FUJ00164815

Document Title: Host BRDB Transaction Correction Tool Low Level Design

Document Reference: DEV/APP/LLD/0142

Document Type: Low Level Design

Release: (Not Applicable)

Abstract: This document is the low level design for the Branch Database transaction correction tool process that is used to add compensating correction records to transactional/accounting/stock tables.

Document Status: DRAFT

Author & Dept: Rajesh Shastri

Internal Distribution: David Harrison

External Distribution: None

Approval Authorities:

Name	Role	Signature	Date
Graham Allen	HNG-X Development		

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

FUJ00164815

0 Document Control

0.1 Table of Contents

FUJ00164815

FUJ00164815

0.2 Document History

Version No.	Date	Summary of Changes and Reason for Issue	Associated Change - CP/PEAK/PPRR Reference
0.1	10-Oct-07	Draft Version	
0.2	01-Nov-2007	Fix design issues	
0.3	01-Nov-2007	Incorporate HLD changes	
0.4	13-Nov-2007	Amend section 2.4.1 and 3.1 to provide more details	
0.5	03-Jul-2009	Revise to reflect major revisions made for PC0178207	
0.6	29-Sep-2009	[CW] Add Transaction Correction Journal Auditing	

0.3 Review Details

Review Comments by :	31-Oct-2007
Review Comments to :	GRO &

Mandatory Review	
Role	Name
Solution Design / Infrastructure Design	Nasser Siddiqui
System Test	Harjinder Hothi
SSC	Mik Peach
Optional Review	
Role	Name
Security	Bill Membery
Business Continuity	Tony Wicks
Service Support	Peter Thompson
HNG-X Service Transition	Steve Godson
Service Network	Alex Kemp
Data Centre Migration	Martin Brett
SV&I Manager	Sheila Bamber
Tester	Hamish Munro
RV Manager	James Brett (POL)
VI Manager	Peter Rickson
TE Manager	Peter Rickson
Development Host Team Manager	David P Harrison
Development Host Team Member	Graham Allen
Development Host Team Member	David Pooley
Development Host Team Member	Anona Stevens
Development Host Team Member	Wing Pang

FUJ00164815

Development Host Team Member	Steve Goddard
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 Services Post Office Account HNG-X Document Template	Dimensions
DES/APP/HLD/0020	0.5	29-Oct-07	Branch Database HLD	Nasser Siddiqi

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
BRDB	Branch Database
HNG-X	Horizon Next Generation X
SSC	Support Service Centre

0.6 Glossary

Term	Definition
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.
Instance	An instance is composed of memory structures and the Oracle background processes that run on a server.

0.7 Changes Expected

Changes

0.8 Accuracy

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

FUJ00164815

0.9 Copyright

FUJ00164815

1 Description

1.1 Overview

This document provides the low level design for the branch database transaction correction tool module. The utility will allow SSC to correct transactions by inserting balancing records to transactional / accounting / stock tables in the BRDB system. It will also audit the changes made. There will be no updating / deleting of records in the Branch database.

Warning: The use of this powerful tool has inherent risks. If the SQL statement is incorrect or badly written, it is possible to cause unintended consequences, some of which may cause serious problems to the Branch Database. It is expected that only a small number of skilled staff will run this tool and that they will have detailed guidance as to when and how to use the tool.

1.2 Solution Components

There are 5 main components to the solution:

- Unix shell script BRDBX015.sh which is in the /app/brdb/trans/support/brdbx015 directory. It is deliberately kept separate from the standard $BRDB_SH directory so that access to the script and the associated components can be restricted to authorised users. The shell script calls the PL/SQL package PKG_BRDB_TXN_CORRECTION.
- PL/SQL package PKG_BRDB_TXN_CORRECTION, which resides within the Branch Database and is owned by Oracle user OPS$SUPPORTTOOLUSER. The PL/SQL package is the component that validates, creates and audits the balancing transaction.
- A set of template files, one for each transaction table for which balancing transactions are allowed to be inserted. Each file contains a template for a SQL INSERT statement for the table in question. This makes it easier for users to produce new transaction files by basing them on the template files
- Branch ‘seeding’ - as part of it’s processing, the Transaction Correction Tool uses table brdb_txn_corr_tool_ctl. To allow new branches to be processed by the Tool, the estate management interface has been modified to add a ‘seed’ record to this table for each new branch. The ‘seed’ records have a node_id of 99 and current_jsn of zero.
- Transaction correction journal auditing – a new process generates audit files for the input day's auditable transaction correction records. See section 5 for details.

