPS_TXN_TOTALS is included here.

3.36.1 Dependencies

Schedule BRDB_TPS_TOTALS depends on the completion of schedule BRDB_SOB.

Job BRDBC008_CHECK_TPS_TXN_TOTALS depends on jobs BRDBX007_TPS_TXN_TOTALS_1...4.



3.36.2 Job BRDBX007_TPS_TXN_TOTALS_1...4

These jobs (one per node) perform the aggregation of the outlet transaction totals.

3.36.2.1 Implementation

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

3.36.2.2 Rerun Action

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

3.36.3 Job BRDBC008_CHECK_TPS_TXN_TOTALS

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

3.36.3.1 Implementation

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

3.36.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.37 Schedule BRDB_TOTL_TO_TPS

This schedule is run daily. It runs the Transactions Totals to TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_TXN_TOTALS_TO_TPS_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_TXN_TOTALS_TO_TPS is included here.

3.37.1 Dependencies

Schedule BRDB_TOTL_TO_TPS depends on the completion of schedule BRDB_TPS_TOTALS.

Job BRDBC008_CHECK_TXN_TOTALS_TO_TPS depends on jobs
BRDBX003_TXN_TOTALS_TO_TPS_1...4.

3.37.2 Job BRDBX003_TXN_TOTALS_TO_TPS_1...4

These jobs (one per node) run the feed that copies the Transactions Totals to TPS.

3.37.2.1 Implementation

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



3.37.2.2 Rerun Action

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

3.37.3 Job BRDBC008_CHECK_TXN_TOTALS_TO_TPS

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

3.37.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_TXN_TOT_TO_TPS.

3.37.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.38 Schedule BRDB_APS_TOTALS

This schedule is run daily. It aggregates the APS transaction totals. 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_APS_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_APS_TXN_TOTALS is included here.

3.38.1 Dependencies

Schedule BRDB_APS_TOTALS depends on the completion of schedule BRDB_SOBI.

Job BRDBC008_CHECK_APS_TXN_TOTALS depends on jobs BRDBX007_APS_TXN_TOTALS_1...4.

3.38.2 Job BRDBX007_APS_TXN_TOTALS_1...4

These jobs (one per node) perform the aggregation of the APS transaction totals.

3.38.2.1 Implementation

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

3.38.2.2 Rerun Action

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

3.38.3 Job BRDBC008_CHECK_APS_TXN_TOTALS

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

3.38.3.1 Implementation

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



3.38.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.39 Schedule BRDB_TOTL_TO_APS

This schedule is run daily. It runs the Transactions Totals to APS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_TXN_TOTALS_TO_APS_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_TXN_TOTALS_TO_APS is included here.

3.39.1 Dependencies

Schedule BRDB_TOTL_TO_APS depends on the completion of schedule BRDB_APS_TOTALS.

Job BRDBC008_CHECK_TXN_TOTALS_TO_APS depends on jobs BRDBX003_TXN_TOTALS_TO_APS_1...4.

3.39.2 Job BRDBX003_TXN_TOTALS_TO_APS_1...4

These jobs (one per node) run the feed that copies the Transactions Totals to APS.

3.39.2.1 Implementation

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

3.39.2.2 Rerun Action

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

3.39.3 Job BRDBC008_CHECK_TXN_TOTALS_TO_APS

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 feed name BRDB_TXN_TOT_TO_APS.

3.39.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.40 Schedule BRDB_TXNS_TO_APS

This schedule is run daily. It runs the APS transactions to APS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_TXNS_TO_APS_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_TXNS_TO_APS is included here.



3.40.1 Dependencies

Schedule BRDB_TXNS_TO_APSP depends on the completion of schedules BRDB_SOBS and APS_BULK_HARV.

Job BRDBC008_CHECK_TXNS_TO_APSP depends on jobs BRDBX003_TXNS_TO_APSP_1...4.

3.40.2 Job BRDBX003_TXNS_TO_APSP_1...4

These jobs (one per node) run the feed that copies the APSP transactions to TPS.

3.40.2.1 Implementation

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

3.40.2.2 Rerun Action

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

3.40.3 Job BRDBC008_CHECK_TXNS_TO_APSP

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

3.40.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_APSP_TXN_TO_APSP.

3.40.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.41 Schedule BRDB_APSP_COMPL

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

3.41.1 Dependencies

Schedule BRDB_APSP_COMPL depends on the completion of schedules BRDB_TXNS_TO_APSP and BRDB_TOTL_TO_APSP.

3.41.2 Job COMPLETE

This job simply echoes a message before exiting.

3.41.2.1 Implementation

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



3.41.2.2 Rerun Action

None.

3.42 Schedule BRDB_NWB_TO_DRS

This schedule is run daily. It runs the NWB transactions to DRS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_NWB_TO_DRS_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_NWB_TO_DRS is included here.

3.42.1 Dependencies

Schedule BRDB_NWB_TO_DRS depends on the completion of schedule BRDB_SOB.

Job BRDBC008_CHECK_NWB_TO_DRS depends on jobs BRDBX003_NWB_TO_DRS_1...4.

3.42.2 Job BRDBX003_NWB_TO_DRS_1...4

These jobs (one per node) run the feed that copies the NWB transactions to DRS.

3.42.2.1 Implementation

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

3.42.2.2 Rerun Action

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

3.42.3 Job BRDBC008_CHECK_NWB_TO_DRS

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

3.42.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_NWB_TXN_TO_DRS.

3.42.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.43 Schedule BRDB_DCS_TO_DRS

This schedule is run daily. It runs the DCS transactions to DRS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_DCS_TO_DRS_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_DCS_TO_DRS is included here.



3.43.1 Dependencies

Schedule BRDB_DCS_TO_DRS depends on the completion of schedule BRDB_SO.

Job BRDBC008_CHECK_DCS_TO_DRS depends on jobs BRDBX003_DCS_TO_DRS_1...4.

3.43.2 Job BRDBX003_DCS_TO_DRS_1...4

These jobs (one per node) run the feed that copies the DCS transactions to DRS.

3.43.2.1 Implementation

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

3.43.2.2 Rerun Action

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

3.43.3 Job BRDBC008_CHECK_DCS_TO_DRS

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

3.43.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_DCS_TXN_TO_DRS.

3.43.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.44 Schedule BRDB_DRS_COMPL

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

3.44.1 Dependencies

Schedule BRDB_DRS_COMPL depends on the completion of schedules BRDB_NWB_TO_DRS and BRDB_DCS_TO_DRS.

3.44.2 Job COMPLETE

This job simply echoes a message before exiting.

3.44.1.1 Implementation

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

3.44.1.2 Rerun Action

None.



3.45 Schedule BRDB_HYD_TP_EPOS

This Hydra schedule is run daily. It runs the EPOSS transactions from TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_EPOSS_FROM_TPS_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_EPOSS_FROM_TPS is included here.

3.45.1 Dependencies

Schedule BRDB_HYD_TP_EPOS depends on the completion of schedule BRDB_EPOS_TO_TPS.

Job BRDBC008_CHECK_EPOSS_FROM_TPS depends on jobs
BRDBX003_EPOSS_FROM_TPS_1...4.

3.45.2 Job BRDBX003_EPOSS_FROM_TPS_1...4

These jobs (one per node) run the feed that copies the EPOSS transactions from TPS.

3.45.2.1 Implementation

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

3.45.2.2 Rerun Action

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

3.45.3 Job BRDBC008_CHECK_EPOSS_FROM_TPS

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

3.45.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_EPOSS_TXN_FROM_TPS.

3.45.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.46 Schedule BRDB_HYD_TP_AP

This Hydra schedule is run daily. It runs the APS transactions from TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_AP_FROM_TPS_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_AP_FROM_TPS is included here.

3.46.1 Dependencies

Schedule BRDB_HYD_TP_AP depends on the completion of schedule BRDB_APS_TO_TPS.



Job BRDBC008_CHECK_AP_FROM_TPS depends on jobs BRDBX003_AP_FROM_TPS_1...4.

3.46.2 Job BRDBX003_AP_FROM_TPS_1...4

These jobs (one per node) run the feed that copies the APS transactions from TPS.

3.46.2.1 Implementation

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

3.46.2.2 Rerun Action

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

3.46.3 Job BRDBC008_CHECK_AP_FROM_TPS

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

3.46.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_APX_TXN_FROM_TPS.

3.46.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.47 Schedule BRDB_HYD_TP_BDC

This Hydra schedule is run daily. It runs the BDC transactions from TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_BUREAU_FROM_TPS_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_BUREAU_FROM_TPS is included here.

3.47.1 Dependencies

Schedule BRDB_HYD_TP_BDC depends on the completion of schedule BRDB_BDC_TO_TPS.

Job BRDBC008_CHECK_BUREAU_FROM_TPS depends on jobs BRDBX003_BUREAU_FROM_TPS_1...4.

3.47.2 Job BRDBX003_BUREAU_FROM_TPS_1...4

These jobs (one per node) run the feed that copies the Bureau transactions from TPS.

3.47.2.1 Implementation

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



3.47.2.2 Rerun Action

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

3.47.3 Job BRDBC008_CHECK_BUREAU_FROM_TPS

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

3.47.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_BDC_TXN_FROM_TPS.

3.47.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.48 Schedule BRDB_HYD_TP_NWB

This Hydra schedule is run daily. It runs the NWB transactions from TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_NWB_FROM_TPS_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_NWB_FROM_TPS is included here.

3.48.1 Dependencies

Schedule BRDB_HYD_TP_NWB depends on the completion of schedule BRDB_NWB_TO_TPS.

Job BRDBC008_CHECK_NWB_FROM_TPS depends on jobs BRDBX003_NWB_FROM_TPS_1...4.

3.48.2 Job BRDBX003_NWB_FROM_TPS_1...4

These jobs (one per node) run the feed that copies the NWB transactions from TPS.

3.48.2.1 Implementation

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

3.48.2.2 Rerun Action

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

3.48.3 Job BRDBC008_CHECK_NWB_FROM_TPS

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

3.48.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_NWB_TXN_FROM_TPS.



3.48.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.49 Schedule BRDB_HYD_TP_DCS

This Hydra schedule is run daily. It runs the DCS transactions from TPS feed. It performs two tasks, firstly running the feed itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX003_DCS_FROM_TPS_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_DCS_FROM_TPS is included here.

3.49.1 Dependencies

Schedule BRDB_HYD_TP_DCS depends on the completion of schedule BRDB_DCS_TO_TPS.

Job BRDBC008_CHECK_DCS_FROM_TPS depends on jobs BRDBX003_DCS_FROM_TPS_1...4.

3.49.2 Job BRDBX003_DCS_FROM_TPS_1...4

These jobs (one per node) run the feed that copies the DCS transactions from TPS.

3.49.2.1 Implementation

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

3.49.2.2 Rerun Action

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

3.49.3 Job BRDBC008_CHECK_DCS_FROM_TPS

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

3.49.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant feed name BRDB_DCS_TXN_FROM_TPS.

3.49.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.50 Schedule BRDB_HYD_ONGOING

This Hydra schedule is run daily. It runs the Reconciliation Blob from TPS interface 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_RECON_FROM_TPS is included here.



3.50.1 Dependencies

Schedule BRDB_HYD_ONGOING depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

3.50.2 Job BRDBX003_RECON_FROM_TPS

This job runs the Reconciliation Blob from TPS interface feed.

3.50.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_RECON_XML_FROM_TPS.

3.50.2.2 Rerun Action

Alert Operations on failure.

3.51 Schedule BRDB_HYD_INDAY

This Hydra schedule is run daily. It runs the In-Day Migration Blob from TPS interface 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_INDAY_FROM_TPS is included here.

3.51.1 Dependencies

Schedule BRDB_HYD_INDAY depends on the completion of schedules BRDB_SOBI and TPS_BULK_HARV.

3.51.2 Job BRDBX003_INDAY_FROM_TPS

This job runs the In-Day Migration Blob from TPS interface feed.

3.51.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_INDAY_XML_FROM_TPS.

3.51.2.2 Rerun Action

Alert Operations on failure.

3.52 Schedule BRDB_HYD_PREPROC

This Hydra schedule is run daily. It runs the BRDB Hydra Preparation. It performs two tasks, firstly running the preparation process itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX033_BRDB_HYDRA_PROCESSING_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_BRDBX033_PREP_RECON is included here.



3.52.1 Dependencies

Schedule BRDB_HYD_PREPROC depends on the completion of schedule BRDB_HYD_ONGOING.

Job BRDBC008_BRDBX033_PREP_RECON depends on jobs
BRDBX033_BRDB_HYDRA_PROCESSING_1...4.

3.52.2 Job BRDBX033_BRDB_HYDRA_PROCESSING_1...4

These jobs (one per node) perform the preparation of Horizon outlets for HNGX.

3.52.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX033.sh.

3.52.2.2 Rerun Action

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

3.52.3 Job BRDBC008_BRDBX033_PREP_RECON

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

3.52.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDBX033_PREP_RECON.

3.52.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.53 Schedule BRDB_XFR_COMPL

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

3.53.1 Dependencies

Schedule BRDB_XFR_COMPL depends on the completion of schedules BRDB_HYD_TP_EPOS (H), BRDB_HYD_TP_AP (H), BRDB_HYD_TP_NWB (H), BRDB_HYD_TP_DCS (H), BRDB_HYD_TP_BDC (H), BRDB_EVT_TO_TPS, BRDB_TOTL_TO_TPS, BRDB_TXNS_TOAPS and BRDB_DRS_COMPL.

3.53.2 Job COMPLETE

This job simply echoes a message before exiting.

3.53.2.1 Implementation

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



3.53.2.2 Rerun Action

None.

3.54 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.

3.54.1 Dependencies

Schedule BRDB_FEED_ERRORS depends on the completion of schedule BRDB_XFR_COMPL.

3.54.2 Job BRDBX007_RAISE_FEED_DATA_EXCEPTIONS

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

3.54.2.1 Implementation

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

3.54.2.2 Rerun Action

Alert Operations on failure.

3.55 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_AGG is included here.

3.55.1 Dependencies

Schedule BRDB_NCU_TXN_AGG depends on the completion of schedules BRDB_HYD_TP_EPOS (H), BRDB_HYD_TP_AP (H), BRDB_HYD_TP_BDC (H), BRDB_HYD_TP_NWB (H) and BRDB_HYD_TP_DCS (H).

Job BRDBC008_CHECK_NON_CUMU_TXN_AGG depends on jobs BRDBX007_NON_CUMU_TXN_TOTALS_1...4.

3.55.2 Job BRDBX007_NON_CUMU_TXN_TOTALS_1...4

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

3.55.2.1 Implementation

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



3.55.2.2 Rerun Action

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

3.55.3 Job BRDBC008_CHECK_NON_CUMU_TXN_AGGR

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

3.55.3.1 Implementation

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

3.55.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

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

3.56.1 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.

3.56.2 Job BRDBX007_CUMU_TXN_AGGR_1...4

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

3.56.2.1 Implementation

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

3.56.2.2 Rerun Action

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

3.56.3 Job BRDBC008_CHECK_CUMU_TXN_AGGR

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

3.56.3.1 Implementation

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



3.56.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.57 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_JSON_USN_SSN is included here.

3.57.1 Dependencies

Schedule BRDB_BBNI_MAINT depends on the completion of schedule BRDB_CU_TXN_AGG.

3.57.2 Job BRDBX031_JSON_USN_SSN

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

3.57.2.1 Implementation

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

3.57.2.2 Rerun Action

*** Prompts for rerun – action? **

3.58 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.

3.58.1 Dependencies

Schedule BRDB_SUMMARY_DTE depends on the completion of schedule BRDB_BBNI_MAINT.

3.58.2 Job BRDBX011_SET_DAILY_SUMMARY_DATE

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

3.58.1.1 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.

3.58.1.2 Rerun Action

Alert Operations on failure.



3.59 Schedule BRDB_HYDRA_PROC

This Hydra schedule is run daily. It performs the Hydra XML processing. It performs three tasks, firstly performing the processing itself on all active nodes, with automatic waiting and rerunning; see section 3.1 above for details. Only the main jobs BRDBX030_BRDB_HYDRA_PROCESSING_1...4 are included here. The second and third tasks checks for completion of the two parts of the previous task (reconciliation and in-day), and can be run on any active node; see section 3.2 above for details. Only the parent jobs BRDBC008_BRDBX030_RECON and BRDBC008_BRDBX030_INDAY are included here.

3.59.1 Dependencies

Schedule BRDB_HYDRA_PROC depends on the completion of schedules BRDB_SUMMARY_DTE, BRDB_HYD_INDAY (H) and BRDB_HYD_PREPROC.

Job	BRDBC008_BRDBX030_RECON	depends	on	jobs
	BRDBX030_BRDB_HYDRA_PROCESSING_1...4			

Job BRDBC008_BRDBX030_INDAY depends on job BRDBC008_BRDBX030_RECON.

3.59.2 Job BRDBX030_BRDB_HYDRA_PROCESSING_1...4

These jobs (one per node) perform Hydra processing of in-day and reconciliation data.

3.59.2.1 Implementation

These jobs are implemented by a call to the shell script BRDBX030.sh.

3.59.2.2 Rerun Action

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

3.59.3 Job BRDBC008_BRDBX030_RECON

This job checks for the successful completion of the reconciliation part of the processing job for all FAD-Hashes.

3.59.3.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDBX030_RECON.

3.59.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.59.4 Job BRDBC008_BRDBX030_INDAY

This job checks for the successful completion of the in-day part of the processing job for all FAD-Hashes.



3.59.4.1 Implementation

This job is implemented by a call to the executable BRDBC008 specifying the relevant process name BRDBX030_INDAY.

