Oracle® Financial Services Lending and Leasing

Regression Testing Tool

Release 14.12.0.0.0
F82321-01
August 2024

Oracle Financial Services Lending and Leasing Regression Testing Tool, Release 14.12.0.0.0

F82321-01

Copyright © 2022, 2024, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1

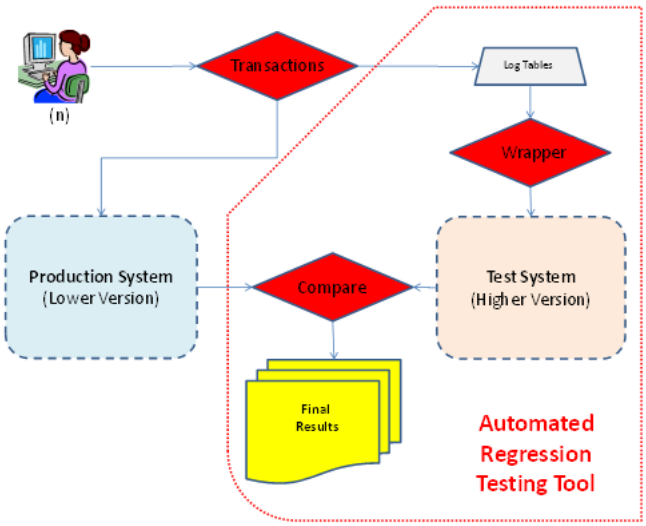
Regressions testing tool

The purpose of this document is to outline the high level requirements of the scope of the Testing tool for regression tests during DLS/OFSLL upgrades by performing a parallel testing between a lower version of DLS/OFSLL (production) and a higher one (test).

2
Scope

The testing tool will log certain operations done in the production system and replicate them in the Test system. This will enable parallel testing by comparison of the production and test systems, specifically with regards to the General Ledger transactions. The aim is not to replicate user keystrokes in the Production system but instead is to ensure that the eventual Posted values are captured in the Test system in the same way they were in the Production system.

Figure 2-1 Automated Regression Testing Tool



This tool was designed to work for all DLS/OFSLL upgrades, independently of versions and customizations.

The Testing Tool covers the Origination (App Entry, Funding, Account Creation) and Customer Service, as detailed below:

- Origination
  The testing tool will perform the following: -
- Customer Service
  The testing tool will perform the following: -

2.1 Origination

The testing tool will perform the following: -

App Entry:

1. Applications entered in Production on that day will be replicated in the Application tables of the Test system.

2. The application will pass through the Data Entry and Pre screen edits.
3. The application will pull the Credit bureau as defined by the current system parameter configuration.
4. The application will be placed in a New: Review Required status.

Funding:

1. All applications funded in Production on that day will be replicated in the Test environment. The application in the Test environment will be funded using the same values that were used to fund the application in Production.

Assumptions/Restrictions:

- Changes to the application values by the users in the Underwriting screen will not be replicated in the Test Environment.
- Intermediate steps between Application Entry and Funding viz. Fax/Letters on Status change, NADA pulls will not be replicated in the test system. This will require manual testing efforts. (The final values in the Production system will be used to fund the application)
- As part of the manual testing the user will not set the Application to Approved: Funded since it is possible that the values used by the user in the test system will be different from the values used in Production resulting indifferent GL entries.
- As part of the manual testing, the Fax numbers of all producers must be set to an internal fax number to avoid fax documents being set back to producers.

2.2 Customer Service

The testing tool will perform the following: -

Monetary Transactions posted on Accounts in Production will be replicated in Test for that particular day. This will allow comparison of the Transaction history on Accounts in the Test and Production instances. To replicate the monetary transactions, the Testing tool will perform the following:

1. Payments applied to Accounts in Production will be replicated in Test
    - Payment batches and details will be created in the replicated in the test system (TXNS_GROUP_TEMP, TXNS_TEMP table will be created)
2. Monetary transactions posted on Accounts in Production will be replicated in Test
    - Corresponding entries in the Customer Service > Maintenance tab will be available (BATCH_MODE_TXNS, BATCH_MODE_TXNS_TEMP)
3. Assessment of Work Order expenses on an Account will be handled as a monetary transaction on an account.