1.3 Assumptions

It is assumed that the insert statement being passed in to balance the record is a valid SQL statement and is not more than 32K in size. If not, the process will fail with error code 1.

FUJ00164815

2 Module Positioning

2.1 Calling Modules

No calling modules. The BRDBX015 will be initiated manually.

2.2 Called Modules

BRDBX015 calls PL/SQL module PKG_BRDB_TXN_CORRECTION and common packages PKG_BRDB_EXCEPTION, PKG_BRDB_COMMON and PKG_BRDB_FEED_COMMON.

2.3 Business Functions Implemented

DESAPPHLD0020 Branch Database High Level Design V0.4 Section 7.2.12

2.4 Objects Used

2.4.1 Database Objects used

Object Name	Object Type	Ins	Sel	Upd	Del	Oth
BRDB_OPERATIONAL_EXCEPTIONS	Table	X	X	X		
BRDB_SYSTEM_PARAMETERS	Table		X			
BRDB_FAD_HASH_OUTLET_MAPPING	Table		X			
BRDB_PROCESS_AUDIT	Table	X	X	X		
BRDB_PROC_AUD_SEQ	Seq		X			
BRDB_TXN_CORR_TOOL_JOURNAL	Table	X	X	X		
BRDB_FAD_HASH_CURRENT_INSTANCE	Table		X			
BRDB_TXN_CORR_TOOL_CTL	Table	X	X	X		
BRDB_BRANCH_INFO	Table		X			
BRDB_OPER_EXCP_SEQ	Seq		X			

The following transaction tables have been granted INSERT privileges to OPS$SUPPORTTOOLUSER. The transaction correction statement is only allowed to insert into these tables.

Object Name	Object Type	Ins	Sel	Upd	Del	Oth
BRDB_RX_BUREAU_TRANSACTIONS	Table	X				
BRDB_RX_EPOSS_TRANSACTIONS	Table	X				
BRDB_RX_APS_TRANSACTIONS	Table	X				

FUJ00164815

BRDB_RX_EPOSS_EVENTS	Table	X				
BRDB_RX_NWB_TRANSACTIONS	Table	X				
BRDB_RX_REP_SESSION_DATA	Table	X				
BRDB_RX_DCS_TRANSACTIONS	Table	X				
BRDB_RX_REP_EVENT_DATA	Table	X				
BRDB_RX_CUT_OFF_SUMMARIES	Table	X				

The BRDBX015.sh script logs into Oracle as '/' (i.e. OPS$<SSCusername>), therefore in order to run, all of the Oracle OPS$ users for the SSC users require database privileges on objects in OPS$BRDB as follows:

Object Name	Object Type	Ins	Sel	Upd	Del	Exe
PKG_BRDB_TXN_CORRECTION	Package					X

Note that the create_db_user.sh script has been changed to grant the above privilege to new SSC Oracle users when they are created. Existing SSC Oracle users will need to have the privilege granted to them manually.

2.1.2 Files Used

The process uses the following files:

- Transaction file containing a SQL INSERT statement that creates the required balancing transaction. The file must be placed in this directory:

  /app/brdb/trans/support/brdbx015/input

- If the process completes successfully, the transaction file will be moved to:

  /app/brdb/trans/support/brdbx015/output

- Log file will be written to:

  /app/brdb/trans/support/brdbx015/log

  Log file will be named using the following convention:

  <transaction_file_name>_<CCYYMMDDHHMISS>.log

The /app/brdb/trans/support/brdbx015/input directory is referenced by an Oracle directory object – BRDBX015_DIR – note that the READ privilege on this object must be granted to the OPS$SUPPORTTOOLUSER user.

The PL/SQL package reads the transaction file using the standard Oracle UTL_FILE mechanism. In order for this to be possible, the EXECUTE privilege on SYS.UTL_FILE must be granted to the OPS$SUPPORTTOOLUSER user.