3.59.4.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.60 Schedule BRDB_GEN REP

This schedule is run daily. It generates the reconciliation reports. 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:

GENERIC_CREATE_REPORT_VIEWS
GENERIC_CREATE_RECON_REPORTS

3.60.1 Dependencies

Schedule BRDB_GEN REP depends on the completion of schedules BRDB_HYDRA_PROC and BRDB_REF_DATA_SLA.

Job GENERIC_CREATE_RECON_REPORTS depends on job GENERIC_CREATE_REPORT_VIEWS.

3.60.2 Job GENERIC_CREATE_REPORT_VIEWS

This job creates the generic views for reconciliation reporting.

3.60.2.1 Implementation

This job is implemented by a call to the shell script GREPX001.sh.

3.60.2.2 Rerun Action

*** Prompts for rerun – action? **

3.60.3 Job GENERIC_CREATE_RECON_REPORTS

This job creates the generic reconciliation reports.

3.60.1.1 Implementation

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

3.60.1.2 Rerun Action

*** Prompts for rerun – action? **



3.61 Schedule BRDB_TO_DWH

This schedule is run daily. It performs the file transfer for the BRDB Branch Migration Status 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 BRDBX020_BRDB_XFER_TO_DWH is included here.

3.61.1 Dependencies

Schedule BRDB_TO_DWH depends on the completion of schedule BRDB_GEN REP.

3.61.2 Job BRDBX020_BRDB_XFER_TO_DWH

This job performs the file transfers for the BRDB Branch Migration Status and Reference data feeds.

3.61.2.1 Implementation

This job is implemented by a call to the shell script BRDBX020.sh.

3.61.2.2 Rerun Action

Alert Operations on failure. This job may be re-runnable, depending on the error (see failures below for deciding if re-runnable or not).

3.61.2.1.1 Failures

"Source file <n> <filename> does not exist"

Ensure Job GENERIC_CREATE_RECON_REPORTS completed successfully and if all expected reports are present in \${BRDB_MSU_OUTPUT}

Expected reports are:

- DW_Branch_Migration_Extract.csv
- DW_Reference_Data_SLA.csv

Once the cause of the 'missing' reports is resolved, ensure the following files are removed (if present)

- \${REPOSITORY}/brdb/YYMMDD/YYMMDD00.bac
- \${REPOSITORY}/brdb/YYMMDD/YYMMDD00.bms

BRDBX020_BRDB_XFER_TO_DWH may then be rerun.

"Destination file <n> <filename> already exists"

The above error suggests that the script has already been run successfully. Alert Operations as this will require more investigation into why the script has failed.

3.62 Schedule BRDB_AGG_COMPL

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



3.62.1 Dependencies

Schedule BRDB_AGG_COMPL depends on the completion of schedules BRDB_SUMMARY_DTE and BRDB_TO_DWH.

3.62.2 Job COMPLETE

This job simply echoes a message before exiting.

3.62.2.1 Implementation

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

3.62.2.2 Rerun Action

None.

3.63 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.

3.63.1 Dependencies

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

3.63.2 Job BRDBX003_REFDATA_FROM_RDDS

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

3.63.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_HOST_REF_FROM_RDDS.

3.63.2.2 Rerun Action

*** Prompts for rerun – action? **

3.64 Schedule BRDB_FROM_TPS

This schedule is run daily at 00:10. It runs the Outlets/Transaction Modes data from TPS 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_TPS is included here.

3.64.1 Dependencies

Schedule BRDB_FROM_TPS depends on the completion of schedules BRDB_SOBI and TPSEOD.TPSC207.



3.64.2 Job BRDBX003_REFDATA_FROM_TPS

This job runs the Outlets/Transaction Modes data from TPS data feed.

3.64.2.1 Implementation

This job is implemented by a call to the shell script BRDBX003.sh specifying the relevant feed name BRDB_REF_COPY_FROM_TPS.

3.64.2.2 Rerun Action

*** Prompts for rerun – action? **

3.65 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 *** (or 03:00???) **. This is implemented within the BRDB_MONITOR schedule – see section 3.77.

3.65.1 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.

3.65.2 Job BRDBC002_AUDIT_1...4

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

3.65.2.1 Implementation

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

3.65.2.2 Rerun Action

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

3.65.3 Job BRDBC008_CHECK_AUDIT_FEED

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

3.65.3.1 Implementation

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



3.65.3.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.65.4 Job BRDBC033_AUDIT

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

3.65.4.1 Implementation

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

3.65.4.2 Rerun Action

As specified in section 3.2, alert Operations if rerun fails.

3.66 Schedule BRDB_ORA_STATS

This schedule is run daily. It gathers the Oracle optimiser statistics. 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 BRDBX005_SCHEMA is included here.

3.66.1 Dependencies

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

3.66.2 Job BRDBX005_SCHEMA

This job gathers the Oracle optimiser statistics.

3.66.2.1 Implementation

This job is implemented by a call to the shell script BRDBX005.sh specifying the parameter SCHEMA.

3.66.2.2 Rerun Action

*** Prompts for rerun – action? **

3.67 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.

3.67.1 Dependencies

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



3.67.2 Job BRDBC004

This job runs the Audit, Archive and Purge process.

3.67.2.1 Implementation

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

3.67.2.2 Rerun Action

*** Prompts for rerun – action? **

3.67.3 Job BRDBX006

This job runs the BRDB File Housekeeping process.

3.67.3.1 Implementation

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

3.67.3.2 Rerun Action

*** Prompts for rerun – action? **

3.67.4 Job BRDB_HKP_ORAFILES1

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

3.67.4.1 Implementation

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

3.67.4.2 Rerun Action

*** Prompts for rerun – action? **

3.67.5 Job BRDB_HKP_ORAFILES2

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

3.67.1.1 Implementation

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

3.67.1.2 Rerun Action

*** Prompts for rerun – action? **



3.68 Schedule BRDB_PAUSE_FEED2

This schedule is run daily. It stops the two NPS copy processes prior to end of day processing. 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

3.68.1 Dependencies

Schedule BRDB_PAUSE_FEED2 depends on the completion of schedules BRDB_ADMIN and BRDB_CSH_TO_LFS.

3.68.2 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).

3.68.2.1 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".

3.68.2.2 Rerun Action

Alert Operations on failure.

3.68.3 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).

3.68.3.1 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".

3.68.3.2 Rerun Action

Alert Operations on failure.

3.69 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.



3.69.1 Dependencies

Schedule BRDB_EOD depends on the completion of schedule BRDB_PAUSE_FEED2.

3.69.2 Job BRDBC009

This job runs the BRDB end of day utility.

3.69.2.1 Implementation

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

3.69.2.2 Rerun Action

*** Prompts for rerun – action? **

3.70 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

3.70.1 Dependencies

Schedule BRDB_START_FEED2 depends on the completion of schedule BRDB_EOD.

3.70.2 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).

3.70.2.1 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".

3.70.2.2 Rerun Action

Alert Operations on failure.

3.70.3 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).

3.70.3.1 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".



3.70.3.2 Rerun Action

Rerun on failure.

3.71 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.

3.71.1 Dependencies

Schedule BRDB_TT_TO_NPS2 depends on the completion of schedule BRDB_START_FEED2.

3.71.2 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.

3.71.2.1 Implementation

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

3.71.2.2 Rerun Action

Rerun on failure.

3.72 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.

3.72.1 Dependencies

Schedule BRDB_GREV_NPS2 depends on the completion of schedule BRDB_START_FEED2.

3.72.2 Job BRDBX003_GREV_TO_NPS_1...4_NOPAGE

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

3.72.1.1 Implementation

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

3.72.1.2 Rerun Action

Rerun on failure.



3.73 Schedule BRDB_START_BKP

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

3.73.1 Dependencies

Schedule BRDB_START_BKP depends on the completion of schedule BRDB_EOD.

3.73.2 Job COMPLETE

This job simply echoes a message before exiting.

3.73.2.1 Implementation

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

3.73.2.2 Rerun Action

None.

3.74 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.

3.74.1 Dependencies

Schedule BRDB_BACKUP_0 depends on the completion of schedule BRDB_START_BKP.

3.74.2 Job BRDB_LVL0_BACKUP

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

3.74.2.1 Implementation

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

3.74.2.2 Rerun Action

*** Prompts for rerun – action? **

3.75 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.



3.75.1 Dependencies

Schedule BRDB_BACKUP_1 depends on the completion of schedule BRDB_START_BKP.

3.75.2 Job BRDB_LVL1_BACKUP

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

3.75.2.1 Implementation

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

3.75.2.2 Rerun Action

*** Prompts for rerun – action? **

3.76 Schedule BRDB_BKP_COMPLETE

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.

3.76.1 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.

3.76.2 Job CREATE_BRDB_COMPLETE_FLAG

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

3.76.2.1 Implementation

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

3.76.2.2 Rerun Action

*** Prompts for rerun – action? **

3.77 Schedule BRDB_MONITOR

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

3.77.1 Dependencies

None

3.77.2 Job BRDB_MON_STARTUP

This checks that the BRDB_STARTUP job has completed by the required time of 06:00.



3.77.2.1 Implementation

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

3.77.2.2 Rerun Action

None.

3.77.3 Job BRDB_MON_PAUSE_FEED1

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

3.77.3.1 Implementation

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

3.77.3.2 Rerun Action

None.

3.77.4 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??) **.

3.77.4.1 Implementation

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

3.77.4.2 Rerun Action

None.

3.77.5 Job BRDB_MON_EOD

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

3.77.5.1 Implementation

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

3.77.5.2 Rerun Action

None.



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).

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 backup (incremental level 1 or 2).
- 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 & Streams

RMAN, by default, is configured to remove any archivelogs after a successful backup. Streams 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 Streams Capture process.

If Streams 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 Streams no longer need the archivelog, it will be released for deletion by RMAN.

4.2 Restoring files with RMAN

DBA's 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 importance!

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!

4.3 Failure and Recovery

All failure detections and subsequent recovery actions are in all circumstances managed and co-ordinated by the SMC. The SSC must also in all cases be included in failure and recovery decisions and communication.



If at any point there is doubt in any form about anything related database recovery, the DBA and UNIX Support Teams must be contacted for clarification.

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 Logically Corrupt Redolog Group

The solution to this scenario may be a moot point as the quickest solution to this problem may be to failover the Primary database to Standby.

4.3.2.1.1 Scenario and Recovery Solution

Scenario: This failure scenario involves having all redo logs of a particular redo log group, logically corrupt. Each redo log group has multiple numbers of files each, mirrored and multiplexed, however they are all corrupt. A logfile group will belong to a single instance of the database, e.g. BRDB2.

Solution: The solution to this recovery scenario is to recreate the affected instance by removing and re-creating the affected redologs. These will only apply to a single logfile group.

There is also the ability to restore the database, using Oracle RMAN, to a known point-in-time before the corruption and recover the database. The archivelogs will have multiple copies in multiple destinations. All possible copies of these logs will need to be available. However, this scenario is likely overkill as the whole database has not been affected, just a single instance has. This solution would be a different story if all redolog groups failed.

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.

There is a **very notable caveat** to this if all 4 nodes go down, and that is that OCFS2 must be checked and verified as started (on each node). If not, the clusters will not be able to correctly communicate with each other and it has been noted (during periods of testing) that cluster timeouts can occur in this scenario and cause further cluster failures. Verifying the status of OCFS2 is critical.

To check that OCFS2 is started and has the correct configuration information, perform the following check on all affected nodes as the **root** user (the output should show "configured" matching "active"):-

```
$> /etc/init.d/ocfs2 status
Configured OCFS2 mountpoints: /u02/oradata /u03/oradata /u04/oradata
Active OCFS2 mountpoints: /u02/oradata /u03/oradata /u04/oradata
```

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 as much a logical part of the application DNA as they are literal entities of an Oracle RAC database. 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 is not aware of that.

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 our recommended course of action** as the RAC cluster has been designed to and in addition has proven it's ability to handle the extra load with 1 node removed from it, i.e. a 3 node cluster.

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 Andrew Aylward for fast recovery of instance due to unexpected node failure. Authorisation given by Graham Allen.

HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE<Host Name>: 11tpbdb001 (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 <i>brdb</i> 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 <i>instance_name</i> , <i>host_name</i> and <i>status</i> .	<pre>\$> . oraenv [now type in BRDB1 (assuming you're on node 1)]</pre> <pre>\$> sqlplus / SQL> SELECT * FROM v\$instance;</pre>
2.	Execute this DML to re-instate the availability of the instance in question. Commit your change	<pre>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)); COMMIT;</pre>

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 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 5.6) 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 5.6.



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 ***LIS*TENER_<node name>**, e.g. ***LIS*TENER_LPRPBDB001**
- The port the listener has been configured to use is **1529**
- The node (and in turn the IP) the listener has been configured to accept connections for is ***lprp<type>00[1234]j-vip***, e.g. for BDB node 1 the node name is ***lprpbdb001-vip***

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** and the **listener.ora** (found in **\$ORACLE_HOME/network/admin**)

5.1.1.1 Oracle Net Config. Files

The files have been formerly delivered by Tivoli Provisioning Manager and won't be needed to be changed unless there is a specific problem. In all cases the golden source of these files will be TPM. The following, however, shows a few excerpts of what the files should look like (October 2009): -



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



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.

```
aaylw01@lprpbdb001:~  

# sqlnet.ora.1prpbdb001 Network Configuration File: /u01/app/oracle/  

# Generated by Oracle configuration tools.  

#  

# NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)  

#  

# +-----+  

# | Network Configuration File: sqlnet.ora |  

# | Generated by HNGX configuration tool: - |  

# | - brdb_editsqlnet.sh (using 'tag' values) |  

# | |  

# +-----+  

#  

SQLNET.INBOUND_CONNECT_TIMEOUT=15  

SQLNET.EXPIRE_TIME=5  

~
```

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_LPRPBDB004 =  

(DESCRIPTION =  

  (ADDRESS = (PROTOCOL = TCP) (HOST = lprpbdb004-vip) (PORT = 1529))  

)  

BRDB4 =  

(DESCRIPTION =  

  (ADDRESS = (PROTOCOL = TCP) (HOST = pbdb004-vip) (PORT = 1529))  

  (CONNECT_DATA =  

    (SERVER = DEDICATED)  

    (SERVICE_NAME = BRDB)  

    (INSTANCE_NAME = BRDB4 )  

  )  

)  

BRDB =  

(DESCRIPTION =  

  (ADDRESS = (PROTOCOL = TCP) (HOST=pbdb001-vip) (PORT=1529))  

  (ADDRESS = (PROTOCOL = TCP) (HOST=pbdb002-vip) (PORT=1529))  

  (ADDRESS = (PROTOCOL = TCP) (HOST=pbdb003-vip) (PORT=1529))  

  (ADDRESS = (PROTOCOL = TCP) (HOST=pbdb004-vip) (PORT=1529))  

  (CONNECT_DATA =  

    (SERVER = DEDICATED)  

    (SERVICE_NAME = BRDB)
```



HOST BRANCH DATABASE SUPPORT GUIDE

COMMERCIAL IN CONFIDENCE



listener.ora

`INBOUND_CONNECT_TIMEOUT_LISTENER_LPRPBDB001 = 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_LPRPBDB001 = ON enforces the administration of the listener to an authorised user only, i.e. **oracle**

```
# | Generated by HNGX configuration tool: - |  
# | - brdb_edit_listener.sh (using 'tag' values) |  
# | |  
# +-----+  
  
LISTENER_LPRPBDB001 =  
(DESCRIPTION_LIST =  
(DESCRIPTION =  
    (ADDRESS = (PROTOCOL = TCP) (HOST = lprpbdb001-vip) (PORT = 1529) (IP = FIRST))  
    (ADDRESS = (PROTOCOL = TCP) (HOST = lprpbdb001) (PORT = 1529) (IP = FIRST))  
    #(ADDRESS = (PROTOCOL = IPC) (KEY = extproc))  
)  
)  
  
STARTUP_WAIT_TIME_LISTENER_LPRPBDB001 = 0  
INBOUND_CONNECT_TIMEOUT_LISTENER_LPRPBDB001 = 10  
ADMIN_RESTRICTIONS_LISTENER_LPRPBDB001 = ON  
  
SID_LIST_LISTENER_LPRPBDB001 =  
(SID_LIST =  
(SID_DESC =  
    (SID_NAME = +ASM1)  
    (ORACLE_HOME = /u01/app/oracle/product/10.2.0/db_1)  
)  
#(SID_DESC =  
# (SID_NAME = PLSExtProc)  
# (ORACLE_HOME = /u01/app/oracle/product/10.2.0/db_1)  
# (PROGRAM = extproc)  
#)  
)
```

5.1.1.2 Interaction with the Listener

Starting and stopping the listener is done via Oracle CRS as follows: -

```
$> srvctl stop listener -n lprpbdb001
```

```
$> srvctl start listener -n lprpbdb001
```

Checking the status of the listener and it's services is done as follows: -

```
$> lsnrctl status LISTENER LPRPBDB001
```

```
Listener Parameter File      /u01/app/oracle/product/10.2.0/db_1/network/admin/listener.ora
Listener Log File          /u01/app/oracle/product/10.2.0/db_1/network/log/listener_lprpbdb001.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=172.18.50.16) (PORT=1529)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=172.18.50.1) (PORT=1529)))
Services Summary...
Service "+ASM1" has 1 instance(s).
  Instance "+ASM1", status UNKNOWN, 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 "BRDB_XPT" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
```



```
Service "SYS$STRADMIN.BRDB_CAPTQ.BRDB" has 1 instance(s).
  Instance "BRDB1", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Executing `lsnrctl services` `LISTENER_LPRPBDB001` 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.
- `BRDB_DGB` and `SYSS$STRADMIN.BRDB_CAPTQ.BRDB` are Oracle defined services and relate to and are used by Data Guard and Streams respectively.