Assumptions

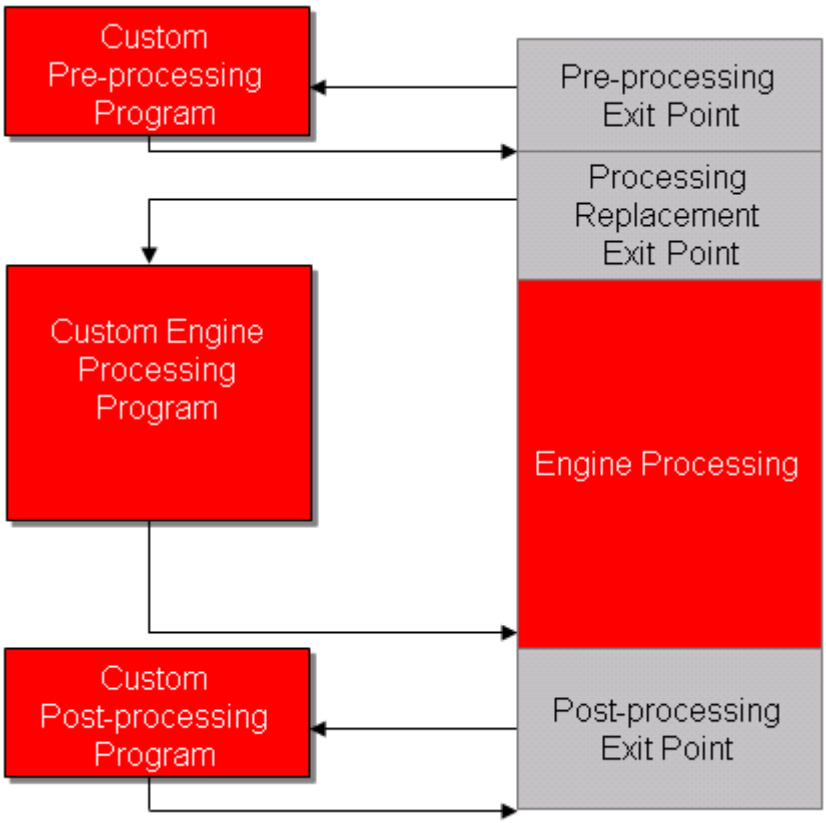
- Work Order actions will not be replicated. Work Order Expenses assessed on an account will be handled by the monetary transactions replication.
- A one-time refresh at the beginning of the testing will be done instead of a daily refresh. The only caveat would be that any errors, that affect data found during the testing, would remain in the database till the fix is received.
- The need for subsequent refreshes during the testing period will be discussed.

- Test environment Setup will be in sync with the setup in the Production environment. Setup changes to the production environment during the day will be replicated in the Test environment, before the Testing tool is run for that day.

3
Architecture / Technical Design

The Testing Tool was only possible due to OFSLL architecture, based on a wrapper- engine model. Like in OFSLL customizations, the Testing Tool takes advantage of the existing Exit Points to log and re-post the actions done by the users, allowing an automated parallel testing between the Production (lower version) and Test (higher version) environments of OFSLL.

Figure 3-1 Exit Point Diagram



The next two Diagrams show where in the Transaction Processing Engine the Testing Tool was introduced:

Figure 3-2 Non monetary Transaction Processing Engine

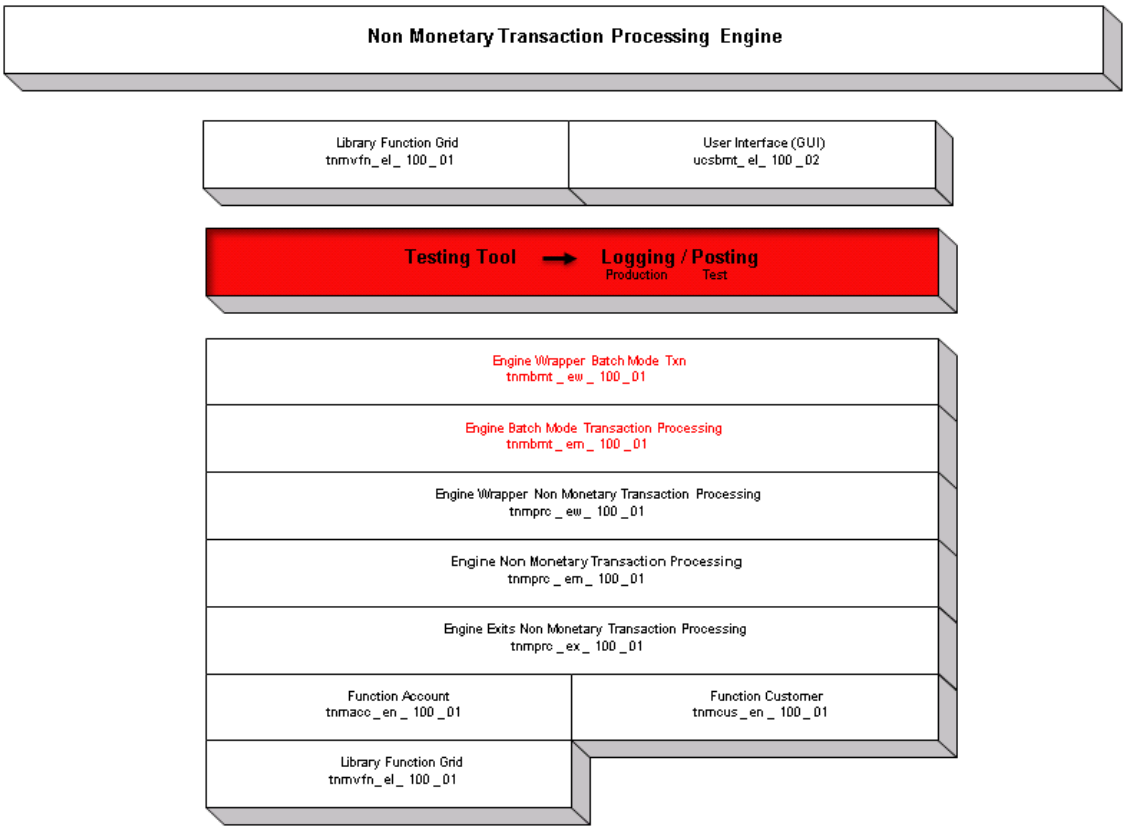
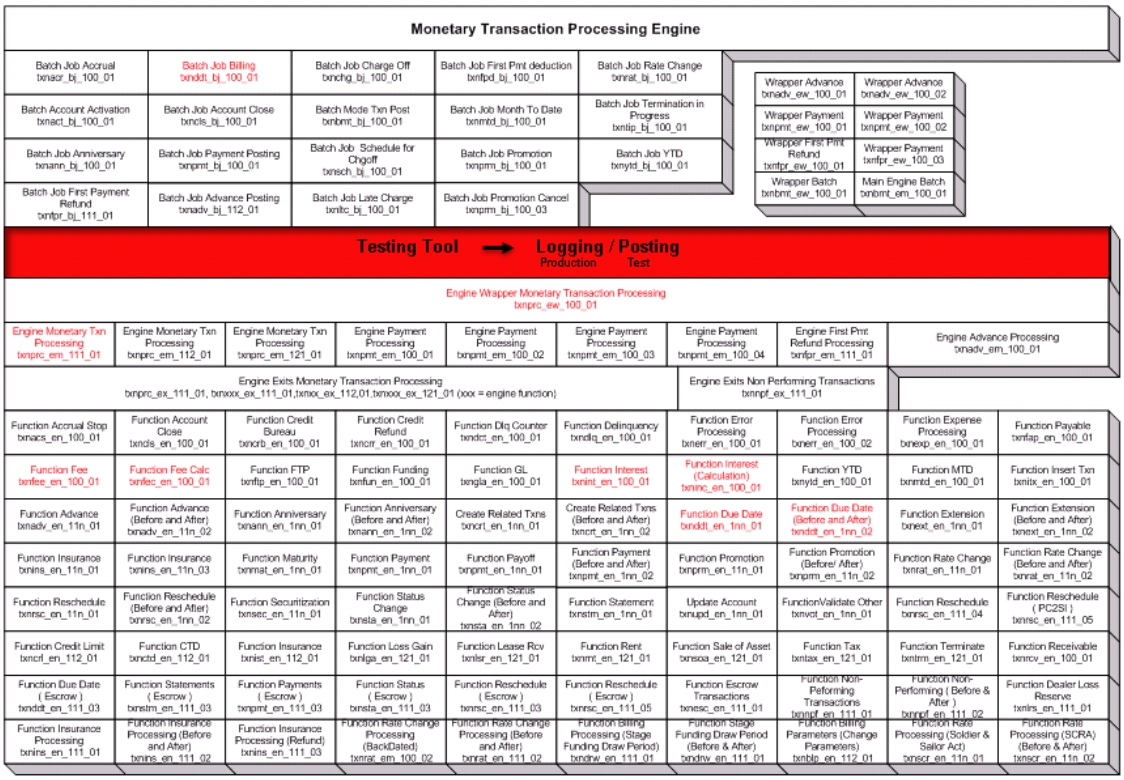
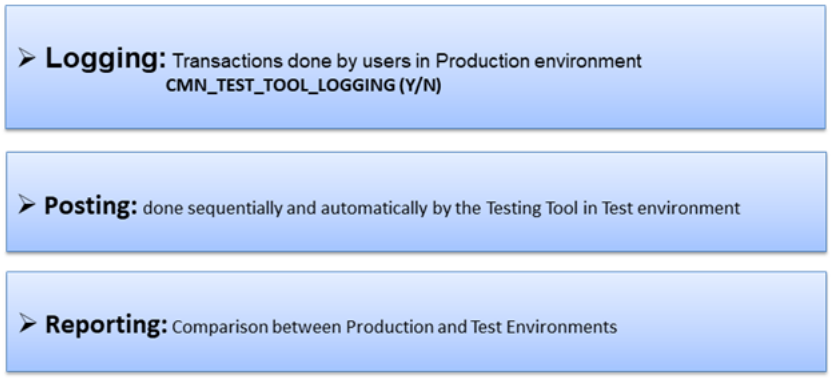