FUJ00164815

3 Processing

3.1 Method

Having logged into their own Unix user, the SSC team members will change directory to the /app/brdb/trans/support/brdbx015 directory and place their transaction file in the /app/brdb/trans/support/brdbx015/input sub-directory. They will then invoke BRDBX015 manually. The shell script module will be owned by the Unix user "supporttooluser".

The module will read the contents of the input transaction file, which will be in the form of a SQL INSERT statement. Only a single insert statement is allowed and (after an optional introductory comment) it must start with the 'INSERT INTO' clause. The tables referred to in the insert statement must be prefixed with the user OPS$BRDB.

The module will first validate the transaction file (see below for details of the specific validation carried out). If the file is valid, then the module will log the SQL statement in the BRDB_TXN_CORR_TOOL_JOURNAL table, execute it as OPS$SUPPORTTOOLUSER, and finally commit. If the process completes successfully, the input transaction file will be moved to "/app/brdb/trans/support/brdbx015/output".

Example:

- Login to the Unix box as an authorised user
- Ensure that the standard environment variables are set (e.g. $ORACLE_SID etc)
- Change directory to /app/brdb/trans/support/brdbx015
- Place transaction file in /app/brdb/trans/support/brdbx015/input
- From the Unix command prompt, execute the following

  ./BRDBX015.sh MyTransactionFile.sql 2001

  where the first parameter is the transaction file name and the second parameter is the branch code where the balancing transaction is going to be applied. Note that the branch code must exist in the database, and must not be for a closed branch. If this is not the case, then an error message will be shown and the run aborted.

  The transaction file is tightly constrained in terms if what it may contain. Details of these constraints together with template transaction files for each of the tables for which balancing transactions are allowed are provided later in this document.

  If valid, the SQL statement is modified by substituting actual values for bind variables, and the modified SQL statement is executed and logged in the journal as an XML string.

More details on the process and the risks will be defined in the Branch Database Support Guide DES/APP/SPG/0001.

3.2 Initialisation

Initialise the process by writing process audit information to BRDB_PROCESS_AUDIT.

3.3 Recovery

If an Oracle node/instance failure occurs, the utility will fail with an error code of 99. For all other failures, it will fail with an error code of 1 and log an operational exception in BRDB_OPERATIONAL_EXCEPTIONS.

FUJ00164815

Note that the tool does not use the standard process control handling since the nature of the tool makes its use inappropriate.

FUJ00164815

3.4 Transaction File

3.4.1 Content

The transaction file contains a SQL statement that, when executed, will insert a balancing transaction. The SQL must be a single INSERT statement with a standard form and layout. Additionally, the tool can be configured to enforce certain values for certain fields. Templates for the different INSERT statements that are required for the different transaction tables are provided. The following example illustrates a valid transaction file.

```
/*
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.

The values listed below are illustrative only and are not to be treated as
default values for the fields.

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,
```

FUJ00164815


```
  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 ('29/04/2009 00:00:00', 'DD/MM/YYYY HH24:MI:SS'),  -- trading_date
  ':bind_SSC_user',  -- branch_user     ********* DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED !!! **********
  27.12,  -- amount
  99,  -- node_id     ********* DO NOT CHANGE THIS VALUE FROM THE ONE SPECIFIED !!! **********
  78,  -- quantity
  NULL,  -- pouch_id
  54673,  -- prod_id
  12,  -- transaction_mode_id
  NULL,  -- week_number
  NULL,  -- transaction_end_date
  'T'   -- 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;
```

FUJ00164815

As can be seen from the above example, the statement begins with a comment describing the SQL. This should be tailored as appropriate. The SQL itself must be a single insert statement of the form INSERT INTO ….. SELECT ….. FROM dual, (SELECT ….. FROM …. WHERE …..). Therefore only a single balancing transaction can be created per run of the correction tool.

The sub-query fetches values for some important columns based upon the branch code passed as an argument to BRDBX015, and must be present in the SQL. Where these important values are used, they must not be changed. Any values that must remain unchanged are commented as such in the supplied templates.