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

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/admin/<DB>/adump`
- `/u01/admin/<DB>/bdump`
- `/u01/admin/<DB>/cdump`
- `/u01/admin/<DB>/udump`
- `$ORACLE_HOME/network/log`
- `$ORACLE_HOME/rdbms/log`

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
------	-------------------	-----------------



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



SYS (See Section 5.1.3.1)	Oracle Grid Control Oracle Data Guard	Grid Control Agents will be unable to logon Standby Database log shipping and coordination will fail
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	EMDB Interface will fail to refresh branch info
OMDBUSER	The OMDB Interface	OMDB Interface will fail to refresh branch info
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
STRADMIN	Oracle Streams	Streams administration will not be possible after change
TRBALUSER[1-4]	Training Counter Connections	Counter training will not be possible
TWS TWSSUP	The TWS Scheduler	All schedules will fail to run

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 change without the other, all remote logons will fail with an "Insufficient Privileges" ORA- error.

When these passwords are changes, for whatever reason, the changes when done, **must** be done in sync with the respective password file(s). 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 **BRSS** 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:

```
$> sqlplus '/as sysdba'  
SQL> ALTER USER SYS IDENTIFIED BY b0bsy0uruncl3;  
SQL> EXIT;
```

Recreate the password file:

```
$> cd $ORACLE_HOME/dbs  
$> orapwd file=/u01/app/oracle/product/10.2.0/db_1/dbs/orapwBRDB3  
password=b0bsy0uruncl3 entries=5
```

Note: The process for changing the ASM password is the same as that for the database instance.

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 BladeFrame(s) are protected by either being mirrored or the disks will be replicated via SRDF (EMC² DMXs only).

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.

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 the next day, therefore this process would have to fail twice 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.

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.

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_EXCEPTIONS 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>_PARTITION_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 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>_partitionCreates 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>'
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

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>_partitionCreates 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);

```

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>_partitionCreates
  (table_name,
   partition_range_value,
   status,
   status_date)
SELECT atp.table_name,

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

        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
                      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 [DES/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 1-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)?

Is there enough storage space, e.g. check `+SBRDB_FLASH, /archredo`? 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/admin/SBRDB/bdump/drcSBRDB1.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;
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



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/admin/[S|B]BRDB/bdump/drc[S|]BRDB[1-4].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 10.2.0.4.0 - 64bit Production

Copyright (c) 2000, 2005, Oracle. All rights reserved.

```

Welcome to DGMGRL, type "help" for information.

```

DGMGRL> connect /
Connected.
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

```



5.5 Oracle Streams

5.5.1 Introduction

Streams 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] and Low Level Design [DES/APP/LLD/0151].

5.5.2 Assumptions

A single-source replication environment is configured and has the following characteristics:

- One Capture process named BRDB_CAPTURE located in BRDB node 1
- One Propagation process named BRDB_PROPAG located in BRDB node 1.
- One Apply process named BRSS_APPLY located in BRSS node 1.

5.5.3 Overview

This section only cover the diagnostics are required by the support persons when troubleshooting the Streams processes.

5.5.4 Troubleshooting

The following is a list of tables and views that are useful, in troubleshooting Streams issues. This is basically “reference” information (more detailed information can be found in the Oracle Streams administration guide):

Capture Process

dba_capture:	basic status, error info
v\$streams_capture:	detailed current status info
dba_capture_parameters:	configuration information

Propagate Process

dba_propagation:	basic status, error info
v\$propagation_sender:	detailed current status

Apply Process

dba_apply:	basic status, error info
v\$streams_apply_reader:	status of the current apply reader
v\$streams_apply_server:	status of current apply server(s)
v\$streams_apply_coordinator:	overall status, latency info
dba_apply_progress	
dba_apply_parameters:	configuration information

“Miscellaneous” Tables and Views

v\$buffered_queues: view that displays the current and cumulative number of messages enqueued and spilled, for each buffered queue.

sys.streams\$_apply_spill_msgs_part: table that the apply process uses, to “spill” messages from large transactions to disk.

system.logmnr_restart_ckpt\$: table that holds capture process “checkpoint” information.



Whenever there is an error or performance problems present themselves, the first place to check will be to note any Grid Control errors, then the alert_<SID>.log and finally the relevant database instance trace files. Messages about each capture process, propagation, and apply process are recorded in trace files for the database in which the processes are running.

- Capture - BRDB1
 - alert files = /u01/admin/BRDB/bdump/alert_BRDB1.log
 - trace files = /u01/admin/BRDB/bdump/brdb1_cnnn_iiii.trc
- Propagation - BRDB1
 - alert files = /u01/admin/BRDB/bdump/alert.log
 - trace files = /u01/admin/BRDB/bdump/brdb1_jnnn_iiii.trc
- Apply - BRSS1
 - alert files = /u01/admin/BRSS/bdump/alert.log
 - trace files = /u01/admin/BRSS/bdump/brdb1_pnnn_iiii.trc

NOTE

nnnn: refers to the Oracle process.
 iiii: refers to the UNIX process number.
 e.g. `brdb1_cjq0_14883.trc`

5.5.4.1 Troubleshooting Capture Problems

The Oracle Streams Capture process resides within the Branch Database. It captures all data (DML/DDL) changes to objects own by OPS\$BRDB. At time, the Capture process simply stops capturing the changes or running very slow. Some of the useful methods describes in this section can use to diagnose the problem and resolve them: -

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 status FROM dba_capture WHERE capture_name = 'BRDB_CAPTURE' ;
```

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
  FROM dba_capture
 WHERE status = 'ABORTED' AND capture_name = 'BRDB_CAPTURE' ;
```

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 = 'BRDB_CAPTURE' ;
```

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.

DICTIONARY INITIALIZATION: Processing a dictionary build.

MINING (PROCESSED SCN = scn_value): Mining a dictionary build at the SCN scn_value.

LOAD (step X of 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.

- 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'.

5.5.4.2 Troubleshooting Propagation Problems

The Oracle Streams propagation process resides within the Branch Database. It propagates the captured events to the destination queue in the target database (BRSS). Some of the useful methods describes in this section can use to diagnose the propagation problem and resolve them.

- Check the database link is configured correctly and active.
- Check whether a propagation job status is enabled, disabled or aborted by querying the DBA_PROPAGATION dictionary view:

```
select status from dba_propagation where propagation_name =
'BRDB_PROPG';
```



If status is 'disabled' or 'aborted' then check the error message:

```
select status,destination_dblink,error_date, error_message
from dba_propagation
where propation_name = 'BRDB_PROPAG' ;
```

Correct the error and restart the propagation process

```
begin
    DBMS_PROPAGATION_ADMIN.START_PROPAGATION( 'BRDB_PROPAG' );
end;
```

- If the propagation job is enabled, but is not propagating messages, then try stopping and restarting the propagation.
- Check propagation schedule is enabled and associated with a job queue process, any failures or errors received, the date and time the propagation schedule will be started

```
select SCHEDULE_DISABLED,PROCESS_NAME,
       LAST_RUN_DATE,NEXT_RUN_DATE,
       TOTAL_NUMBER,FAILURES,
       LAST_ERROR_TIME,LAST_ERROR_MSG,
  from    dba_queue_schedules s, dba_propagation p
 where   s.DESTINATION = p.destination_dblink ;
```

SCHEDULE_DISABLE = 'N' Enable

SCHEDULE_DISABLE = 'Y' Disable

- Determine the number of messages sent and the number of messages that have been acknowledged.

```
select queue_name,schedule_status,
       high_water_mark,acknowledgement
  from v$propagation_sender;
```

5.5.4.3 Troubleshooting Apply Problems

The Oracle Streams Apply process resides within the Branch Support Database. It dequeues all the captured data and applies them to the OPS\$BRDB schema. Some of the useful methods described in this section can be used to diagnose the apply problem and resolve them.

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 status
  FROM dba_apply
 WHERE apply_name = 'BRSS_APPLY' ;
```

The possible values are ENABLED, DISABLED and ABORTED.

If the apply process is disabled, then try restarting it: -

```
DBMS_APPLY_ADMIN.START_APPLY( apply_name => 'BRSS_APPLY') ;
```

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 = 'BRSS_APPY';
```

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, (num_msgs - spill_msgs) mem_msgs,
       spill_msgs
  FROM v$buffered_queues
 WHERE queue_name = 'BRSS_APPLQ';
```

Or using the v\$streams_apply_coordinator view: -

```
SELECT total_received,
       total_applied,
       total_errors, (total_assigned - total_rollbacks + total_applied)
  being_applied
  FROM v$streams_apply_coordinator
 WHERE apply_name = 'BRSS_APPY';
```

Check the Error Queue

When an apply process cannot apply a message, it moves the event and all the other events in the same transaction into the error queue.

Query DBA_APPLY_ERROR to determine if there are errors in the error queue.

```
SELECT local_transaction_id,
       message_number,error_message
  FROM    dba_apply_error
 WHERE  apply_name = 'BRSS_APPY';
```

If there are apply errors, then you can either try to re-execute the transactions that encountered the errors, or you can delete the transactions. If you want to re-execute a transaction that encountered an error, then first correct the condition that caused the transaction to raise an error.

The execute_error procedure re-executes the specified error transaction.

Syntax:

```
DBMS_APPLY_ADM.EXECUTE_ERROR(local_transaction_id IN VARCHAR2,
                               execute_as_user    IN BOOLEAN DEFAULT false);
```

The error transaction can be dropped from the from the error queue by invoking the delete_error procedure. The delete_error procedure deletes the specified error transaction.

Syntax:

```
DBMS_APPLY_ADM.DELETE_ERROR(local_transaction_id IN VARCHAR2);
```

5.5.4.4 Working with Apply Errors

The Oracle Streams Apply process will store all rows of every transaction that fails to apply. These failed transactions with their associated errors are available to query from DBA_APPLY_ERROR. Errorred transactions can be applied once the problem that caused the failure has been rectified (if possible). The following is a little help in finding them and extracting them to help with their resolution: -

```
set lines 140
set pages 45
SELECT TO_CHAR(error_creation_time,'YYYY/MM/DD') created,
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

        error_number,
        COUNT(1)
    FROM dba_apply_error
GROUP BY error_number, TO_CHAR(error_creation_time,'YYYY/MM/DD')
ORDER BY 1, 2;

CREATED      ERROR_NUMBER      COUNT(1)
-----
2009/06/23      1403            6
2009/06/24      1403            6

```

Once you have the list of errors, choose whichever error you may be interested in and use it in the following query, which will provide you with the **error_message** and the **local_transaction_id**:-

```

SET lines 140
SET pages 80
col error_message FOR a70
SELECT local_transaction_id, error_message
  FROM dba_apply_error
 WHERE error_number = 1403;

```

LOCAL_TRANSACTION_ID	ERROR_MESSAGE
23.28.244	ORA-01403: no data found
	ORA-01403: no data found
3.33.140771	ORA-01403: no data found
17.30.370	ORA-01403: no data found

Using the **local_transaction_id** you can perform the following procedure execution of **PKG_BRSS_STREAMS.PR_PRINT_TRANSACTION**, which will show every record affected by the error in question, information such as the table, the columns affected the before and after values, etc. are shown:-

```

SET SERVEROUTPUT ON
BEGIN
    stradmin.pkg_brss_streams.pr_print_transaction('3.33.140771');
END;
/
----- Local Transaction ID: 3.33.140771
----- Source Database: BRDB
-----Error in Message: 1
-----Error Number: 1403
-----Message Text: ORA-01403: no data found

--message: 1
type name: SYS.LCR$_ROW_RECORD
source database: BRDB
owner: OPS$BRDB
object: BRDB_SYSTEM_PARAMETERS
is tag null: Y
command_type: UPDATE
old(1): PARAMETER_NAME
BRDB_PDEL_TO_LFS_STOP_YN
old(2): VERSION_NUMBER
1
old(3): UPDATE_TIMESTAMP
typename is SYS.TIMESTAMP

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

old(4): PARAMETER_TEXT
N
new(1): UPDATE_TIMESTAMP
typename is SYS.TIMESTAMP
new(2): PARAMETER_TEXT
Y
transaction name:
--message: 2
type name: SYS.LCR$_ROW_RECORD
source database: BRDB
owner: OPS$BRDB
object: BRDB_PROCESS_CONTROL
is tag null: Y
command_type: UPDATE
old(1): INSTANCE_NAME
BRDB_PDEL_TO_LFS_STOP_YN
old(2): RUN_NUMBER
2
old(3): PROCESS_NAME
BRDBX011
old(4): SYSTEM_DATE
23-JUN-09
old(5): END_DATE
new(1): END_DATE
23-JUN-09
transaction name:
--message: 3
type name: SYS.LCR$_ROW_RECORD
source database: BRDB
owner: OPS$BRDB
object: BRDB_PROCESS_AUDIT
is tag null: Y
command_type: INSERT
new(1): PROCESS_SEQUENCE_NUMBER
432918
new(2): INSTANCE_NAME
BRDB_PDEL_TO_LFS_STOP_YN
new(3): PROCESS_NAME
BRDBX011
new(4): BLOCK_CHANGES
new(5): BLOCK_GETS
new(6): CONSISTENT_CHANGES
new(7): CONSISTENT_GETS
new(8): ORACLE_SERIAL#
new(9): ORACLE_SID
new(10): RUN_NUMBER
new(11): PHYSICAL_READ
new(12): UNIX_PID
new(13): INSERT_DATE
23-JUN-09
new(14): SYSTEM_DATE
23-JUN-09
new(15): START_FINISH_INDICATOR
F
new(16): USER_NAME
OPS$BRDBBLV2
transaction name:

```

Once again, the errored transaction can be applied once the cause of the failed apply has been identified.

```
exec dbms_apply_adm.execute_error('3.33.140771');
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



One could also print all the current errors by executing PKG_BRSS_STREAMS.PR_PRINT_ERRORS. This procedure prints **all** errored transactions regardless of **transaction_id** or **error_number**.

```
BEGIN
    stradmin.pkg_brss_streams.print_all_errors;
END;
/
```



5.5.4.5 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#;
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



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 = 'BRDB_CAPTURE';

  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
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```
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;
/
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



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 = 'BRDB_CAPTURE' ;
```

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
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```
AND r.serial# = s.serial#
AND r.apply_name = ap.apply_name;
```

v. The following query displays the information about each transaction currently being applied for which the apply process has spilled messages:

```
column apply_name      heading 'Apply Name' format A20
column 'Transaction ID' heading 'Transaction ID' format A15
column first_scn      heading 'First SCN'    format 99999999
column message_count   heading 'Message Count' format 99999999

SELECT apply_name,
       xidusn || '.' ||
       xidslt || '.' ||
       xidsqn "Transaction ID",
       first_scn,
       message_count
  FROM dba_apply_spill_txn;
```

vi. 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 BRDBX015 – Transaction Correction Tool

The transaction correction tool module BRDBX015.sh will allow Support Service Centre to correct transactions by inserting balancing records to transactional/accounting/stock tables in the BRDB system. It takes two parameters, the file name containing the insert statement and the branch code. If the process completes successfully, the insert statement is audited in BRDB_TXN_CORR_TOOL_JOURNAL. If there is an error, the entire transaction is rolled back and nothing is written to the database. The module uses process audit and does not use process control (allows multiple runs).

The file containing the insert statement must be copied to the /app/brdb/trans/support/brdbx015/input directory on the Linux box, and the module run from the directory /app/brdb/trans/support/brdbx015. If the module completes successfully, the file will be moved to /app/brdb/trans/support/brdbx015/output. A log file will be written to /bvnw01/brdb/brdbx015/log using the file name template <transaction_file>_<CCYYMMDDHHMISS>.log

This module can be run only by the Linux user “supporttooluser” which has only the necessary privileges required to run the module. The module will call a package procedure which runs under Oracle user ‘OPS\$SUPPORTTOOLUSER’ which allows inserts only into selected tables. This is a powerful tool which has inherent risks and care must be taken when constructing the insert statement. The SQL statement must begin with an ‘INSERT INTO’ clause and can only insert one row into the corresponding transactional table. This is validated in the tool and will raise an error if the condition is not met.

The format of the SQL statement should be based on the templates supplied in Appendix C.

The following tables have been granted insert privileges to OPS\$SUPPORTTOOLUSER:-

BRDB_RXAPS_TRANSACTIONS
BRDB_RXBUREAU_TRANSACTIONS
BRDB_RXCUT_OFF_SUMMARIES
BRDB_RXDCS_TRANSACTIONS
BRDB_RXEPOSS_EVENTS
BRDB_RXEPOSS_TRANSACTIONS
BRDB_RXNWB_TRANSACTIONS
BRDB_RXREP_EVENT_DATA
BRDB_RXREP_SESSION_DATA

5.6.1 Parameters

The tool must be supplied with 2 parameters:

- Transaction File Name (not including path) e.g. t1.sql
- Branch Code (numeric) e.g. 8009

5.6.2 Scheduling

This task is scheduled on an ad hoc basis, as and when transaction corrections need to be applied.

5.6.3 Sample output

This is an example of the output written to standard output and the log file when the module is successful:



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

Wed 21-Oct-2009 14:35:08 Starting BRDBX015.sh
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: Debug message level for this program is 0
Wed 21-Oct-2009 14:35:08 BRDBX015.sh:
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: In check_parameters()
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: ORACLE_HOME = /u01/app/oracle/product/10.2.0/db_1
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: ORACLE_SID = BRDB1
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: TRANS_FILE = ./input/t1.sql
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: BRANCH_CODE = 9004
Wed 21-Oct-2009 14:35:08 BRDBX015.sh:
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: Script <BRDBX015.sh> started on Wed Oct 21 14:35:08 BST
2009
Wed 21-Oct-2009 14:35:08 BRDBX015.sh:
Wed 21-Oct-2009 14:35:08.750 Started PKG_BRDB_TXN_CORRECTION.LOAD_DATA
Wed 21-Oct-2009 14:35:08.750 Version information: $Logfile: /HNG-
X/035.CTR020[00.90]/BRDB/Database and Schema Build/PLSQL Objects/pkg_brdb_txn_correction_body.sql
$$Revision: 29 $
Wed 21-Oct-2009 14:35:08.750 This Feed does not use process control
Wed 21-Oct-2009 14:35:08.995 Number of rows inserted = 1
Wed 21-Oct-2009 14:35:09.011 Completed PKG_BRDB_TXN_CORRECTION.LOAD_DATA

PL/SQL procedure successfully completed.

Wed 21-Oct-2009 14:35:08 Return code is 0
Wed 21-Oct-2009 14:35:08 BRDBX015.sh:
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: BRDBX015.sh ran successfully
Wed 21-Oct-2009 14:35:08 exit 0
Wed 21-Oct-2009 14:35:08 BRDBX015.sh:
Wed 21-Oct-2009 14:35:08 BRDBX015.sh: Finished on Wed Oct 21 14:35:09 BST 2009
Wed 21-Oct-2009 14:35:08

```

5.6.4 Diagnostics

The module may fail for one of the following reasons

- May not be logged in as 'supporttooluser'.
- Transaction file containing SQL statement is not present in /app/brdb/trans/support/brdbx015/input directory.
- The Oracle directory name 'BRDBX015_DIR' must be mapped to physical directory /app/brdb/trans/support/brdbx015/input. Connect to the Linux box as user 'supporttooluser'. Login to SQL and run the following command:-

```

SELECT owner, substr(directory_path,1,40) directory_path
FROM all_directories
WHERE directory_name = 'BRDBX015_DIR'
/

```

The above statement must return 1 row.