Figure 3-3 Monetary Transaction Processing Engine



The Testing Tool is divided in 3 main processes:

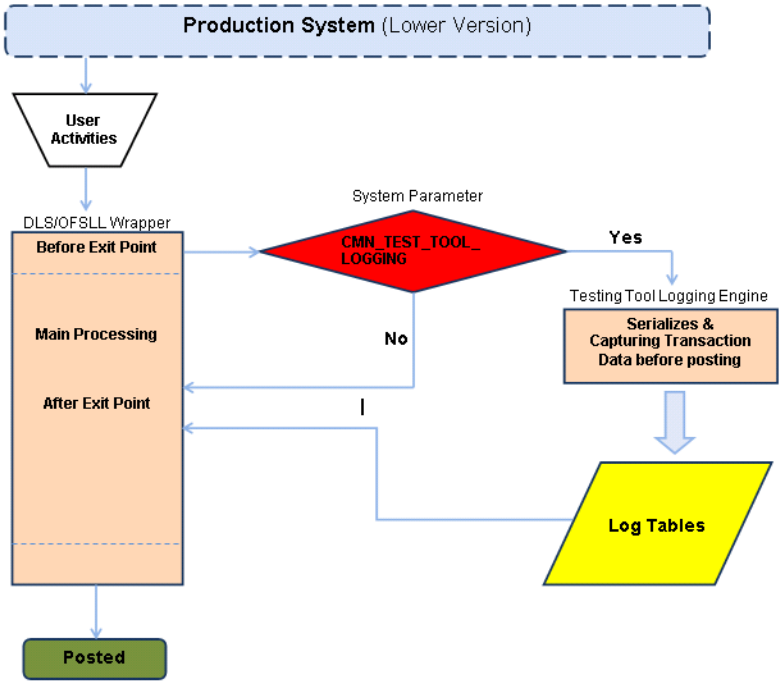
Figure 3-4 Testing Tool - Processes



Logging: Transactions done by users in Production environment

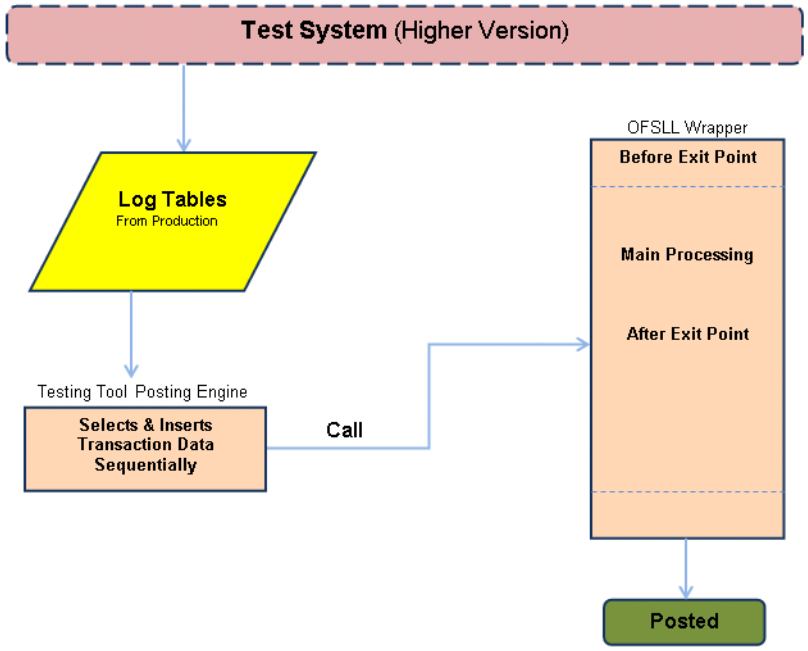
The logging process in Production environment can be turned on/off by setting the System Parameter CMN_TEST_TOOL_LOGGING. When its value is "**N**" system will behave normally. When its value is "**Y**", the system will log all the transactions done by the users.