The SQL also includes a number of bind variables (e.g :bind_branch_code). Actual values for these fields will be substituted into the SQL before it is executed. The available bind variables are listed below, together details of the values that will be substituted for them if they appear in a transaction file:

- :bind_branch_code : substituted with the values of the branch code argument of BRDBX015
- :bind_SSC_user : substituted with the Oracle user that is carrying out the actual insert i.e. SUPPORTTOOLUSER
- :bind_instance_name : substituted with the name of the Oracle instance upon which the tool is run

3.4.2 Constraints

The correction tool places a number of constraints on the contents of the transaction file. These are necessary in order to provide a defined baseline upon which it can base its operation. The constraints are as follows:

- The transaction file must be less than 32K in size
- The transaction file must only contain Unix-style end of line markers (EOL), not DOS format end of line markers (CR/EOL)
- The transaction file can only contain a single SQL statement. If more than one balancing transaction is required then more than one transaction file must be created, each of which is executed with a separate run of the tool
- If the transaction file contains an introductory comment, then it must be a '/* …… */' style comment, not a '-- ……' style comment
- The closing '*/' of the introductory comment must have a trailing space (i.e. '….. */ ')
- The run symbol at the end of the SQL must be a ';' , not '/', and must have a trailing space (i.e. '…..; ')
- The SQL must be a valid SQL statement according to the normal Oracle SQL parsing rules (e.g. valid syntax, objects accessible etc)

FUJ00164815

- The SQL must begin with 'INSERT INTO OPS$BRDB.' and be of the form 'INSERT INTO ….. SELECT ….. FROM dual, (SELECT ….. FROM …. WHERE …..)'.
- The table name must be one of the tables named in the BRDB_TXN_CORRECTION_ALLOWED_TABLES1 or BRDB_TXN_CORRECTION_ALLOWED_TABLES2 configuration parameters
- All of the columns that exist on the table in question must be explicitly named. It is not necessary for every listed column to be on a separate line, but this is advisable for readability.
- The values to be inserted must be provided by the 'SELECT … FROM dual …'. Each value must be on a separate line. Trailing comments are allowed, but must be a '-- …..' style comment. Any such comment must not include any commas. All columns must have values provided for them (even if that value is NULL).
- Certain columns are common between a subset of the transaction tables. In some cases, these columns should be set to the same value no matter what table is in use. With the exception of the bind variables listed earlier, the value that the SQL will try to insert is under the control of the user (i.e. it is determined by the value specified in the SQL). However, the tool can be configured to validate that the value specified in the SQL matches that expected. In order to do this, set the BRDB_TXN_CORRECTION_ENFORCED_VALUES configuration parameter to include the field and the required value.

  The parameter is populated as a comma-delimited list of name/value pairs, where the name is the name of the column name, and the value is the value to be enforced. As released, this configuration parameter is set to:

  NODE_ID=99,APP_SERVER_NODE_NAME=999,BRANCH_USER=:bind_SSC_user,BRDB_INSTANCE_NAME=:bind_instance_name

  which, for example, ensures that if a 'node_id' column exists on the transaction table, it's value is specified as 99. If there is no 'node_id' on the transaction table, then no value is enforced for that field. Note that if the parameter does not exist, then no values are enforced in the SQL.

3.4.3 Templates

Each of the transaction tables that are allowed to have balancing transactions inserted on them has an associated template file. Each file contains a template of an INSERT statement for that table, in the required format, and listing all of the columns on the table. Users should create their own transaction file based upon the relevant template file, substituting the values they require into the SQL. Note that some of the column values specified in the template should not be changed – these are annotated with comments as appropriate.

The collection of template files are available from source control along with the other components.

FUJ00164815

FUJ00164815

4 Core Procession

4.1 Error Handling

Any errors within the package will be handled by standard exception handler pkg_brdb_exception.fn_brdb_log_oracle_exception.

Note that the tool does not use the standard process control handling since the nature of the tool makes it's use inappropriate.