- SQL statement does not begin with 'INSERT INTO' statement or contains more than one insert statement.
- The following in-line select statement has not been added to the end of the insert statement

```

(SELECT fhom.branch_accounting_code,
     fhom.fad_hash,
     tctc.current_jsn
  FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
       ops$brdb.brdb_txn_corr_tool_ctl tctc
 WHERE fhom.branch_accounting_code = :bind_branch_code
   AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

- Check the insert statement for any syntax errors.



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 [DES/APP/LLD/0152]. This section details the failover procedures in changing the role of a database, in our case BRDB or SBRDB. The method described in sections 6.1 and 6.2, 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 6.4 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 6.2 has been provided for, in the event that the Data Guard Broker is unavailable.

6.1 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
Ass um ptio ns	i. User is logged onto the Standby Database Server as <i>oracle</i> . 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.	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.	<pre>\$> . oraenv [now type in SBRDB1] \$> dgmgrl DGMGRL> CONNECT sys/<sys password></pre>
2.	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.	<pre>DGMGRL> FAILOVER TO 'SBRDB' ;</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



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.	<p>Issue the SHOW CONFIGURATION command to verify the failover.</p>	<pre>DGMGR> SHOW CONFIGURATION;</pre> <p><i>You should see ...</i></p> <pre>Configuration Name: BRDB_DATAGUARD_CFG Enabled: YES Protection Mode: MaxPerformance Fast-Start Failover: DISABLED Databases: BRDB - Physical standby database (disabled) SBRDB - Primary database</pre> <p><i>Current status for "BRDB_DATAGUARD_CFG":</i> SUCCESS</p>
4.	<p>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.</p>	<pre>DGMGR> SHOW DATABASE 'BRDB' ;</pre> <p><i>You should see "Enabled: NO" and a message returned, similar to "Current status for "BRDB": Error: ORA-16661: the standby database needs to be reinstated".</i></p>
5.	<p>Check that all the indexes – database wide – are available for use.</p> <p>If any indexes are marked as 'UNUSABLE' they need to be rebuilt. See example to the right of this cell.</p>	<pre>SQL> SELECT owner, index_name FROM dba_indexes WHERE status = 'UNUSABLE' ;</pre> <pre>SQL> ALTER INDEX <OWNER>.<index> REBUILD ONLINE [PARALLEL <# CPU's>] ;</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>6a.</p> <p>Depending on the timing of the failover to Standby, there is a possibility that the BladeFrame may not have the full complement of four pBlades configured and started for the Standby cluster.</p> <p>If not already done, add the 3 additional pBlades into the BladeFrame: -</p> <ol style="list-style-type: none"> 1. Physically "plug" the pBlades into the frame. 2. Configure and start the pServers. 3. Once you're able to log on as oracle, bring up the remaining database instances starting with SBRDB2, e.g. <code>srvctl start instance -d SBRDB -i SBRDB2</code> 	<p>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 its new role as Primary.</p> <p>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.</p> <p>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='*' ;".</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;"><u>Parameter</u></th><th style="text-align: left; padding: 2px;"><u>Future Value</u></th><th style="text-align: left; padding: 2px;"><u>Likely Current Value</u></th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">audit_trail</td><td style="padding: 2px;">DB</td><td style="padding: 2px;">NONE</td></tr> <tr> <td style="padding: 2px;">cluster_database_instances</td><td style="padding: 2px;">4</td><td style="padding: 2px;">1</td></tr> <tr> <td style="padding: 2px;">control_file_record_keep_time</td><td style="padding: 2px;">21</td><td style="padding: 2px;">NULL</td></tr> <tr> <td style="padding: 2px;">instance_number</td><td style="padding: 2px;">1-4</td><td style="padding: 2px;"><See action 1 below></td></tr> <tr> <td style="padding: 2px;">instance_name</td><td style="padding: 2px;">NULL</td><td style="padding: 2px;"><See action 2 below></td></tr> <tr> <td style="padding: 2px;">local_listener</td><td style="padding: 2px;">LISTENER_<node></td><td style="padding: 2px;"><See action 3 below></td></tr> <tr> <td style="padding: 2px;">log_archive_dest_3</td><td style="padding: 2px;">NULL</td><td style="padding: 2px;">'LOCATION=/archredo/<DB> OPTIONAL'</td></tr> <tr> <td style="padding: 2px;">log_archive_dest_state_3</td><td style="padding: 2px;">NULL</td><td style="padding: 2px;">'ENABLE'</td></tr> <tr> <td style="padding: 2px;">sessions</td><td style="padding: 2px;">2205</td><td style="padding: 2px;">610</td></tr> <tr> <td style="padding: 2px;">thread</td><td style="padding: 2px;">1-4</td><td style="padding: 2px;"><See action 4 below></td></tr> </tbody> </table> <p>[1] An "ALTER SYSTEM ... SID=' SBRDB2'" statement required on each instance, e.g. <code>instance_number=2</code> for node 2, <code>3</code> for node 3, et cetera.</p> <p>[2] An "ALTER SYSTEM ... SID=' SBRDB2'" statement required on each instance, e.g. <code>instance_name=' SBRDB2'</code> for node 2, <code>' SBRDB3'</code> for node 3, et cetera.</p> <p>[3] An "ALTER SYSTEM ... SID=' SBRDB2'" statement required on each instance, e.g. <code>local_listener='LISTENER_<node002>'</code> for node 2, <code>'LISTENER_<node003>'</code> for node 3, etc.</p> <p>[4] An "ALTER SYSTEM ... SID=' SBRDB2'" statement required on each instance, e.g. <code>thread=2</code> for node 2, <code>3</code> for node 3, et cetera.</p>	<u>Parameter</u>	<u>Future Value</u>	<u>Likely Current Value</u>	audit_trail	DB	NONE	cluster_database_instances	4	1	control_file_record_keep_time	21	NULL	instance_number	1-4	<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	1-4	<See action 4 below>
<u>Parameter</u>	<u>Future Value</u>	<u>Likely Current Value</u>																																
audit_trail	DB	NONE																																
cluster_database_instances	4	1																																
control_file_record_keep_time	21	NULL																																
instance_number	1-4	<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	1-4	<See action 4 below>																																



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



6c.	<p>Create a text file "copy" of the current spfile (server parameter file) on both the Primary (BRDB) and the Standby SBRDB nodes.</p> <p>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.</p>	<pre>. 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</pre>
7.	<p>After failover, the "new" Primary database cluster (lprpbds001 - 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.</p> <p>On the first node: -</p> <p>The service should already be enabled, so all that needs to be done is to start the service.</p> <p>If starting the service is unsuccessful for some reason, then try enabling the service.</p> <p>Once again, after enabling the service, try starting the service again.</p> <p>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.</p>	<p><u>Syntax</u></p> <pre>srvctl add service -d <db_unique_name> -s <service_name> -r <preferred_list></pre> <p><u>Command</u></p> <pre>srvctl add service -d SBRDB -s BRDB -r SBRDB1,SBRDB2,SBRDB3,SBRDB4</pre> <pre>srvctl start service -d SBRDB -s BRDB</pre> <pre>srvctl enable service -d SBRDB -s BRDB</pre> <p>[A.] crs_stat -t</p> <p>[B.] lsnrctl status listener_lprpbds001</p>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



The correct output seen, should be similar to the following: -

[A.]

Name	Type	Target	State	Host
ora....DB1.srv	application	ONLINE	ONLINE	lprpbds001
ora....DB2.srv	application	ONLINE	ONLINE	lprpbds002
ora....DB3.srv	application	ONLINE	ONLINE	lprpbds003
ora....DB4.srv	application	ONLINE	ONLINE	lprpbds004
ora....BRDB.cs	application	ONLINE	ONLINE	lprpbds001

[B.]

```

LSNRCTL for Linux: Version 10.2.0.4.0 - Production on ...
Copyright (c) 1991, 2007, Oracle. All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=lprpbds001-
vip) (PORT=1529) (IP=FIRST)))
STATUS of the LISTENER
-----
Alias                      LISTENER_LPRPBDS001
Version                    TNSLSNR for Linux: Version 10.2.0.4.0 - Production
Services Summary
...
Service "BRDB" has 1 instance(s).
  Instance "SBRDB1", status READY, has 1 handler(s) for this service
...

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>The Primary database cluster (lprpbdb001 – 4) after failover will be the former Standby database cluster (lprpbds001 - 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.</p> <p>In order to accomplish this, the following should be followed: -</p> <p>[1.] Update ACD001 to change the PBDB00X-VIP alias to point to associated BDS servers, e.g. (lprpbds001 - 4)</p> <p>[2.] Flush DNS cache on all Linux DNS servers (DNP and DNS)</p> <pre>/usr/sbin/rndc flush</pre> <p>[3.] Clear the DNS cache on all servers that address BDB on VIP alias</p> <pre>/usr/sbin/nsqd --invalidate=hosts</pre> <p>[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.</p> <p>[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.</p> <pre>sqlplus lvbaluser1/<lvbaluser1 password>@BRDB</pre> <p>[6.] To allow TWS to access and run schedules on the new Primary nodes, edit the following file and change all occurrences of "lprpbdb" to "lprpbds": -</p> <pre>tws.cpu</pre> <p><u>WARNING</u> 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.</p>	<p>8.</p>
<p><u>WARNING</u></p> <p>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.</p> <p>9.</p> <p>Edit the crontab.</p> <p>Once the crontab has loaded (output should reflect schedule shown below).</p> <p>Use vi commands to add a "#" in front of every line where one does not exist. Then save and quit the file.</p>	<p>As the oracle user ...</p> <pre>\$> crontab -e</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