Figure 3-5 Logging process



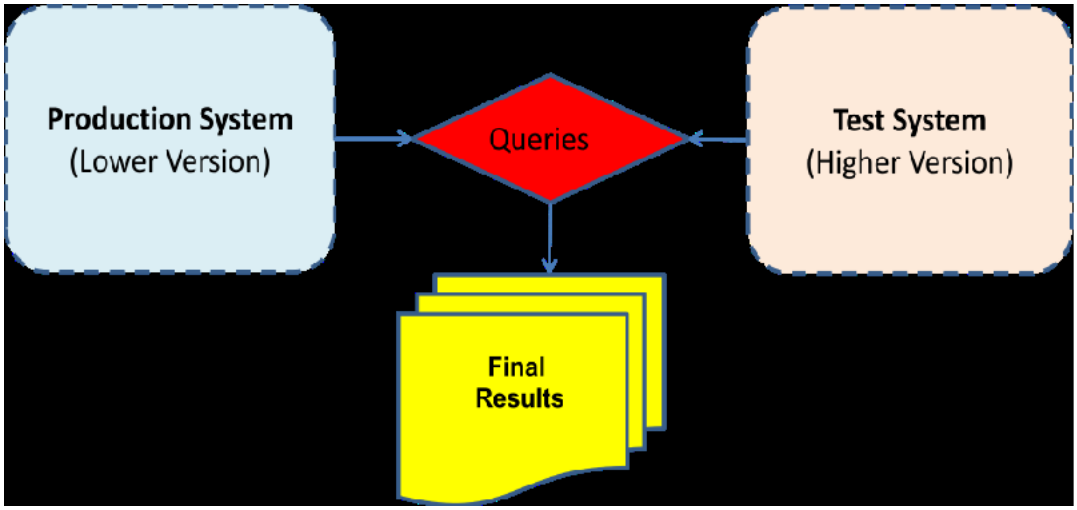
Posting: Done sequentially and automatically by the Testing Tool in Test environment

Figure 3-6 Posting process



Reporting: Comparison between Production and Test Environments

Figure 3-7 Reporting process



4

Steps to Clone Base OFSLL Environment

Following are the steps to Clone Base OFSLL Environment.

1. Copy x* packages and so from production to test after patch up of test schema. (some of these objects in production are newer versions than ones in upgrade patches.)
2. Perform export from production schema.
3. Confirm that CMN_SERVER_HOME path is identical between test and production servers.
4. Run **alter table** and **alter index** scripts against production schema.
5. Run **compare schema** scripts between the production and test schemas. Reconcile differences.
6. Perform export (no data) from test (upgraded) schema.
7. Perform import (ignore=yes) on production schema from step five's export. Confirm that packages, views, and types are correct versions.
8. Perform export from test (upgraded) schema of **setup** tables (LESS producers tables!).
9. Copy schema specific system parameters values from production schema
10. Truncate **setup** tables on production schema.
11. Perform import on production schema from step seven's export.
12. Overlay system parameters value from step 8
13. Create backup copies of folders in the CMN_SERVER_HOME tree on production app server.
14. Copy CMN_SERVER_HOME tree from test to production app servers. Consider results of step two.
15. Copy any other patch objects (.so's, etc.) into production environment. (If step 1 is done, we can copy without risk of overwriting newer versions in prod)
16. Restore any X* packages necessary. (if step 1 is done, we can copy without risk of overwriting newer versions in prod).

5
Automated Testing Procedure

Following is the **suggested** sequence of steps that would need to be performed on a daily basis as part of the **Automated Testing**.

Table 5-1 Automated Testing Procedure

Day 1
Test Database to be refreshed with Production values *
1. Truncate the Log tables in Production
2. Logging is Turned On in Production after last job has completed
Cut off time for manual transactions/user activities in the Production Environment. Logging by testing tool will be turned off.
1. Take a Production database snapshot (S1) of the required tables
2. Ensure sequences on test match sequences on Production (for applications and accounts)
3. Dat files and log tables are moved to the Test database
4. Testing tool is Run on Test instance (Ensure previous files are removed)
Comparison/Validation scripts are run against S1 and the Test database. Report (R1) is generated. (This gives us the comparison before the jobs are run)
A copy of input file in Production (lockbox) should be available in Test
Batch jobs are run (Batch job setup is identical to production) on both Test & Production
Take a Production database snapshot (S2) of the required tables
Comparison Scripts are run against S2 and the Test database. Report (R2) is generated. (This gives us the comparison before the jobs are run)
Day 2
Validates R1 & R2 and reports discrepancies to OFSLL
Investigate the discrepancies. Some may be reconciled and some may require a fix
Manual testing proceeds in the interim
Next run is planned
Test Database to be refreshed with Production values.