4.2 Data Structure

4.2.1 Global Data

Variable Name	Usage
NA	

4.2.2 Static Data

4.2.3 System Parameters

Parameter Name	Select	Update	Delete
DEBUG_LEVEL_FOR_BRDBX015	X		
PKG_BRDB_TXN_CORRECTION_DEBUG_LEVEL Single-digit Integer Number : Mandatory Determines the level of debugging messages that the PL/SQL process should output. Allowed values: 0 (minimum) to 3 (maximum).	X		
BRDB_TXN_CORRECTION_ENFORCED_VALUES Character String : Optional Contains a comma-delimited list of name/value pairs for fields that may exist in the transaction files. The tool validates that if a configured name exists in the field list of the SQL, the configured value matches that in the SQL. If the parameter does not exist, then no values are enforced	X		
BRDB_TXN_CORRECTION_ALLOWED_TABLES1 Character String : Mandatory	X		

FUJ00164815

Contains a comma-delimited list of tables on which the tool is allowed to insert balanced transactions			
BRDB_TXN_CORRECTION_ALLOWED_TABLES2 Character String : Mandatory Contains a comma-delimited list of tables on which the tool is allowed to insert balanced transactions	X		

4.2.4 Environment Variables

Standard environment variables used by OPS$SUPPORTTOOLUSER user.

4.2.5 Constants

4.2.6 Command–line parameters

Position	Description	Possible Values
1 (Mandatory)	Transaction File Name	Character
2 (Mandatory)	Branch Code	Numeric

4.3 Function/Subroutine Call Hierarchy

Validate Parameters

Process Audit Log

Read Transaction File

Validate Transaction File

Audit SQL Statement

Execute SQL Statement

Commit Transaction

Process Audit Log

Move Transaction File

4.3.1 Validate Parameters

Pseudo-code:

```
If fewer than or more than 2 parameters passed then

   Print 'Invalid number of command-line parameters'

   Return 1
Fi
```

FUJ00164815


```
Check that the transaction file name exists and is readable

If error then
   Print 'Could not open transaction file name identified by <name>'

   Return 1
Fi

Check that the transaction file name is not empty

If error then
   Print 'Transaction file is empty'

   Return 1
Fi

Check that the transaction file size does not exceed 32K chars

If error then
   Print 'Transaction file size exceeds limit of 32K'

   Return 1
Fi
```

4.3.2 Process Audit Log

Process audit log at beginning of process execution by calling standard package procedure pkg_brdb_common.pr_process_audit

4.3.3 Read Transaction File

Read the transaction file using UTL_FILE method. This file contains the SQL insert statement to create the correction record in the BRDB database.

4.3.4 Validate Transaction File

The SQL in the transaction file is validated as follows. Any validation failures are displayed to standard output and logged to the log file.

- Check that the file does not contain any carriage returns, indicating DOS format EOL markers
- Check that the SQL in the transaction file parses according to the standard Oracle rules (e.g. syntax, privileges etc). This is done using the standard Oracle DBMS_SQL.PARSE procedure.
- Check that there is only a single SQL statement in the transaction file. Note that in most cases, this will be detected by the previous parsing step. However, the fact that the parsing does this is not described in the Oracle documentation, so it may be changed in future releases of Oracle. Therefore, this validation provides security if the behaviour of the Oracle procedure is changed at a later date.
- Check that the SQL begins with '`INSERT INTO OPS$BRDB.`'
- Check that the table named in the SQL is one of the tables listed in the two BRDB_TXN_CORRECTION_ALLOWED_TABLES<n> configuration parameters. Note that as long as the privileges are set up correctly (i.e. OPS$SUPPORTTOOLUSER only

FUJ00164815

has insert privileges on the allowed tables), any attempt to insert a balancing transaction on a non-allowed table will cause the previous parsing step to fail (because the user would not have the necessary privileges). Therefore, this validation provides security in case the privileges are not correctly set up.

- Check that all the columns named in the SQL exist on the table, and that all the columns on the table are named in the SQL
- Check that the values to be inserted are provided by a SELECT … FROM dual, (SELECT … FROM … WHERE) i.e. not a VALUES
- Check that if any of the name/value pairs that are listed in the BRDB_TXN_CORRECTION_ENFORCED_VALUES configuration parameter are present on the table, they are set to the listed value.