# HouseKeeping
0 6  * * *  /app_sw/brdb/sh/HousekeepWrapper.sh SBRDB > cron.sbrdb.out 2>&1
5 6  * * *  /app_sw/brdb/sh/HousekeepWrapper.sh +ASM > cron.asm.out 2>&1
#
# RMANBackup
#0 4  * * Sun /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 0 > cron.rb.sun.out
2>&1
#0 4  * * Mon /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 1 > cron.rb.mon.out
2>&1
#0 4  * * Tue /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 1 > cron.rb.tue.out
2>&1
#0 4  * * Wed /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 0 > cron.rb.wed.out
2>&1
#0 4  * * Thu /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 1 > cron.rb.thu.out
2>&1
#0 4  * * Fri /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 1 > cron.rb.fri.out
2>&1
#0 4  * * Sat /app_sw/brdb/sh/RMANBackupWrapper.sh SBRDB 1 > cron.rb.sat.out
2>&1
#

```

10.	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 6.3, as one must re-create the primary database from a copy of the new primary database.
	Manual <i>Complete Failover</i> through DGMGRL is complete.

Table 3: Data Guard Failover Procedure.



6.2 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 <i>oracle</i> . 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.	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 <i>instance_name</i> , <i>host_name</i> and <i>status</i> .	<pre>\$> . oraenv [now type in SBRDB1] \$> sqlplus '/as sysdba' SQL> SELECT * FROM v\$instance;</pre>
2.	Initiate the failover by issuing the following. <i>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.</i>	<pre>SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH FORCE;</pre>
3.	Convert the physical standby database to the production role. <i>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.</i>	<pre>SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



4a.	Open the new production (primary) database by issuing the following statement.	<pre>SQL> ALTER DATABASE OPEN;</pre>
4b.	<p>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.</p> <p>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.</p>	<pre>SQL> SELECT value FROM v\$dataguard_stats WHERE name = 'standby has been open'; SQL> SHUTDOWN IMMEDIATE; SQL> STARTUP;</pre>
5.	Check that all the indexes – database wide – are available for use.	See Step [5.] of Section 6.1
6.	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 its new role as Primary.	See Step [6.] of Section 6.1
7.	<p>After failover, the “new” Primary database cluster (lprpbds001 - 4) and database, SBRDB, must accept connections from all applications without changing any application connection properties.</p> <p>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.</p>	See Steps [7.] and [8.] of Section 6.1
8.	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 original Standby Database deployment handover notes as one must re-create the primary database from a copy of the new primary database.	
	Manual <i>Complete Failover</i> through SQL*Plus is complete.	

Table 4: SQL*Plus Failover Procedure.



6.3 Standby Database Re-instantiation

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
Ass um ptio ns	<p>i. User is logged onto the Standby Database Server as <i>oracle</i>.</p> <p>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.</p> <p>iii. Only one node should be used as the new standby database node.</p>	
1.	<p><u>New Prim Server</u></p> <p>Backup the new primary (SBRDB) database using RMAN. Ensure there is sufficient space on the device you specify as <RMAN DIR>.</p> <p>Logon to RMAN.</p> <p>Execute the backup commands as they appear, e.g. <code>run { ... }</code></p> <p>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. <code>list backup summary;</code></p>	<pre>\$> . 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' archive log all not backed up 1 times; } \$> cd <RMAN DIR> \$> ls -l</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



2.	<p><u>New Prim Server</u></p> <p>Ensure the entire backup, which will consist of a number of files is copied across from this server to the new standby server.</p> <p>Note: The backup must exist in the same location on both servers!</p>	<pre>\$> scp <RMAN DIR>/* pbdb001<RMAN DIR></pre>
3.	<p><u>Old Prim Server (node 1)</u></p> <p>Cleanup the old archive directory as it would be full of files that are no longer needed. Type YES, if prompted.</p> <p>Note: The "rm -r" is extremely destructive! Make sure you're on the correct server and that BRDB most definitely has been failed over.</p>	<pre>\$> . oraenv [now type in +ASM1] \$> asmcmd -p ASMCMD [+] > cd BRDB_FLASH/arch ASMCMD [+BRDB_FLASH/arch] > rm -r brdb*.arc</pre>
4.	<p><u>Old Prim Server (node 1)</u></p> <p>Set the environment for the new standby database.</p> <p>Log onto RMAN and execute the restore of the new primary as the new standby.</p> <p>Ensure there are no errors in this restore. Otherwise, fix the errors and run again.</p>	<pre>\$> . oraenv [now type in BRDB1] \$> \$ORACLE_HOME/bin/rman TARGET=sys/<SYS_PSWD>@sbrdb AUXILIARY / RMAN> duplicate target database for standby;</pre>
5.	<p><u>Old Prim Server (node 1)</u></p> <p>The standby database should already be mounted, but if not, mount the new standby database.</p>	<pre>SQL> ALTER DATABASE MOUNT STANDBY DATABASE;</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



6.	<p><u>Old Prim Server (node 1)</u></p> <p>Start the standby database in managed recovery mode.</p> <p>This must have completed successfully. To check that it has, query <code>v\$dataguard_status</code>.</p> <p>Also, check that the application of logs is performing well, query <code>v\$dataguard_stats</code>.</p>	<pre>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;</pre>
7.	<p><u>New Prim Server</u></p> <p>Ensure that the <code>tnsnames.ora</code> has an entry for the new primary.</p>	<pre>\$> cd /app_sw/sbrdb/standby \$> SBRDB_edit_tnsnames.sh -v -s lprpbds001</pre>
8.	<p>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: -</p> <pre>\$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</pre>	
	Manual re-instantiation of Standby Database Complete.	

Table 5: Primary Re-instantiation Procedure.



6.4 Opening Standby Database “READ ONLY”

As explained earlier, the changing of roles of the Standby to Primary is irreversible, so in a scenario where, for whatever reason, the Standby database needs to be read to check data available only in an “OPEN” state then the steps in the following table will apply.

There is a (an extremely critical) caveat to performing this operation. This operation opens the SBRDB database in “READ ONLY” mode. Oracle Data Guard is aware that this operation would have taken place and keeps a record of each time this is done. It is important to note that performing this operation changes the way the database is recovered if it ever becomes Primary. This is noted by the step detailed in [4b.] of Section 6.2.

Step	Description	Server Execution
Assumptions	<p>i. User is logged onto the Standby Database Server as <i>oracle</i>.</p> <p>ii. The application of archivelogs/redo on the Standby Database will be cancelled and will therefore fall behind Primary for the period the database is in a “READ ONLY” state and that the relevant authority to perform this task has been sought, being completely aware of the consequences detailed above.</p> <p>Recommendation: Open a second terminal session to the Standby Server and “tail -f” the SBRDB alert log.</p>	
1.	<p>Logon to the database as SYSDBA.</p> <p>Check that the standby database is applying logs at good rate, i.e. ensure that the lag isn’t too great.</p>	<pre>\$> . oraenv [now type in SBRDB1] \$> sqlplus '/as sysdba' SQL> set lines 140 SQL> set pages 45 SQL> col value for a20 SQL> select * from v\$dataguard_stats;</pre>
2.	<p>Cancel the real-time apply of archivelogs to standby.</p> <p>The alert log will show the following error which can safely be ignored: - “ORA-16037: user requested cancel of managed recovery operation”</p>	<pre>SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;</pre>
3.	<p>Open the Standby database in “READ ONLY” mode.</p> <p>The database is now open for any “SELECT” queries that may be run against it.</p>	<pre>SQL> ALTER DATABASE OPEN READ ONLY;</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



4.	<p>After having queried the information required, shutdown the database once again.</p>	<pre>SQL> SHUTDOWN IMMEDIATE;</pre>
4a.	<p>At this point it is possible that you may experience symptoms of the following bugs for Oracle 10.2.0.4:</p> <ul style="list-style-type: none"> - 6737051 - 7677840 (base bug is 6737051) - 8533262 (base bug is 7677840) <p>The ORA-00600 messages (see 4b. below) in the alert log can be ignored at this point, as at the time of the authoring of this document (June 2009) a fix for these bugs has not yet been released. In addition continuing with step 5. below does not reveal any additional "ORA-" errors and real-time apply continues as expected without hiccups.</p>	<pre>SQL> SHUTDOWN IMMEDIATE;</pre>
4b.	<p>The following are the messages you may see in the alert log: -</p> <pre>ORA-00600: internal error code, arguments: [kjcv04], [], [], [], [], [], [], [] ORA-00600: internal error code, arguments: [kjr_pool_free_all:resp], [0x0B4E44B98], [0x0BF73B138], [0x0BF73B138], [0], [8], [], []</pre>	
5.	<p>Bring the Standby database back up into a "MOUNTED" state.</p> <p>The Data Guard broker should take over at this point and start the real-time apply automatically. To check that this has indeed taken place, check that the following is evident in the alert log: -</p> <pre>"Completed: ALTER DATABASE RECOVER MANAGED STANDBY DATABASE THROUGH ALL SWITCHOVER DISCONNECT USING CURRENT LOGFILE"</pre> <p>Double-check that the database is correctly applying as expected. Using the dataguard broker check the status of the configuration. You should see: -</p> <pre>Current status for "BRDB_DATAGUARD_CFG": SUCCESS</pre>	<pre>SQL> CONNECT /as sysdba SQL> STARTUP NOMOUNT SQL> ALTER DATABASE MOUNT STANDBY DATABASE; \$> dgmgrl DGMGRRL> connect / DGMGRRL> show configuration;</pre>



6.	<p>If the above checks are inconclusive, start the real-time apply by manually.</p> <p>Check the alert log that messages similar to the following are being successfully logged: -</p> <p>“Media Recovery Log +SBRDB_FLASH/arch/brdb_000...”</p>	<pre>SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION PARALLEL 8;</pre>
	Manual opening of Standby in <i>READ ONLY</i> mode is complete.	

Table 6: Opening Standby Database Read-only

6.5 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, the likelihood of there needing to be software installations, UNIX patches, database software upgrades, database patches, etc. is high.

Therefore the following describes a possible means of accomplishing a software update across all standby nodes in order to keep them functionally in sync with ***lprpbds001*** (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	<p>These alternatives are presented at a high level and the level of detail required, is beyond the scope of this document.</p> <p>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).</p>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>Alternative A.</p> <p><u>Primary Cluster</u></p> <p>vii. SMC: Give the go-ahead that all schedules are held for the affected node(s)</p> <p>viii. DBA: Using Grid Control, initiate a blackout of all components on the affected nodes, e.g. agents, listeners, database instances, etc.</p> <p>ix. UNIX: Using the BladeFrame PAN Manager, shutdown the pServers which correspond to nodes 2, 3 and 4, e.g. <code>1prpbdb002 - 004</code></p> <p>x. UNIX: Remove the pBlades from the Primary BladeFrame</p> <p><u>Standby Cluster</u></p> <p>xi. UNIX: Place the pBlades from (iii.) into the Standby BladeFrame</p> <p>xii. UNIX: Using the BladeFrame PAN Manager, startup the pServers which correspond to nodes 2, 3 and 4, e.g. <code>1prpbds002 - 004</code></p> <p>xiii. DBA/UNIX/3rd Party: Perform the required change, installation, patch, etc.</p> <p>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 its former, fully operational state, including all Grid Control blackouts and notification to SMC. There must be no unresolved Grid Control alerts or exceptions in <code>BRDB_OPERATIONAL_EXCEPTIONS</code> at the end of this process.</p> <p>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 4.3.3.2 to logically bring the nodes/instances back into operation.</p>	<p>For Alternative B, the process defined in Alternative A above, should be followed in exactly the same way, e.g. power down and remove pBlade, move pBlade to standby, power up and make change/install and then reverse the process back to full operation.</p> <p>However, <i>the major difference with this alternative is that it should be completed for only 1 node at a time</i>, thus always leaving 3 nodes fully operational at all times.</p> <p>Alternative B cannot be run if the node in question has a specific TWS schedule/job running or perhaps that is still to be run, as the node-specific schedules will fail.</p>
<p>Alternative B</p>	

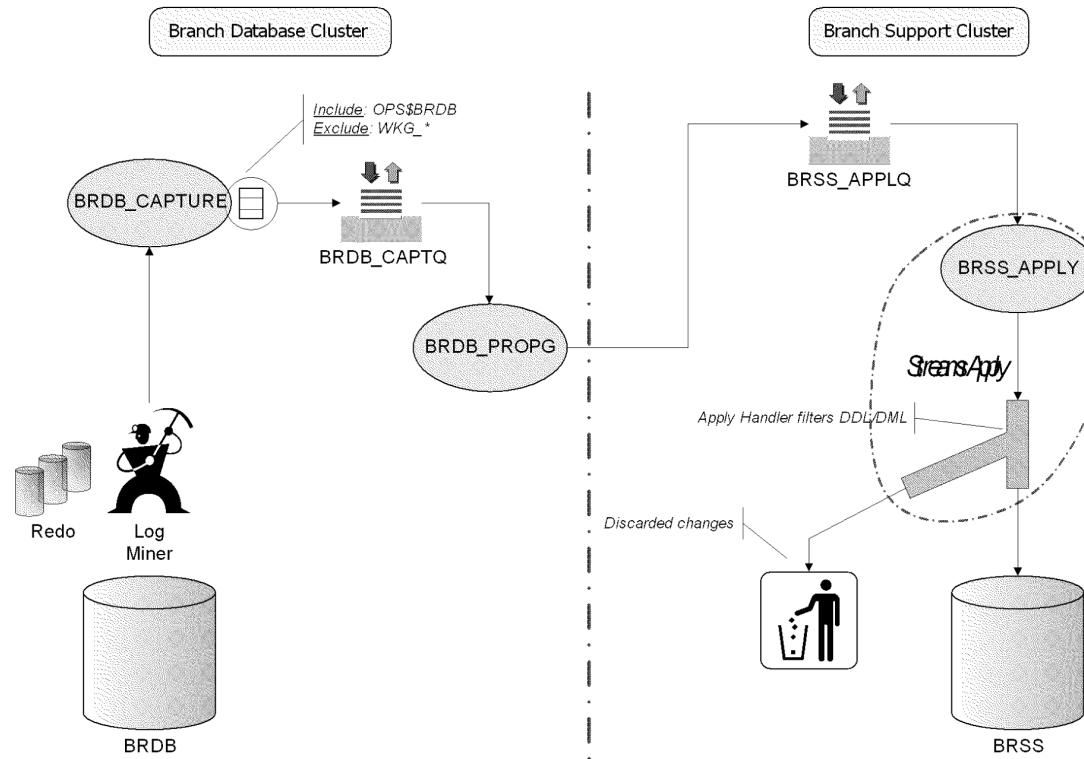
Table 7: Alternatives for Managing Software Installations on BDS nodes.



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 Streams. Oracle Streams 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).



7.1 Cleanup and Re-instantiation of Oracle Streams

7.1.1 Assumptions

Oracle Streams, as mentioned before, is an extremely effective replication technology but is rather complex. The cleaning up of streams is an action of **last resort!** Therefore you must have exhausted every other means of resolution before attempting the steps in the following procedure.



7.1.2 Cleanup and Re-instantiation Procedure

The procedure ...

Step	Description	Server Execution
	<p><u>Assumptions</u> User is logged on as oracle, on both the BRDB and BRSS servers, i.e. 2 sessions.</p> <p><u>Sections and Scenarios</u></p> <p>Steps 1 – 3 Pre-build Section: <i>This section is related to a scenario where BRDB has been built to phase x and a period of 12 hours or more occurs, such that the partitions are moved forward out-of-sync with BRSS. BRSS will also need to be built to phase x, but with the correct dates.</i></p> <p>Steps 4 – 11 Main: <i>This section is related to general cleanup and re-creation of the Streams configuration and associated objects.</i></p> <p>Step/Section A: <i>Specific cleanup scenario related to Step 5b.</i></p> <p>Step/Section B: <i>Streams recovery section with resolution of the following scenarios:-</i></p> <p><u>Scenario 1</u> - BDB has “moved on” in terms of partition management and BRS has fallen behind. There will be a number of partitions that exist in BDB for a particular table, but not in BRS. As a result data inserted into that table will fail in BRS as the required partition is missing.</p> <p><u>Scenario 2</u> - Scenario 1 is a confirmed case and required archivelogs, i.e. any single archivelog or a number of, or all archivelogs created after required_checkpoint_scn are permanently missing and hence Streams cannot recover.</p> <p><u>Scenario 3</u> - It has been determined that Streams has fallen behind to an unacceptable level and a decision has been made to re-instantiate the Streams configuration.</p>	



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>BDB:</p> <p>Login into the database (as oracle)</p> <p>Run the following SQL.</p> <p>Find the oldest partition date which exists in the output as "MIN_PARTS". In this example it is "20080803"</p>	<pre>. oraenv [now type in BRDB1] sqlplus '/as sysdba' SET lines 120 SET pages 45 SELECT table_name, MIN(SUBSTR(partition_name,LENGTH(partition_name)-7,9)) min_parts, MAX(SUBSTR(partition_name,LENGTH(partition_name)-7,9)) max_parts, COUNT(SUBSTR(partition_name,LENGTH(partition_name)-7,9)) count_parts FROM all_tab_partitions WHERE table_owner = 'OPS\$BRDB' AND composite = 'YES' GROUP BY table_name ORDER BY table_name;</pre> <table border="1"> <thead> <tr> <th>TABLE_NAME</th><th>MIN_PARTS</th><th>MAX_PARTS</th><th>COUNT_PARTS</th></tr> </thead> <tbody> <tr> <td>BRDB_DAILY_CUMULATIVE_SUMMARY</td><td>20080803</td><td>20080927</td><td>56</td></tr> <tr> <td>BRDB_DAILY_SUMMARY</td><td>20080927</td><td>20080927</td><td>1</td></tr> <tr> <td>...</td><td></td><td></td><td></td></tr> <tr> <td>BRDB_RX REP_EVENT_DATA</td><td>20080923</td><td>20080927</td><td>5</td></tr> <tr> <td>BRDB_RX REP_SESSION_DATA</td><td>20080803</td><td>20080927</td><td>56</td></tr> <tr> <td>BRDB_TX TRANSACTION TOTALS</td><td>20080923</td><td>20080927</td><td>5</td></tr> </tbody> </table>	TABLE_NAME	MIN_PARTS	MAX_PARTS	COUNT_PARTS	BRDB_DAILY_CUMULATIVE_SUMMARY	20080803	20080927	56	BRDB_DAILY_SUMMARY	20080927	20080927	1	...				BRDB_RX REP_EVENT_DATA	20080923	20080927	5	BRDB_RX REP_SESSION_DATA	20080803	20080927	56	BRDB_TX TRANSACTION TOTALS	20080923	20080927	5
TABLE_NAME	MIN_PARTS	MAX_PARTS	COUNT_PARTS																										
BRDB_DAILY_CUMULATIVE_SUMMARY	20080803	20080927	56																										
BRDB_DAILY_SUMMARY	20080927	20080927	1																										
...																													
BRDB_RX REP_EVENT_DATA	20080923	20080927	5																										
BRDB_RX REP_SESSION_DATA	20080803	20080927	56																										
BRDB_TX TRANSACTION TOTALS	20080923	20080927	5																										
<p>BRS:</p> <p>Now continue with the build instructions as described in the software handover note, but using the oldest partition date from (1.) as the input.</p>	<pre>BRSSSchemaBuild.sh -p 20080803</pre>																												



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>3.</p> <p><u>BRS:</u></p> <p>Using "BRSS Start of Day" BRSSC001, roll the database forward to the present day, i.e. the same date known to be configured for BRDB.</p> <p>This is most likely the greatest value of the "MAX_PARTS" column from the output shown in (1.). In this example it is "20080927".</p> <p>Note: BRDB and BRSS should now both be in sync.</p>	<table border="1"> <thead> <tr> <th>TABLE_NAME</th><th>MIN_PARTS</th><th>MAX_PARTS</th><th>COUNT_PARTS</th></tr> </thead> <tbody> <tr> <td>BRDB_DAILY_CUMULATIVE_SUMMARY</td><td>20080803</td><td>20080927</td><td>56</td></tr> <tr> <td>BRDB_DAILY_SUMMARY</td><td>20080927</td><td>20080927</td><td>1</td></tr> <tr> <td>...</td><td></td><td></td><td></td></tr> <tr> <td>BRDB_RX REP_EVENT_DATA</td><td>20080923</td><td>20080927</td><td>5</td></tr> <tr> <td>BRDB_RX REP_SESSION_DATA</td><td>20080803</td><td>20080927</td><td>56</td></tr> <tr> <td>BRDB_TX TRANSACTION TOTALS</td><td>20080923</td><td>20080927</td><td>5</td></tr> </tbody> </table>	TABLE_NAME	MIN_PARTS	MAX_PARTS	COUNT_PARTS	BRDB_DAILY_CUMULATIVE_SUMMARY	20080803	20080927	56	BRDB_DAILY_SUMMARY	20080927	20080927	1	...				BRDB_RX REP_EVENT_DATA	20080923	20080927	5	BRDB_RX REP_SESSION_DATA	20080803	20080927	56	BRDB_TX TRANSACTION TOTALS	20080923	20080927	5
TABLE_NAME	MIN_PARTS	MAX_PARTS	COUNT_PARTS																										
BRDB_DAILY_CUMULATIVE_SUMMARY	20080803	20080927	56																										
BRDB_DAILY_SUMMARY	20080927	20080927	1																										
...																													
BRDB_RX REP_EVENT_DATA	20080923	20080927	5																										
BRDB_RX REP_SESSION_DATA	20080803	20080927	56																										
BRDB_TX TRANSACTION TOTALS	20080923	20080927	5																										
<p>4a.</p> <p><u>BRS:</u></p> <p>If at this point, Streams exists and you're unable to continue with [4b.], skip to [5a.].</p>	<p>Skip to [5a.]</p>																												
<p>4b.</p> <p><u>BRS:</u></p> <p>Now continue with the build instructions as described in the handover note and run the BRSS Streams installation.</p>	<p>From /app_sw/brss/build/streams ... run ...</p> <pre>brdbStreams.sh -a BRSS -d +BRSS_FLASH -s <Streams Tablespace Size></pre>																												

HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE

5a.

BDB:

As in [4b.] complete the configuration of streams on BRDB. However, if Streams has already been configured, then the old components need to be removed first.