* Refresh Test database

Example: To run the testing tool in the Test environment on 4/23, a snapshot of the Production database AFTER the batch jobs have completed on the morning of 4/23 and before any other user activity for 4/23 begins needs to be taken. This snapshot will be used to refresh the test database for the testing tool to be run on the evening of 4/23.

6

Validations and Checks

Table 6-1 Validations and Checks

Before txns posting
Set logging parameter to '**N**'
Turn scheduler off on Test system
Ensure that setup tables are not truncated before refresh (only in case of a new refresh).
Ensure that jobs, job_sets, job_threads and job_buckets are copied over from Prod onto test (only in the case of new refresh).
Make sure that the batch jobs are setup the same in Prod and Test.
Ensure sequences are in sync with Production
Bump up sequences - recreate sequences
Ensure that the Test specific setup scripts are run for 1) all seed data 2) DML scripts to create data in the new tables
Check GL post date
Check log tables have just one day's transactions
Query by txn_tcd_code, check numbers match on Prod and Test
Drop index TXNT_LOG_UDX
Make sure that the batch jobs have no sequence numbers overlapping - the scheduler might not start because of this.
Run all the DML scripts - for seed data and others
Check for database locks on both test systems
Check run_dt_nxt on all batch jobs - should be updated correctly

7

Comparing Results

Once the tool run, we will be able to start comparing the results. Below is an example of a high level results extracted from a Production and a Test environment, where the tool replicated all the user actions executed during one day.

Table 7-1 Production and a Test environment

PRODUCTION		TEST	
TXN_TCD_CODE	SUM(TXN_AMT)	TXN_TCD_CODE	SUM(TXN_AMT)
ACCRUAL_STOP	0	ACCRUAL_STOP	0
ACC_MAINT_MONETARY_CBT	-1	ACC_MAINT_MONETARY_CBT	-1
ACTIVE	0	ACTIVE	0
ADV_RECOURSE_CHGOFF_CBT	5.25	CHGOFF	0
ADV_WAIVE	7	CREDIT_REFUND_REV	6312.85
CHGOFF	0	DDT	29034.13
CREDIT_REFUND	2875.01	DDT_REV	12494.97
CREDIT_REFUND_REV	6762.65	ERPO	15262
DDT	403310.84	ERPO_REV	855
DDT_REV	12494.97	ERPO_WAIVE	90
ERPO	18762	ERPO_WAIVE_REV	5
ERPO_REV	4355	ESVC	175
ERPO_WAIVE	108.42	ESVC_REV	75
ERPO_WAIVE_REV	5	EXTENSION	606.9
ESVC	175	EXTENSION_REV	0
ESVC_REV	75	FEXT	2093.28
ESVC_WAIVE	0.2	FEXT_REV	68.18
EXTENSION	606.9	FIN_1	2521.19
EXTENSION_REV	0	FIN_2	2357.28
FEXT	2093.28	FIN_3	10605
FEXT_REV	68.18	FLC	263.63
FIN_1	2521.19	FLC_REV	195.15
FIN_2	2357.28	FLC_WAIVE	15
FIN_3	10210	FNSF	0
FLC	8071.77	FUN_1	658203.33
FLC_REV	209.44	FUN_2	14259.2
FLC_WAIVE	15	FUN_3	85
FNSF	0	FUN_4	305.55

Table 7-1 (Cont.) Production and a Test environment

PRODUCTION		TEST	
FOTH1_WAIVE	5	INT	88904.16
FPHP	1659	INT_ESTIMATED	128900.35
FUN_1	630790.18	INT_REBATE	8654.78
FUN_2	13648.2	INT_REV	3038.93
FUN_3	85	LNR	788796.75
FUN_4	284.55	NP_EXCESS	2388.78

8
Testing Tool Exceptions

Following is the list of exceptions/limitations of the testing tool

1. Direct Reversal of an Indirect transaction
2. Multiple Payment Hold Assessments/Reversals
3. Payment not allocated to Phone Pay Fee

1. **Direct Reversal of an 'Indirect' transaction**
   An Indirect transaction is one that is created by a different transaction. For example, FNSF is an indirect transaction created by the PAYMENT_VOID transaction. The tool replicates the PAYMENT_VOID transaction that in turn creates the FNSF. Along the same lines, reversal of the PAYMENT_VOID transaction will result in the reversal of the FNSF transaction (This is normal Ofsll processing) and will be replicated by the tool.