4.3.5 Audit SQL Statement

The SQL statement being executed will be logged in the table BRDB_TXN_CORR_JOURNAL. The format of the data to be written to the column JOURNAL_XML is:

"<?xml version="1.0" encoding="UTF-8"?>

<Support_Insert>

<Unix_User>Unix User Name</Unix_User>

<Oracle_User>Oracle User Name</Oracle_User>

<Sql>SQL Statement</Sql>

</Support_Insert>"

where :

- Unix User Name is the Unix user name under which the user logged in
- Oracle User Name is Oracle user that is carrying out the actual insert i.e. SUPPORTTOOLUSER
- SQL Statement is the final (i.e. after substituting actual values for bind variables) SQL that is executed to insert the balancing transaction

4.3.6 Execute SQL Statement

The SQL statement is executed using the EXECUTE IMMEDIATE command and run under the privileges of the PL/SQL package owner (OPS$SUPPORTTOOLUSER). The number of rows inserted will be written to standard output.

4.3.7 Commit Transaction

Commit all transactions. An Oracle error at this point will be trapped by the standard exception handler.

4.3.8 Process Audit Log

Process audit log at the end of process execution by calling standard package procedure pkg_brdb_common.pr_process_audit.

FUJ00164815

4.1.9 Move Transaction File

If the process completes successfully (exit code 0), move the transaction file from /app/brdb/trans/support/brdbx015/input to /app/brdb/trans/support/brdbx015/output directory. If the process fails (e.g. transaction file is found to be invalid), then the transaction file will not be moved and an error message will be written to standard output.

FUJ00164815

5 Transaction Correction Journal Auditing

The transactions that are applied via the Transaction Correction Tool are audited via a new process, namely BRDBC033. This process is essentially the same as the existing audit process (BRDBC002), and has been created using the same program code (with minor adaptations).

Note

The following sections detail the **specifics of the new process**, and do not duplicate the documentation for the main mechanisms and processes already found in BRDBC002.

5.1 Module Usage

The new module is called BRDBC033 and is written in Pro*C. It is initiated (typically from the BRDB schedule) as follows:

BRDBC033 Business-Day(CCYYMMDD) [FAD-Hash(n)]

The Business-Day is mandatory e.g. 20090928

The FAD-Hash is optional and allows the program to be executed for a specific FAD hash value. e.g. 27

5.2 Database objects used

The new process makes a new reference to the following database object(s):

Object Name	Object Type	Ins	Sel	Upd	Del	Oth
BRDB_FAD_HASH_VALUES	Table		X			

5.3 Environment variables

The new process uses the following new environment variables:

Environment Variable Name	Typical value
BRDB_TCT_AUDIT_OUTPUT	/app/brdb/trans/audit/tctaudit
BRDB_TCT_FILE_TEMP	/app/brdb/trans/support/working

5.4 System parameters

The new process uses the following new system parameters: (all are mandatory)

Parameter Name	Description	Type	Typical value

FUJ00164815

DEBUG_LEVEL_FOR_BRDBC033	Indicates the debug level for the program, and controls the amount of output that is generated.	Number	0
BRDBC033_MAX_FILE_SIZE	Indicates the maximum byte size of a Transaction Correction Journal Audit file.	Number	1073741824
BRDBC033_JOURNAL_SEQ_DENSE_CHECK	Indicates whether journal sequence number checking is required.	Text	TRUE

5.5 Files created

The new process creates audit files with the following naming convention:

AUDIT_TCT_<Trading date>_<FAD Hash>_<File seq>.aud

For example: AUDIT_TCT_20090814_036_001.aud

where…

1) <Trading date> is the business date (CCYYMMDD) for which the program is being run.

2) <FAD Hash> is the FAD Hash of the data contained within the file.

3) <File seq> is the file sequence number – starts at 1 and increments by 1 with each subsequent file for a trading date/FAD Hash.

5.6 Processing details

BRDBC033 is scheduled to run on one node within the RAC cluster, and processes all FAD hashes in a single run.

The process extracts data from table: brdb_txn_corr_tool_journal where journal_date is between 00:00 and 23:59 of the business day being processed and where column is_auditable = 'Y'.

The output files are initially created in the 'TEMP' directory, and then moved to the 'OUTPUT' directory on successful completion.

Unlike the files produced by BRDBC002, the files produced by BRDBC033 are not compressed.

The output files produced will be subsequently processed by the existing audit mechanism.