```
. oraenv
[now type BRDB1]

sqlplus stradmin/<stradmin_password>
[at the "SQL>" prompt, run the following]

SELECT propagation_name FROM dba_propagation;
[BRDB_PROPAG should be returned]

exec dbms_propagation_adm.drop_propagation('BRDB_PROPAG');
exec dbms_streams_adm.remove_streams_configuration;

col object_name for a40
SELECT object_name, object_type FROM user_objects WHERE object_name like
'%BRDB%';

[If any queue tables exist, remove them by running the following]
exec dbms_aqadm.drop_queue_table(queue_table =>
'STRADMIN.BRDB_CAPT_QUEUE_TABLE', force => TRUE);

[Then run this again]
exec dbms_streams_adm.remove_streams_configuration;
SELECT object_name, object_type FROM user_objects WHERE object_name like
'%BRDB%';

[Then repeat the process ... if any queue tables exist, remove them. However, if
any other objects exist, then follow [5b.] and the Oracle Metalink Note:
356323.1]
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



5b.	<p><u>BDB:</u></p> <p>Oracle Metalink Note 356323.1 → <i>[Shown in item "A" at the end of this procedure.]</i></p>	<p>Oracle Metalink Note: 356323.1 leads to --> OM Note: 203225.1 which explains more on the symptoms and how to resolve issues for 10g (else OM Note: 236898.1 for 9.2).</p>
5c.	<p><u>BDB:</u></p> <p>Continue with the cleanup, e.g. drop the STRADMIN user and then drop the old tablespace.</p>	<p>[Now logout and log back in as sysdba]</p> <pre>DROP USER stradmin CASCADE; DROP TABLESPACE brdb_streams_data INCLUDING CONTENTS AND DATAFILES;</pre>
5d.	<p><u>BRS:</u></p> <p>At this point [4b.] needs to be complete before continuing. Once complete, continue with [5e.]</p>	<p>Complete [4b.]</p>
5e.	<p><u>BDB:</u></p> <p>Once the cleanup is 100% complete. Run the BRDB Streams installation.</p>	<p>From /app_sw/brdb/build/streams/ ... run ...</p> <pre>brdbStreams.sh -a BRDB -d +BRDB_FLASH -s <Streams Tablespace Size></pre>
6.	<p><u>BDB:</u></p> <p>Check that Streams was successfully installed.</p>	<p>... su back to oracle ...</p> <pre>. oraenv [now type BRDB1]</pre> <p>sqlplus stradmin/<stradmin_password> [at the "SQL>" prompt, run the following]</p> <pre>SELECT SYSDATE FROM dual@BRSS; [If you don't see the system time, the installation failed. Retry.]</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



7.	BDB: Activate the Streams configuration.	From /app_sw/brdb/build/streams/ ... run ... streams_start.sh
8.	BDB: Check that Streams has activated. STATUS should be "ENABLED" Continue to Step 9 ...	sqlplus '/as sysdba' SELECT capture_name, queue_name, status, logminer_id, error_number, error_message FROM dba_capture
	The following steps (Steps 9 – 11), are required to "install" releases of Streams which were delivered subsequent to the original build. These releases were made in order that capture and apply processes perform better as well as introduce additional DML and DDL filters.	
9.	BRS: Execute the following DML in order to "reset" the patch release data. This will enable the re-running of the required database patch, BRSSPatch0035.sh, in order to install the change. As <i>oracle</i> execute the script ... One should only see 7 rows deleted.	DELETE FROM OPS\$BRSS.patch_release_management WHERE patch_name = 'BRSSPatch0035.sh' AND patch_step BETWEEN 20 AND 26; COMMIT; BRSSPatch0035.sh -v -i BRSS1 -s OPS\\$/BRSS



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>BRS:</p> <p>Execute the following DML. The patch, BRSSPatch0029.sh will install the change.</p> <p>10. As oracle execute the script ... One should only see 14 rows deleted.</p> <p>After the patch is complete, execute the following SQL. One should see a count of 45.</p>	<pre>DELETE FROM OPS\$BRSS.patch_release_management WHERE patch_name = 'BRSSPatch0029.sh'; COMMIT;</pre> <p>BRSSPatch0029.sh -v -i BRSS1 -s OPS\\$BRSS</p> <pre>SELECT COUNT(1) FROM dba_apply_dml_handlers WHERE object_owner = 'OPS\$BRDB'</pre>
<p>BDB:</p> <p>Execute the following DML. The patch, BRDBPatch0208.sh will install the change.</p> <p>11. As oracle execute the script ... One should only see 5 rows deleted</p>	<pre>DELETE FROM OPS\$BRDB.patch_release_management WHERE patch_name = 'BRDBPatch0208.sh'; COMMIT;</pre> <p>BRDBPatch0208.sh -v -i BRDB1 -s OPS\\$BRDB</p>

HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE

Following Metalink Note: 203225.1, here is an example of the sequence of commands used in a previous resolution to a streams-cleanup problem: -

```
SQL> col OBJECT_NAME format a30
SQL> SELECT object_name, object_type FROM user_objects WHERE object_name like '%BRDB%';
```

OBJECT_NAME	OBJECT_TYPE
AQ\$_BRDB_CAPT_QUEUE_TABLE_C	TABLE
AQ\$_BRDB_CAPT_QUEUE_TABLE_Y	INDEX

```
SQL> ALTER SESSION SET EVENTS '10851 trace name context forever, level 2';
```

```
Session altered.
```

A.

```
SQL> drop table AQ$_BRDB_CAPT_QUEUE_TABLE_C cascade constraints;
```

```
Table dropped.
```

```
SQL> SELECT object_name, object_type FROM user_objects WHERE object_name like '%BRDB%';
```

```
no rows selected
```

```
SQL> select object_name, object_type from dba_objects where owner = 'STRADMIN';
```

OBJECT_NAME	OBJECT_TYPE
BRSS	DATABASE LINK

```
SQL> exit
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



B.	<p><u>Item B</u> is a section of this document which deals with a few scenarios of Streams Failure and Recovery.</p>	
B1.	<p><u>BDB and BRS:</u></p> <p>The OPS\$BRDB schema on both BDB and BRS are not the same, in terms of structure, for whatever reason and need to be synchronised. <i>With the assumption that historical data and partitions are not required.</i></p> <p>Repeat all steps until there are no structural differences. When Streams is re-activated there can be no differences which will cause data to error on insertion to BRS.</p>	<p>Export the OPS\$BRDB schema on both databases:</p> <pre>exp system@brdb OWNER=ops\\$brdb FILE=<BRDB Export>.dmp LOG=<BRDB Export>.log ROWS=N BUFFER=10485760 COMPRESS=N GRANTS=N STATISTICS=N</pre> <pre>exp system@brss OWNER=ops\\$brdb FILE=<BRSS Export>.dmp LOG=<BRSS Export>.log ROWS=N BUFFER=10485760 COMPRESS=N GRANTS=N STATISTICS=N</pre> <p>Create two users, e.g. BDB_COMP and BRS_COMP, on some other temporary database where it is safe to perform an investigation. Then import the OPS\$BRDB schema exports into the new schemas:</p> <pre>imp system@brdb FILE=<BRDB Export>.dmp LOG=<BRDB COMP Import>.log FROMUSER=OPS\\$BRDB TOUSER=BDB_COMP BUFFER=10485760 IGNORE=Y</pre> <pre>imp system@brss FILE=<BRSS Export>.dmp LOG=<BRSS COMP Import>.log FROMUSER=OPS\\$BRDB TOUSER=BRS_COMP BUFFER=10485760 IGNORE=Y</pre> <p>Now perform an investigation/comparison of both schemas. Produce a script to run, in order to synchronise OPS\$BRDB on BRSS with that of BRDB.</p>
B2.	<p><u>BDB and BRS:</u></p> <p>Again, <i>with the assumption that historical data and partitions are not required,</i> complete Steps [5a.] to [5e.]</p>	Complete Steps [5a.], [5b.], [5c.], [5d.] and [5e.]



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



<p>B3.</p> <p><u>BRS:</u></p> <p>The schemas should structurally now be the same.</p> <p><u>WARNING!!!</u> DATA LOSS WILL OCCUR DURING THIS STEP.</p> <p>If the current failure of Streams was caused by the schedules not being run and you're sure that cleaning out partition and schedule related data won't be a problem, then continue...</p> <p>Clean out all metadata related to the partitions in OPS\$BRDB, stored in the OPS\$BRSS schema, and correctly populate the metadata, including setting the business date to SYSDATE.</p> <p>The relevant metadata tables are shown below. Check them to see how the scripts affect the metadata.</p>	<p>su - brss</p> <p>From /app_sw/brss/build/schema/ ... run ...</p> <p>pt_cleanup.sh BRSS</p> <p>Query the tables below ...</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



	<p>These tables include amongst others: -</p> <p>BRSS_OPERATIONAL_EXCEPTIONS: This shows the exceptions that may have occurred.</p> <p>BRSS_PARTITION_CREATES: Shows partitions created and when. Order by table_name, partition_range_value.</p> <p>BRSS_PARTITION_STATUS_HISTORY: Shows the history of partition maintenance. Order by table_name, partition_name.</p> <p>BRSS_SUBPARTITION_RANGES: Shows the range upper bound required at next run. Relevant column is range_value.</p>
B4.	<p><u>BRS:</u></p> <p>Now, invoke the Start of Day process to create the required partitions for "today" and "tomorrow", correctly updating all partition metadata.</p> <p>Check the metadata tables again, ensuring the metadata is as you expect it to be.</p> <p>su - brss</p> <p>From /app_sw/brss/build/schema/ ... run ...</p> <p>ptcatchup.sh BRSS</p>
B5.	<p><u>BDB:</u></p> <p><u>WARNING!!!</u> DATA LOSS WILL OCCUR DURING THIS STEP.</p> <p>If and only if you're sure that there is a major problem with the schedules or there is no other way around the cleanup of partition metadata, then do the same to BDB as in [B3.].</p> <p>!!! THIS STEP SHOULD NEVER BE RUN IN LIVE !!!</p> <p>su - brdb</p> <p>From /app_sw/brdb/build/schema/ ... run ...</p> <p>pt_cleanup.sh BRDB</p>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



B6.	<p><u>BDB:</u></p> <p>Do the same for BDB as in [B4.].</p>	<p>!!! THIS STEP SHOULD NEVER BE RUN IN LIVE !!!</p> <p>su - brdb</p> <p>From /app_sw/brdb/build/schema/ ... run ...</p> <p>ptcatchup.sh BRDB</p>
B7.	<p><u>BDB:</u></p> <p>Complete steps [6.] and [7.].</p>	<p>Complete Steps [6.] and [7.]</p>
	<p>All done.</p> <p>Note: There may be errors detected on the application of Streams data on the "apply" queue [DBA_APPLY_ERRORS], due to the mismatch of data and partitions. A very good example is the fact that a DELETE may occur on BDB, but the DML fails because the associated row does not exist on BRS. These are acceptable errors.</p> <p>However, similar errors may occur due to a mismatch of partitions. This is not acceptable and must be rectified.</p> <p>See Section 5.5.4.4 for additional information on working with and resolving these errors.</p>	

Table 8: BRSS Support Procedure



7.2 Managing Streams Lag

7.2.1 Context and Assumptions

Oracle Streams 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. Oracle Streams, in most cases, manages the capture-propagate-apply flow of changes fairly well, provided the flow of changes is applied in good time.

Oracle Streams attempts to manage the flow of changes by putting the capture process into a state of "flow control" or by getting the apply process to "spill" changes to disk and in most cases this ends up to be a combination of both.

Flow control, simply means that the capture process stops capturing and put into a "paused" state until the message which caused the pause is either applied or is spilled.

Problems with the rate that the changes are applied comes to bear if transaction sizes are: -

- i. Too large, i.e. greater than 500,000 records.
- ii. Last too long, i.e. longer than 5 minutes.

In these cases, the apply process will spill the LCRs (Logical Change Records) to disk, and will continue to do so for that entire transaction, however long that might take. In addition the apply server (an apply sub-process) that begins the spill and apply will have to complete it before any of the other processes may continue – at great cost.

The assumptions for the rest of this section are therefore that the Streams process has had to either spill transactions or is in a continual state of "PAUSED FOR FLOW CONTROL", or both. In addition the apply process has reached a period of unacceptable lag (see Section 7.2.2) and a decision has been made to re-instantiate Streams.

This re-instantiation is assumed to save all data and the only part of the solution being recreated is Streams and its constituent components.

7.2.2 Lag Evaluation and Escalation

There are a number of factors to consider when deciding whether or not to re-instantiate Streams. The escalation procedures will need to be followed in the first instance and a decision can be made, considering the facts and reasons for the lag.

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.
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? 4 th -Line Support notified of the cause and progress.



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



16 – 20 hrs.	DBA Support notified. Are the original problems reoccurring?
24 hrs.	<p>DBA Support notified. 4th-Line Support notified. Appropriate business owner notified.</p> <p>At this point, there are <i>x</i> number of days (currently 4) which remain in which to continue the investigation or to put in place a fix and prepare for Streams re-instantiation, should the decision be made to do so.</p> <p>Note that <i>x</i> is defined as the lowest number of days for data retention of any table on BRDB. The following query shows the result:</p> <pre>SELECT MIN(retention_period) FROM brdb_archived_tables WHERE retention_period <> 0 AND additional_criteria IS NULL;</pre>
48 hrs.	Re-evaluate the situation and prepare for re-instantiation providing all the approvals have been received.

7.2.3 Lag Assessment and Action Procedure

Step	Description	Server Execution
Assumptions		
User is logged onto the respective Database Server(s) as <i>oracle</i> and into the database as SYSDBA.		
1.	<p>BDB</p> <p>After logging on to the database as SYSDBA.</p> <p>Use Section 5.5.4.1 to check the state of the <i>capture process</i>.</p> <p>For a concise capture summary use Query [i.] of Section 5.5.4.5. For capture errors use Query [iii.] of the same section.</p> <p>1. For capture latency with mining the logs then run [A.] below.</p> <p>For capture latency with enqueueing messages then run [B.] below.</p> <p>For capture memory usage then run [C.] below.</p> <p>For long running transactions (greater than 10 mins), run [D.] below.</p> <p>Ignore the capture of a particular transaction by following Step 4.</p>	<p>[A] Likely to change with every transaction ...</p> <pre>COLUMN CAPTURE_NAME HEADING 'Capture Process Name' FORMAT A12 COLUMN LATENCY_SECONDS HEADING 'Latency in Seconds' FORMAT 999999 COLUMN LAST_STATUS HEADING 'Seconds Since Last Status' FORMAT 999999 COLUMN CAPTURE_TIME HEADING 'Current Process Time' COLUMN CREATE_TIME HEADING 'Message Creation Time' FORMAT 999999 SELECT</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

CAPTURE_NAME,
((SYSDATE - CAPTURE_MESSAGE_CREATE_TIME)*86400) LATENCY_SECONDS,
((SYSDATE - CAPTURE_TIME)*86400) LAST_STATUS,
TO_CHAR(CAPTURE_TIME, 'HH24:MI:SS MM/DD/YY') CAPTURE_TIME,
TO_CHAR(CAPTURE_MESSAGE_CREATE_TIME, 'HH24:MI:SS MM/DD/YY') CREATE_TIME
FROM V$STREAMS_CAPTURE;

[B]
COLUMN CAPTURE_NAME HEADING 'Capture|Process|Name' FORMAT A12
COLUMN LATENCY_SECONDS HEADING 'Latency|in|Seconds' FORMAT 999999
COLUMN CREATE_TIME HEADING 'Message Creation|Time' FORMAT A20
COLUMN ENQUEUE_TIME HEADING 'Enqueue Time' FORMAT A20
COLUMN ENQUEUE_MESSAGE_NUMBER HEADING 'Message|Number' FORMAT 999999999999
SELECT
    CAPTURE_NAME,
    (ENQUEUE_TIME-ENQUEUE_MESSAGE_CREATE_TIME)*86400 LATENCY_SECONDS,
    TO_CHAR(ENQUEUE_MESSAGE_CREATE_TIME, 'HH24:MI:SS MM/DD/YY') CREATE_TIME,
    TO_CHAR(ENQUEUE_TIME, 'HH24:MI:SS MM/DD/YY') ENQUEUE_TIME,
    ENQUEUE_MESSAGE_NUMBER
FROM V$STREAMS_CAPTURE;

[C]
set pages 100
col "capture Process" for a25
col "capture Name" for a25
SELECT
    p.spid Spid,
    'C00'||c.capture#||' '||UPPER(lp.ROLE) "Capture Process",
    c.capture_name "Capture Name",
    p.pga_used_mem "PGA Memory Used",
    p.pga_alloc_mem "PGA Memory Allocated",
    p.pga_max_mem "PGA Maximum Memory"
FROM v$streams_capture c, v$logmnr_process lp, v$session s, v$process P
WHERE c.logminer_id = lp.session_id
    AND lp.ROLE IN ('reader', 'preparer', 'builder')
    AND lp.SID = s.SID
    AND lp.serial# = s.serial#
    AND s.paddr = P.addr
UNION
SELECT p.spid,
    'C00'||c.capture#||' Coordinator',
    c.capture_name,
    p.pga_used_mem,
    p.pga_alloc_mem,
    p.pga_max_mem
FROM v$streams_capture c, v$session s, v$process P
WHERE c.SID = s.SID
    AND c.serial# = s.serial#
    AND s.paddr = P.addr
ORDER BY 6,5;

[D]
col runlength HEAD 'Txns Open|Minutes' format 9999.99
col sid HEAD 'Session' format a13
col xid HEAD 'Transaction|ID' format a18
col terminal HEAD 'Terminal' format a10
col program HEAD 'Program' format a27 wrap
SELECT t.inst_id,
    sid || ',' || serial# sid,
    xidusn || '.' || xidslot || '.' || xidsqn xid,
    (SYSDATE - start_date) * 1440 runlength,
    terminal,
    program

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



	<pre>FROM gv\$transaction t, gv\$session s WHERE t.addr = s.taddr AND (SYSDATE - start_date) * 1440 > 10;</pre>
2.	<p><u>BDB</u></p> <p>Use Section 5.5.4.2 to check the state of the <i>propagation process</i>. For latency in propagation of new messages then run [E.] below.</p> <p>[E]</p> <pre>PROMPT Maximum Wait Time to Propagate a New Message Latency COLUMN START_DATE HEADING 'Start Date' COLUMN PROPAGATION_WINDOW HEADING 'Duration in Seconds' FORMAT 99999 COLUMN NEXT_TIME HEADING 'Next Time' FORMAT A8 COLUMN LATENCY HEADING 'Latency in Seconds' FORMAT 99999 COLUMN SCHEDULE_DISABLED HEADING 'Status' FORMAT A8 COLUMN PROCESS_NAME HEADING 'Process' FORMAT A8 COLUMN FAILURES HEADING 'Number of Failures' FORMAT 99 SELECT DISTINCT TO_CHAR(s.START_DATE, 'HH24:MI:SS MM/DD/YY') START_DATE, s.PROPAGATION_WINDOW, s.NEXT_TIME, s.LATENCY, DECODE(s.SCHEDULE_DISABLED, 'Y', 'Disabled', 'N', 'Enabled') SCHEDULE_DISABLED, s.PROCESS_NAME, s.FAILURES FROM DBA_QUEUE_SCHEDULES s, DBA_PROPAGATION p WHERE p.PROPAGATION_NAME = 'BRDB_PROP' AND p.DESTINATION_DBLINK = s.DESTINATION AND s.SCHEMA = p.SOURCE_QUEUE_OWNER AND s.QNAME = p.SOURCE_QUEUE_NAME;</pre>
3.	<p><u>BRS</u></p> <p>Use Section 5.5.4.3 to check the state of the <i>apply process</i>. For Apply dequeuing info use Query [iv.] of Section 5.5.4.5. For tracking the applying of transactions use Query [v.] of Section 5.5.4.5. And Use [vi.] for more detail of a particular transaction. For latency from capture to dequeue at the apply, then run [F.] below. For latency from capture to actual message apply, then run [G.] below. For a better idea of the types and number of apply errors occurring, run [H.] below.</p> <p>[F]</p> <pre>COLUMN APPLY_NAME HEADING 'Apply Process Name' FORMAT A17 COLUMN LATENCY HEADING 'Latency in Seconds' FORMAT 9999 COLUMN CREATION HEADING 'Message Creation' FORMAT A17 COLUMN LAST_DEQUEUE HEADING 'Last Dequeue Time' FORMAT A20 COLUMN DEQUEUED_MESSAGE_NUMBER HEADING 'Dequeued Message Number' FORMAT 99999999999 SELECT APPLY_NAME, (DEQUEUE_TIME-DEQUEUED_MESSAGE_CREATE_TIME)*86400 LATENCY, TO_CHAR(DEQUEUED_MESSAGE_CREATE_TIME, 'HH24:MI:SS MM/DD/YY') CREATION, TO_CHAR(DEQUEUE_TIME, 'HH24:MI:SS MM/DD/YY') LAST_DEQUEUE, DEQUEUED_MESSAGE_NUMBER FROM V\$STREAMS_APPLY_READER;</pre>



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

[G]
COLUMN APPLY_NAME HEADING 'Apply Process|Name' FORMAT A17
COLUMN 'Latency in Seconds' FORMAT 999999
COLUMN 'Message Creation' FORMAT A17
COLUMN 'Apply Time' FORMAT A17
COLUMN APPLIED_MESSAGE_NUMBER HEADING 'Applied|Message|Number' FORMAT 999999
SELECT
    apply_name,
    (apply_time - applied_message_create_time)*86400 "Latency in Seconds",
    TO_CHAR(applied_message_create_time,'HH24:MI:SS MM/DD/YY') "Message Creation",
    TO_CHAR(apply_time,'HH24:MI:SS MM/DD/YY') "Apply Time",
    applied_message_number
FROM dba_apply_progress;

[H]
set pages 45
SELECT TO_CHAR(error_creation_time, 'YYYY/MM/DD') created,
       error_number,
       COUNT(1) error_count
  FROM dba_error
 GROUP BY error_number,
          TO_CHAR(error_creation_time, 'YYYY/MM/DD')
 ORDER BY 1, 2;