   However when the FNSF transaction is reversed directly, by using the Reverse button on the Transactions screen, the tool is unable to replicate that.

   Another example is the PAYMENT_NONCASH created by modifying the PAYMENT_ERROR transaction on the Payment Maintenance screen. The non-cash transaction is an indirect transaction and directly reversing it or further modifying it will not get replicated in the Test environment. To summarize direct reversal/modification of the **child** transaction is not replicated.

   Note, the above restriction applies only for the same dataset. If the database is refreshed after the creation of the **child** transaction and before its direct reversal, the tool replicates the reversal.

2. **Multiple Payment Hold Assessment/Reversals**
   Since the tool is not recording the call activities, multiple payment holds applied or removed on the same day is NOT replicated. To paraphrase, the existence/absence of the Payment Hold condition will match that in production, however intermediate changes will not be reflected.

   Examples

- Scenario 1
- Scenario 2 – This scenario results in an exception due to mismatch of balances

8.1 Scenario 1

Production

- Pmt Hold condition exists on an Account
- Payment applied, payment goes into PAYMENT_ERROR
- Pmt Hold removed on the same day
- Payment reapplied from Payment Maintenance > Suspense screen. Payment successfully applied to the account as a PAYMENT_NONCASH

Test

- Removal of Pmt hold is replicated on Test

- Payment will be applied successfully to the Account as a PAYMENT.
- No PAYMENT_ERROR is created

8.2 Scenario 2 – This scenario results in an exception due to mismatch of balances

Production

- Pmt Hold condition exists on an Account
- Payment applied, payment goes into PAYMENT_ERROR
- Pmt Hold removed on the same day
- Payment reapplied from Payment Maintenance > Suspense screen. Payment successfully applied to the account as a PAYMENT_NONCASH
- Payment Hold condition reapplied

Test

- Since the final state of the account is a PMT HOLD condition no change will be made to the account in test
- Consequently the payment will go into a PAYMENT_ERROR and will not get applied to the account

In this scenario the account balances DO NOT match.

Phone Pay Fee

Currently in Production One time Phone Pays created during the day are picked up by the SET-CBT-ACH0 > ACCOUNT ACH PROCESSING JOB in the middle of the day (3:15 pm). This job creates the Phone Pay fee (FPHP) (The payment batch is created by the same job). The payments, which post later in the day, get allocated to the FPHP.

On the test environment, the tool posts transactions including Phone pay payments for the entire day. The SET-CBT-ACH0 > ACCOUNT ACH PROCESSING JOB runs AFTER the tool posting is complete. When the payment hits the account, the FPHP is not present on the account. This results in: -

$7 - additional Principal (ADV) being paid

$.01 – less interest being accrued since additional principal has been paid

9
Testing Tool Test Cases

Table 9-1 Testing Tool Test Cases

Module	Function	Sl. No	Test cases
Origination		1	Direct loan
	App Entry	2	Indirect loan
		3	With CRB pull
		4	Without CRB pull
		5	Application in different stages: New Review Required, Auto Approved, Approved Blank, Approved Verifying, Approved Verified
	Funding	6	Loan with Insurances
		7	Loan without Insurances
		8	Loan with Dealer's commission
		9	PreCompute loan
		10	Simple interest loan
Customer Service		11	Back dated payment posting across billing for Simple Interest Loan.
	Payments	12	Back dated payment reversal across billing for Simple Interest Loan Payment reversal from Payment Maintenance screen.
		13	Back dated payment reversal across billing for Simple Interest Loan Already existing payment. Payment reversal from Payment Maintenance screen.
		14	Back dated payment posting across billing for PreCompute Loan.
		15	Back dated payment reversal across billing for PreCompute Loan Payment reversal from Payment Maintenance screen.
		16	Back dated payment reversal across billing for PreCompute Loan Already existing payment. Payment reversal from Payment Maintenance screen.
		17	Overpayment to an account within tolerance.
		18	Overpayment to an account outside tolerance.
		19	Erroneous Payment Batch posting.
		20	ACH Payment Batch posting.