```

4.	<p>BDB</p> <p>Once a transaction has been identified as problematic and can safely be assumed to not be a problem if ignored, then the following steps will help in allowing the capture process to ignore that transaction. This is providing of course that the transaction has not already begun to be mined by the capture process.</p> <p>Identifying the problematic/long running transaction can be done by running query [D.] of step 1 of this procedure.</p> <p>Then run the steps which follow as STRADMIN. Note that <TXN_ID> below refers to the actual transaction id of the transaction you wish to ignore.</p> <pre> EXECUTE dbms_capture_adm.stop_capture('BRDB_CAPTURE'); EXECUTE dbms_capture_adm.set_parameter('BRDB_CAPTURE', '_ignore_transaction', '<TXN_ID>'); EXECUTE dbms_capture_adm.start_capture('BRDB_CAPTURE'); </pre>
----	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



7.2.4 Post Lag Action Procedure

Step	Description	Server Execution
Assumptions		
<ul style="list-style-type: none"> - User is logged onto the respective Database Server(s) as <i>oracle</i> and where possible as SYSDBA - The TWS schedules can continue to run and won't need to be stopped during this procedure. - "Partition Managed Tables" simply means those tables managed by BR/DB SS/C001. 		
Applicable Scenario		
<p>The Branch Support Database has been determined to have unacceptable levels of lag and a decision has been made to re-instantiate Streams.</p> <p>The scenario is as follows</p> <ul style="list-style-type: none"> - The decision to re-instantiate Streams was made "TODAY" - Streams has been lagging for less than 24 hrs. - The data for BRSS partitions of date "TODAY" are lagging. - The data for BRSS partitions of date "YESTERDAY" have replicated without issue. <p>This procedure should therefore be executed approximately 30 – 15 minutes before midnight "TODAY" (before "TOMORROW") and Step 1a MUST be complete before 00:00 (i.e. before "TOMORROW").</p>		
<p>NOTE</p> <p>Throughout this procedure, "YESTERDAY", "TODAY" and "TOMORROW" will remain specific dates in order to avoid confusion, e.g. 28th, 29th and 30th even though this procedure, in reality, will be used before and after midnight and therefore create confusion..</p>		
1a.	<p>BDB and BRS</p> <p>Re-instantiate Streams by following the methodology presented in Section 7.1.2, executing Steps 5a. – 7</p> <p>This will recreate the Streams Configuration and will not affect the data that has already been mined.</p>	
1b.	Following from 1a. above, Steps 8 – 11 of Section 7.1.2, must follow as soon as possible!	
2.	<p>BDB</p> <p>Determine whether the partitions created on BRDB are the same as those on BRSS. This should always be the case, but double-check.</p> <p>There should be <u>no rows returned</u> ... (i.e. the output to the first sub-query should be the same as the second).</p> <p>Please run just these queries as <u>STRADMIN</u>: –</p> <pre>SELECT dtp.table_name, MAX(dtp.partition_name) max_part_name FROM dba_tab_partitions dtp, brdb_partition_creates bpc, brdb_partitioned_tables brt WHERE dtp.table_name = bpc.table_name (+) AND dtp.table_name = brt.table_name (+) AND dtp.table_owner = 'OPS\$BRDB' AND bpc.status = 'NEW' AND dtp.partition_name = brt.partition_root_name '_' bpc.partition_range_value</pre>	



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



	<pre> GROUP BY dtp.table_name MINUS SELECT sdtp.table_name, MAX(sdtp.partition_name) max_part_name FROM dba_tab_partitions@brss sdtp, ops\$brss.brss_partitionCreates@brss sbpc, ops\$brss.brss_partitioned_tables@brss sbrt WHERE sdtp.table_name = sbpc.table_name (+) AND sdtp.table_name = sbrt.table_name (+) AND sdtp.table_owner = 'OPS\$BRDB' AND sbpc.status = 'NEW' AND sdtp.partition_name = sbrt.partition_root_name '_' sbpc.partition_range_value GROUP BY sdtp.table_name </pre>
3.	<p>BDB</p> <ol style="list-style-type: none"> 1. Determine the list of partitions to export from BRDB (for a later import into BRSS) 2. Perform the export(s) – ensure you have the password for SYSTEM 3. Transport the export dump files to <i>/prpbtrs001</i> <p>Find a directory that has enough space to store the export dump files. The create a directory object as follows (SQL*Plus): -</p> <pre>CREATE OR REPLACE DIRECTORY <dump_directory_name> AS '/<dump_directory>';</pre> <p>Create a parfile (vi <i><parfile_name>.par</i>) with the output from the following query (SQL*Plus): -</p> <pre> set pages 0 set trimspool ON set feedback off set echo off SELECT DECODE(ROWNUM,1,'CONTENT=ALL' TABLES=' CHR(39) dtp.table_owner '.' dtp.table_name '.' dtp.partition_name CHR(39) ,18,' CHR(39) dtp.table_owner '.' dtp.table_name '.' dtp.partition_name CHR(39) ' ,' CHR(39) dtp.table_owner '.' dtp.table_name '.' dtp.partition_name CHR(39)) FROM dba_tab_partitions dtp, brdb_partitionCreates bpc, brdb_partitionedTables brt WHERE dtp.table_name = bpc.table_name (+) AND dtp.table_name = brt.table_name (+) AND dtp.table_owner = 'OPS\$BRDB' AND bpc.partition_range_value = TO_CHAR(TRUNC(SYSDATE), 'YYYYMMDD') AND dtp.partition_name = brt.partition_root_name '_' bpc.partition_range_value; </pre> <p>Export the tables by using the following datapump command, substituting the required values: -</p> <pre>expdp system DIRECTORY=<dump_directory_name> PARFILE=<parfile_name>.par DUMPFILE=<dumpfile_name>.dmp LOGFILE=<logfile_name>.log</pre> <p>Remote copy or FTP the <i><dumpfile_name>.dmp</i> dump file to the BRS server. The quickest way to accomplish this may be to create an NFS or NAS mount between the servers.</p>
4.	<p>BRS – Partition Managed Tables ONLY</p> <ol style="list-style-type: none"> 1. Truncate the partitions for “TODAY” only, using the SQL provided. 2. Import the partitions exported from BRDB (see Step 3 above) for “TODAY” only.

HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE

	SQL to be defined in later version impdp command to be defined in later version
5.	<u>BDB – Non-partition Managed Tables ONLY</u> 1. Merge the data from the Branch Database into those of the Branch Support Database. INSERT INTO x@BRSS SELECT * FROM x WHERE y BETWEEN YESTERDAY AND TOMORROW; ...
6.	<u>BDB</u> 1. Run a comparison between the two database schemas. SELECT table_name, partition_name, num_rows, ... FROM dba_tab_partitions WHERE ...



8 Appendix C – Transaction Correction Templates

Section 5.6 describes the use of the transaction correction tool BRDBX015.sh. This is used by SSC to correct transactions by inserting balancing records to transactional/accounting/stock tables in the BRDB system. The tool must be supplied with a file containing a SQL statement that performs the required insert. This statement must be of a particular form, and should be based on one of the templates shown here.

Separate templates are given for each given target table, which reflect the columns of the target table.

8.1 Template for BRDB_RX_APS_TRANSACTIONS

```
/*
The following SQL is a template for the SQL to be placed in a transaction file
for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.
```

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
-----	-----	-----

```
*/
```

```
INSERT INTO ops$brdb.brdb_rx_aps_transactions
( journal_date,
  fad_hash,
  branch_accounting_code,
  branch_code,
  journal_seq_number,
  node_id,
  session_id,
  transaction_id,
  insert_timestamp,
  trading_date,
  transaction_start_date,
  branch_user,
  amount,
  ap_record_type,
  balance_period,
  client_account_code,
  client_code,
  client_service_code,
  entry_method,
  quantity,
  receipt_line_id,
  reversal,
  prod_id,
  trading_period,
  transaction_mode_id,
  transaction_mode_version_no,
  token_id,
  token_type_version_no,
  payment_code,
  transaction_end_date,
  fallback_mode,
  additional_data,
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

ap_failure_code,
customer_reference,
stock_unit,
service_group,
transferred_from_legacy_yn,
reversal_session_id,
reversal_transaction_id,
reversal_node_id )
SELECT
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED *****
    A.fad_hash, -- fad_hash ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
    A.branch_accounting_code, -- branch_accounting_code ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
    A.branch_accounting_code, -- branch_code ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    A.current_json + 1, -- journal_seq_number ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    99, -- node_id ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
    nn, -- session_id ***** TO BE SPECIFIED *****
    nn, -- transaction_id ***** TO BE SPECIFIED *****
    SYSTIMESTAMP, -- insert_timestamp ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_start_date ***** TO BE SPECIFIED *****
    ':bind_SSC_user', -- branch_user ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    nnnnn, -- amount ***** TO BE SPECIFIED *****
    NULL, -- ap_record_type
    NULL, -- balance_period
    NULL, -- client_account_code
    NULL, -- client_code
    NULL, -- client_service_code
    NULL, -- entry_method
    nn, -- quantity ***** TO BE SPECIFIED *****
    NULL, -- receipt_line_id
    NULL, -- reversal
    NULL, -- prod_id
    NULL, -- trading_period
    NULL, -- transaction_mode_id
    NULL, -- transaction_mode_version_no
    NULL, -- token_id
    NULL, -- token_type_version_no
    n, -- payment_code ***** TO BE SPECIFIED *****
    NULL, -- transaction_end_date
    'x', -- fallback_mode ***** TO BE SPECIFIED *****
    NULL, -- additional_data
    NULL, -- ap_failure_code
    NULL, -- customer_reference
    NULL, -- stock_unit
    NULL, -- service_group
    'x', -- transferred_from_legacy_yn ***** TO BE SPECIFIED *****
    NULL, -- reversal_session_id
    NULL, -- reversal_transaction_id
    NULL -- reversal_node_id
FROM dual,
    (SELECT fhom.branch_accounting_code,
        fhom.fad_hash,
        tctc.current_json

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

FROM  ops$brdb.brdb_fad_hash_outlet_mapping fhom,
      ops$brdb.brdb_txn_corr_tool_ctl tctc
WHERE fhom.branch_accounting_code = :bind_branch_code
AND   tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.2 Template for BRDB_RX_BUREAU_TRANSACTIONS

/*
The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
-----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_bureau_transactions
( journal_date,
  fad_hash,
  branch_accounting_code,
  amount,
  branch_code,
  entry_method,
  journal_seq_number,
  node_id,
  quantity,
  reversal,
  prod_id,
  session_id,
  transaction_mode_id,
  transaction_id,
  insert_timestamp,
  trading_date,
  transaction_end_date,
  transaction_start_date,
  fallback_mode,
  application_type,
  branch_user,
  stock_unit,
  balance_period,
  commission_rate,
  bureau_region,
  effective_exchange_rate,
  exchange_rate,
  margin_rate,
  margin_product,
  receipt_line_id,
  reversal_session_id,
  reversal_transaction_id,
  transaction_mode_version_no,
  purchased_value,
  stock_quantity,
  trading_period,
  bureau_currency,
  reversal_stock_unit,
  pouch_id,
  transferred_from_legacy_y_n )

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

SELECT
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED *****
    A.fad_hash, -- fad_hash      ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
    A.branch_accounting_code, -- branch_accounting_code      ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
    nnnn, -- amount      ***** TO BE SPECIFIED *****
    A.branch_accounting_code, -- branch_code      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    n, -- entry_method      ***** TO BE SPECIFIED *****
    A.current_jsn + 1, -- journal_seq_number      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    99, -- node_id      ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
    nnnn, -- quantity      ***** TO BE SPECIFIED *****
    n, -- reversal      ***** TO BE SPECIFIED *****
    nnnn, -- prod_id      ***** TO BE SPECIFIED *****
    nnnn, -- session_id      ***** TO BE SPECIFIED *****
    n, -- transaction_mode_id      ***** TO BE SPECIFIED *****
    nnn, -- transaction_id      ***** TO BE SPECIFIED *****
    SYSTIMESTAMP, -- insert_timestamp      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_end_date      ***** TO BE SPECIFIED *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_start_date      ***** TO BE SPECIFIED *****
    'x', -- fallback_mode      ***** TO BE SPECIFIED *****
    'x', -- application_type      ***** TO BE SPECIFIED *****
    ':bind_SSC_user', -- branch_user      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    'x', -- stock_unit      ***** TO BE SPECIFIED *****
    NULL, -- balance_period
    NULL, -- commission_rate
    NULL, -- bureau_region
    NULL, -- effective_exchange_rate
    NULL, -- exchange_rate
    NULL, -- margin_rate
    NULL, -- margin_product
    NULL, -- receipt_line_id
    NULL, -- reversal_session_id
    NULL, -- reversal_transaction_id
    NULL, -- transaction_mode_version_no
    NULL, -- purchased_value
    NULL, -- stock_quantity
    NULL, -- trading_period
    NULL, -- bureau_currency
    NULL, -- reversal_stock_unit
    NULL, -- pouch_id
    'x' -- transferred_from_legacy_yn      ***** TO BE SPECIFIED *****
FROM dual,
    (SELECT fhom.branch_accounting_code,
        fhom.fad_hash,
        tctc.current_jsn
    FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
        ops$brdb.brdb_txn_corr_tool_ctl tctc
    WHERE fhom.branch_accounting_code = :bind_branch_code
    AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.3 Template for BRDB_RX_CUT_OFF_SUMMARIES



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



/*

The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
-----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_cut_off_summaries
  (fad_hash,
   branch_accounting_code,
   branch_code,
   journal_seq_number,
   insert_timestamp,
   trading_date,
   branch_user,
   amount,
   node_id,
   quantity,
   pouch_id,
   prod_id,
   transaction_mode_id,
   week_number,
   transaction_end_date,
   stock_unit )
SELECT
  A.fad_hash, -- fad_hash      ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_accounting_code      ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_code      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
  A.current_jsn + 1, -- journal_seq_number      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
  SYSTIMESTAMP, -- insert_timestamp      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
  ':bind_SSC_user', -- branch_user      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
  nnnn, -- amount      ***** TO BE SPECIFIED *****
  99, -- node_id      ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
  nn, -- quantity      ***** TO BE SPECIFIED *****
  NULL, -- pouch_id      ***** TO BE SPECIFIED *****
  nnnn, -- prod_id      ***** TO BE SPECIFIED *****
  nn, -- transaction_mode_id      ***** TO BE SPECIFIED *****
  NULL, -- week_number
  NULL, -- transaction_end_date
  'x' -- stock_unit      ***** TO BE SPECIFIED *****
FROM dual,
  (SELECT fhom.branch_accounting_code,
   fhom.fad_hash,
   tctc.current_jsn
  FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
   ops$brdb.brdb_txn_corr_tool_ctl tctc

```



```
WHERE  fhom.branch_accounting_code = :bind_branch_code
AND    tctc.branch_accounting_code = fhom.branch_accounting_code) A;
```

8.4 Template for BRDB_RX_DCS_TRANSACTIONS

/*

The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
-----	-----	-----

*/

```
INSERT INTO ops$brdb.brdb_rx_dcs_transactions
( journal_date,
  fad_hash,
  amount,
  branch_accounting_code,
  branch_code,
  entry_method,
  journal_seq_number,
  node_id,
  quantity,
  prod_id,
  session_id,
  transaction_mode_id,
  transaction_id,
  insert_timestamp,
  trading_date,
  transaction_end_date,
  transaction_start_date,
  fallback_mode,
  branch_user,
  agent_sla_info,
  amount_requested,
  balance_period,
  bank_client,
  card_impounded,
  counter_sla_info,
  fee,
  merchant_number,
  receipt_line_id,
  trading_period,
  transaction_mode_version_no,
  financial_transaction,
  scheme_id,
  recovery_flag,
  response_code,
  routing_gateway,
  transaction_result_code,
  transaction_type,
  receipt_date,
  settlement_date,
  authorisation_code,
  bank_transaction_id,
  currency,
```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

c12_xml,
horizon_transaction_id,
pan,
stock_unit,
transferred_from_legacy_yn,
reversal,
reversal_session_id,
reversal_transaction_id,
reversal_stock_unit )
SELECT
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED *****
    A.fad_hash, -- fad_hash      ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
    nnnn, -- amount      ***** TO BE SPECIFIED *****
    A.branch_accounting_code, -- branch_accounting_code      ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
    A.branch_accounting_code, -- branch_code      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    n, -- entry_method      ***** TO BE SPECIFIED *****
    A.current_jsn + 1, -- journal_seq_number      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    99, -- node_id      ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
    nnnn, -- quantity      ***** TO BE SPECIFIED *****
    nnnn, -- prod_id      ***** TO BE SPECIFIED *****
    nnnn, -- session_id      ***** TO BE SPECIFIED *****
    n, -- transaction_mode_id      ***** TO BE SPECIFIED *****
    nnnn, -- transaction_id      ***** TO BE SPECIFIED *****
    SYSTIMESTAMP, -- insert_timestamp      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_end_date      ***** TO BE SPECIFIED *****
    TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_start_date      ***** TO BE SPECIFIED *****
    'x', -- fallback_mode      ***** TO BE SPECIFIED *****
    ':bind SSC_user', -- branch_user      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    NULL, -- agent_sla_info
    NULL, -- amount_requested
    NULL, -- balance_period
    NULL, -- bank_client
    n, -- card_impounded      ***** TO BE SPECIFIED *****
    NULL, -- counter_sla_info
    n, -- fee      ***** TO BE SPECIFIED *****
    NULL, -- merchant_number
    NULL, -- receipt_line_id
    NULL, -- trading_period
    NULL, -- transaction_mode_version_no
    n, -- financial_transaction      ***** TO BE SPECIFIED *****
    NULL, -- scheme_id
    NULL, -- recovery_flag
    NULL, -- response_code
    NULL, -- routing_gateway
    NULL, -- transaction_result_code
    NULL, -- transaction_type
    NULL, -- receipt_date
    NULL, -- settlement_date
    NULL, -- authorisation_code
    NULL, -- bank_transaction_id
    NULL, -- currency

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

NULL,    -- c12_xml
NULL,    -- horizon_transaction_id
NULL,    -- pan
NULL,    -- stock_unit
'x',    -- transferred_from_legacy_yn      ***** TO BE SPECIFIED *****
NULL,    -- reversal
NULL,    -- reversal_session_id
NULL,    -- reversal_transaction_id
NULL    -- reversal_stock_unit
FROM dual,
  (SELECT fhom.branch_accounting_code,
         fhom.fad_hash,
         tctc.current_jsn
    FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
         ops$brdb.brdb_txn_corr_tool_ctl tctc
   WHERE fhom.branch_accounting_code = :bind_branch_code
     AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.5 Template for BRDB_RX_EPOSS_EVENTS

/*
The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
-----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_eposs_events
( event_date,
  fad_hash,
  branch_accounting_code,
  branch_code,
  event_id,
  node_id,
  journal_seq_number,
  insert_timestamp,
  journal_date,
  trading_date,
  branch_user,
  adjustment_amount,
  balance_period,
  trading_period,
  tc_count,
  changed_branch_user,
  changed_stock_unit,
  stock_unit,
  stock_unit_type,
  transferred_from_legacy_yn )
SELECT
  TO_DATE ('nn/nm/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- event_date
***** TO BE SPECIFIED *****
  A.fad_hash, -- fad_hash      ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_accounting_code      ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****

```

© Copyright Fujitsu Ltd 2009 COMMERCIAL IN CONFIDENCE Ref: DES/APP/SPG/0001

Version: 1.0
Date: 29-Oct-09
Page No: 149 of 159



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

A.branch_accounting_code, -- branch_code      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! ****
      nnnn, -- event_id      ***** TO BE SPECIFIED ****
      99, -- node_id      ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! ****
      A.current_jsn + 1, -- journal_seq_number      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! ****
      SYSTIMESTAMP, -- insert_timestamp      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! ****
      TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED ****
      TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED ****
      ':bind SSC_user', -- branch_user      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! ****
      NULL, -- adjustment_amount
      NULL, -- balance_period
      NULL, -- trading_period
      NULL, -- tc_count
      NULL, -- changed_branch_user
      NULL, -- changed_stock_unit
      NULL, -- stock_unit
      'x', -- stock_unit_type      ***** TO BE SPECIFIED ****
      'x' -- transferred_from_legacy_yn      ***** TO BE SPECIFIED ****
FROM dual,
      (SELECT fhom.branch_accounting_code,
      fhom.fad_hash,
      tctc.current_jsn
      FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
      ops$brdb.brdb_txn_corr_tool_ctl tctc
      WHERE fhom.branch_accounting_code = :bind_branch_code
      AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.6 Template for BRDB_RX_EPOSS_TRANSACTIONS

/*

The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_eboss_transactions
( journal_date,
  fad_hash,
  branch_accounting_code,
  branch_code,
  amount,
  entry_method,
  journal_seq_number,
  node_id,
  prod_id,
  quantity,
  reversal,
  session_id,

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

transaction_id,
transaction_mode_id,
insert_timestamp,
trading_date,
transaction_end_date,
transaction_start_date,
fallback_mode,
branch_user,
stock_unit,
balance_period,
pouch_id,
receipt_line_id,
reversal_session_id,
reversal_transaction_id,
trading_period,
transaction_mode_version_no,
client_reference,
reversal_stock_unit,
tc_reference,
transferred_from_legacy_yn )
SELECT
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED *****
  A.fad_hash, -- fad_hash ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_accounting_code ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_code ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
  nnnnnn, -- amount ***** TO BE SPECIFIED *****
  n, -- entry_method ***** TO BE SPECIFIED *****
  A.current_jsn + 1, -- journal_seq_number ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
  99, -- node_id ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
  nnnn, -- prod_id ***** TO BE SPECIFIED *****
  nnnn, -- quantity ***** TO BE SPECIFIED *****
  n, -- reversal ***** TO BE SPECIFIED *****
  nnnn, -- session_id ***** TO BE SPECIFIED *****
  nnnn, -- transaction_id ***** TO BE SPECIFIED *****
  n, -- transaction_mode_id ***** TO BE SPECIFIED *****
  SYSTIMESTAMP, -- insert_timestamp ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_end_date ***** TO BE SPECIFIED *****
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_start_date ***** TO BE SPECIFIED *****
  'x', -- fallback_mode ***** TO BE SPECIFIED *****
  ':bind_SSC_user', -- branch_user ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
  'x', -- stock_unit ***** TO BE SPECIFIED *****
  NULL, -- balance_period
  NULL, -- pouch_id
  NULL, -- receipt_line_id
  NULL, -- reversal_session_id
  NULL, -- reversal_transaction_id
  NULL, -- trading_period
  NULL, -- transaction_mode_version_no
  NULL, -- client_reference
  NULL, -- reversal_stock_unit
  NULL, -- tc_reference

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

'x'    -- transferred_from_legacy_yn      ***** TO BE SPECIFIED *****
FROM dual,
  (SELECT fhom.branch_accounting_code,
    fhom.fad_hash,
    tctc.current_jsn
  FROM  ops$brdb.brdb_fad_hash_outlet_mapping fhom,
    ops$brdb.brdb_txn_corr_tool_ctl tctc
  WHERE fhom.branch_accounting_code = :bind_branch_code
  AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.7 Template for BRDB_RX_NWB_TRANSACTIONS

/*

The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_nwb_transactions
( journal_date,
  fad_hash,
  branch_accounting_code,
  branch_code,
  amount,
  entry_method,
  journal_seq_number,
  node_id,
  prod_id,
  quantity,
  session_id,
  transaction_mode_id,
  transaction_id,
  insert_timestamp,
  trading_date,
  transaction_end_date,
  transaction_start_date,
  fallback_mode,
  branch_user,
  stock_unit,
  agent_sla_info,
  amount_requested,
  bank_client,
  balance_period,
  card_impounded,
  counter_sla_info,
  currency,
  fee,
  financial_transaction,
  receipt_line_id,
  recovery_flag,
  response_code,
  routing_gateway,
  scheme_id,
  trading_period,

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

transaction_mode_version_no,
transaction_result_code,
transaction_type,
receipt_date,
settlement_date,
authorisation_code,
bank_transaction_id,
c12_xml,
pan,
horizon_transaction_id,
transferred_from_legacy_yn,
application_type,
reversal,
reversal_session_id,
reversal_transaction_id,
reversal_stock_unit )
SELECT
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED *****
  A.fad_hash, -- fad_hash ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_accounting_code ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
  A.branch_accounting_code, -- branch_code ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
  nnnnn, -- amount ***** TO BE SPECIFIED *****
  n, -- entry_method ***** TO BE SPECIFIED *****
  A.current_jsn + 1, -- journal_seq_number ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
  99, -- node_id ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
  nnnn, -- prod_id ***** TO BE SPECIFIED *****
  nnnn, -- quantity ***** TO BE SPECIFIED *****
  nnnn, -- session_id ***** TO BE SPECIFIED *****
  nn, -- transaction_mode_id ***** TO BE SPECIFIED *****
  nnnn, -- transaction_id ***** TO BE SPECIFIED *****
  SYSTIMESTAMP, -- insert_timestamp ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
  TO_DATE ('nn/nn/nnnn 00:00:00', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
  TO_DATE ('nn/nn/nnnn 00:00:00', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_end_date ***** TO BE SPECIFIED *****
  TO_DATE ('nn/nn/nnnn 00:00:00', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_start_date ***** TO BE SPECIFIED *****
  'x', -- fallback_mode ***** TO BE SPECIFIED *****
  ':bind_SSC_user', -- branch_user ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
  'x', -- stock_unit ***** TO BE SPECIFIED *****
  NULL, -- agent_sla_info
  NULL, -- amount_requested
  NULL, -- bank_client
  NULL, -- balance_period
  n, -- card_impounded ***** TO BE SPECIFIED *****
  NULL, -- counter_sla_info
  NULL, -- currency
  n, -- fee ***** TO BE SPECIFIED *****
  n, -- financial_transaction ***** TO BE SPECIFIED *****
  NULL, -- receipt_line_id
  NULL, -- recovery_flag
  NULL, -- response_code
  NULL, -- routing_gateway
  NULL, -- scheme_id
  NULL, -- trading_period

```



```

NULL,    -- transaction_mode_version_no
NULL,    -- transaction_result_code
NULL,    -- transaction_type
NULL,    -- receipt_date
NULL,    -- settlement_date
NULL,    -- authorisation_code
NULL,    -- bank_transaction_id
NULL,    -- c12_xml
NULL,    -- pan
NULL,    -- horizon_transaction_id
'x',    -- transferred_from_legacy_yn      ***** TO BE SPECIFIED *****
'x',    -- application_type      ***** TO BE SPECIFIED *****
NULL,    -- reversal
NULL,    -- reversal_session_id
NULL,    -- reversal_transaction_id
NULL    -- reversal_stock_unit
FROM dual,
      (SELECT fhom.branch_accounting_code,
               fhom.fad_hash,
               tctc.current_jsn
        FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
             ops$brdb.brdb_txn_corr_tool_ctl tctc
       WHERE fhom.branch_accounting_code = :bind_branch_code
         AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.8 Template for BRDB_RX REP EVENT DATA

/*
The following SQL is a template for the SQL to be placed in a transaction file for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below with the values you wish to insert. Please take great care that the values are correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_rep_event_data
  ( event_date,
    fad_hash,
    branch_accounting_code,
    branch_code,
    node_id,
    insert_timestamp,
    app_server_node_name,
    brdb_instance_name,
    balance_period,
    event_id,
    trading_period,
    branch_user,
    event_detail,
    stock_unit,
    changed_branch_user,
    journal_seq_number )
SELECT
  TO_DATE ('nn/nn/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- event_date
***** TO BE SPECIFIED *****

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

A.fad_hash, -- fad_hash      ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
A.branch_accounting_code, -- branch_accounting_code      ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
A.branch_accounting_code, -- branch_code      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
99, -- node_id      ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
SYSTIMESTAMP, -- insert_timestamp      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
'999', -- app_server_node_name      ***** DO NOT CHANGE THIS VALUE FROM THE
ONE SPECIFIED !!! *****
':bind_instance_name', -- brdb_instance_name      ***** DO NOT CHANGE THIS
VALUE FROM THE ONE SPECIFIED !!! *****
NULL, -- balance_period
NULL, -- event_id
NULL, -- trading_period
':bind_SSC_user', -- branch_user      ***** DO NOT CHANGE THIS VALUE FROM THE
ONE SPECIFIED !!! *****
NULL, -- event_detail
'x', -- stock_unit      **** TO BE SPECIFIED ****
NULL, -- changed_branch_user
A.current_jsn + 1 -- journal_seq_number      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
FROM dual,
(SELECT fhom.branch_accounting_code,
fhom.fad_hash,
tctc.current_jsn
FROM ops$brdb.brdb_fad_hash_outlet_mapping fhom,
ops$brdb.brdb_txn_corr_tool_ctl tctc
WHERE fhom.branch_accounting_code = :bind_branch_code
AND tctc.branch_accounting_code = fhom.branch_accounting_code) A;

```

8.9 Template for BRDB_RX REP SESSION DATA

/*
The following SQL is a template for the SQL to be placed in a transaction file
for the BRDB Transaction Correction tool (BRDBX015) to apply to the BRDB.

Unless otherwise specified, please replace the hard-coded values specified below
with the values you wish to insert. Please take great care that the values are
correct, as incorrect values can seriously affect the integrity of the system.

All rows inserted are audited.

Date	Author	Purpose
----	-----	-----

*/

```

INSERT INTO ops$brdb.brdb_rx_rep_session_data
( journal_date,
fad_hash,
branch_accounting_code,
branch_code,
amount,
counter_mode_id,
node_id,
journal_seq_number,
quantity,
prod_id,
session_id,
transaction_id,

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```

insert_timestamp,
trading_date,
transaction_start_date,
branch_user,
stock_unit,
whole_seq_id,
balance_period,
receipt_line_id,
reversal_node_id,
trading_period,
transaction_mode_id,
transaction_mode_version_no,
reversal_timestamp,
update_timestamp,
report_xml,
reversal_user,
transferred_from_legacy_yn,
is_recoverable_yn )
SELECT
    TO_DATE ('nn/nm/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- journal_date
***** TO BE SPECIFIED *****
    A.fad_hash, -- fad_hash      ***** DO NOT CHANGE THIS VALUE FROM THE ONE
SPECIFIED !!! *****
    A.branch_accounting_code, -- branch_accounting_code      ***** DO NOT CHANGE
THIS VALUE FROM THE ONE SPECIFIED !!! *****
    A.branch_accounting_code, -- branch_code      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    nnnnnn, -- amount      ***** TO BE SPECIFIED *****
    NULL, -- counter_mode_id
    99, -- node_id      ***** DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED
!!! *****
    A.current_json + 1, -- journal_seq_number      ***** DO NOT CHANGE THIS VALUE
FROM THE ONE SPECIFIED !!! *****
    nnnn, -- quantity      ***** TO BE SPECIFIED *****
    nnnn, -- prod_id      ***** TO BE SPECIFIED *****
    nnnn, -- session_id      ***** TO BE SPECIFIED *****
    nnnn, -- transaction_id      ***** TO BE SPECIFIED *****
    SYSTIMESTAMP, -- insert_timestamp      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    TO_DATE ('nn/nm/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), -- trading_date
***** TO BE SPECIFIED *****
    TO_DATE ('nn/nm/nnnn nn:nn:nn', 'DD/MM/YYYY HH24:MI:SS'), --
transaction_start_date      ***** TO BE SPECIFIED *****
    ':bind_SSC_user', -- branch_user      ***** DO NOT CHANGE THIS VALUE FROM
THE ONE SPECIFIED !!! *****
    'x', -- stock_unit      ***** TO BE SPECIFIED *****
    NULL, -- whole_seq_id
    NULL, -- balance_period
    NULL, -- receipt_line_id
    NULL, -- reversal_node_id
    NULL, -- trading_period
    nn, -- transaction_mode_id      ***** TO BE SPECIFIED *****
    NULL, -- transaction_mode_version_no
    NULL, -- reversal_timestamp
    NULL, -- update_timestamp
    NULL, -- report_xml
    NULL, -- reversal_user
    NULL, -- transferred_from_legacy_yn
    NULL -- is_recoverable_yn
FROM dual,
    (SELECT fhom.branch_accounting_code,
        fhom.fad_hash,
        tctc.current_json

```



HOST BRANCH DATABASE SUPPORT GUIDE
COMMERCIAL IN CONFIDENCE



```
FROM  ops$brdb.brdb_fad_hash_outlet_mapping fhom,
      ops$brdb.brdb_txn_corr_tool_ctl tctc
WHERE  fhom.branch_accounting_code = :bind_branch_code
AND    tctc.branch_accounting_code = fhom.branch_accounting_code) A;
```