Table 9-1 (Cont.) Testing Tool Test Cases

Module	Function	Sl. No	Test cases
		21	Post backdated payment across billing and NSF the same. NSF from Payment Maintenance screen.
		22	NSF already existing payment. NSF from Payment Maintenance screen.
		23	Create a Payment Batch and Hold the same.
		24	Erroneous Payment Batch posting. Correct the same erroneous transaction and repost successfully.
			Already existing erroneous Payment Batch posting. Correct the same erroneous transaction: Add one row Update one row for date as well as for amount and repost successfully.
			Already existing Open/Hold payment batch: Add one row Remove one row Update one row for date as well as for amount and repost successfully.
		25	Create open Payment Batch and posting the same by running SET-LBT: PAYMENT POSTING
	Payments Combinations	26	Create sequence of Payment Batch & post the same, having mix of valid and erroneous batches. Few erroneous payment batches not corrected and reposted and few corrected and reposted.
		27	Post payments thru lock box with the ach file having: • non-existing a/c no • closed a/c • a/c having payment hold condition • a/c having non-accrual condition, and txn date for the payment is before non-accrual Correct the erroneous payment batches (suspense payments) and repost it. Reverse the payments: • posted correctly first time • corrected and reposted
		28	Post the already existing hold payment batch. Reverse the payment.

Table 9-1 (Cont.) Testing Tool Test Cases

Module	Function	Sl. No	Test cases
		29	Create an erroneous payment batch. Correct the payment batch. Put it on hold. Post the payment batch. Reverse the payment.
		30	Create a payment batch with: • more than one payments • correct and incorrect payments Post the payment batch. Reverse payments: • one which was valid initially • one which was invalid initially
		31	Crete payment batches with: • more than one payments • correct and incorrect payment batches Run the SET-LBT batch job. Reverse the payments: • one which was valid initially • one which was invalid initially
	Transactions	32	PreCompute loan put into non-performing.
		33	PreCompute loan to Simple Interest conversion.
		34	Simple Interest loan put into non-performing.
		35	Work order Service Expense assessment.
		36	Post Extension.
		37	Post Due Date Change.
		38	Charge off an account.
		39	Pay off an account.
		40	Insurance cancellation on a PreCompute loan a/c.
		41	Insurance cancellation on a Simple Interest loan a/c.
		42	Waive Late Charge on an a/c.
		43	Waive Advance on an a/c.
		44	Waive Interest on an a/c.
		45	Reduce Interest Rebate for PreCompute loan a/c.
		46	A/c Monetary maintenance on an a/c to change the advance, rate, payment amount and maturity dates.
		47	Account Due Paid Amount Maintenance posted on an a/c to update the amount paid in due buckets 1 through 4

Table 9-1 (Cont.) Testing Tool Test Cases

Module	Function	Sl. No	Test cases
		48	Post any of the non-monetary transaction.
		49	Reverse any of the already existing monetary transaction.
		50	Post and reverse any of the monetary transaction.
		51	Reverse the charge off transaction for an a/c already charged off.
		52	Post and reverse the charge off transaction.
		53	Reverse the paid off transaction for an a/c already paid off.
		54	Post and reverse the paid off transaction.
		55	Post a monetary erroneous transaction from maintenance screen.
		56	Post a non-monetary erroneous transaction from maintenance screen.
		57	Post an erroneous monetary transaction for which we get the error result in the results pane, correct the same and repost successfully.
		58	Post an erroneous monetary transaction for which we get the popup error message, correct the same and repost successfully.
		59	Void any monetary transaction.
		60	Void any non-monetary transaction.
	Transactions Combinations	61	Put a PreCompute loan into non-performing. Convert the same to Simple Interest. Put the converted loan into non-performing.

Table 9-1 (Cont.) Testing Tool Test Cases

Module	Function	Sl. No	Test cases
		62	Sequence of monetary transactions from maintenance screen, having mix of: • erroneous monetary transaction, for which we get the error result in the results pane • successful non-monetary transactions • erroneous monetary transaction, for which we get the error result in the results pane, corrected and reposted successfully • erroneous transactions for which we get the popup error message • erroneous transactions for which we get the popup error message, corrected and reposted successfully • void monetary transactions • successful monetary transactions
	Full Run	63	• Fund a new backdated loan from scratch. • Post payments. • Post monetary transactions. • Post non-monetary transactions.

10
For Next Releases

Though the Testing Tool is totally dynamic and automates the tests by logging all actions executed on a Production environment, then replicating them in a Test environment, which will have a higher version of OFSLL. This tool still requires technical people, with technical knowledge of OFSLL processes and architecture to be installed and run. For future releases, we can create an installation package and a user friendly front end in order to users to be able to do all the tests independently. Also, some of the current exceptions and limitations could be worked on in order to incorporate to the tool's functionalities.

Glossary