Upgrade Toolkit User Guide

Oracle FLEXCUBE Universal Banking

Release 12.87.04.0.0

Part No. E92710-01

January 2018

ORACLE

Upgrade Toolkit User Guide
January 2018
Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/

Contents

1. Preface

1.1 Introduction

Customers who use the lower versions of Oracle FLEXCUBE Universal Banking Solutions may need to upgrade to the latest version as the support to the older versions will phase out.

This document guides you through the standard strategy for the upgrade activity. In this document, you can find necessary information required to carry out the upgrade activity from any lower version of Oracle FLEXCUBE Universal Banking Solutions to the latest version.

1.2 Intended Audience

This document is intended for the following audience:

- Implementation team
- Partners

1.3 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

1.4 Scope

This document covers the upgrade strategy/methodology for migration from lower version of Oracle FLEXCUBE Universal Banking Solutions (6.2.1.2) to the latest version (12.0.1).

Note

Upgrade from versions FCC6.3, FCC6.4 and FCC6.5 are not in the scope as the target versions may not have all the features of the source version.

1.5 Organization

This manual is organized into the following chapters:

Chapter 1	*Preface* gives information on the intended audience. It also lists the various chapters covered in this User Manual.
Chapter 2	*Upgrade and Conversion Approach* gives an outline of the upgrade and conversion approaches. This also provides a summary of all the activities involved in the entire upgrade process.
Chapter 3	*Mock Upgrade* discusses the prerequisites and guides you through the process of mock upgrade of Oracle FLEXCUBE Universal Banking Solutions from a lower version to higher version.
Chapter 4	*Module Upgrade* discusses the data migration methods specific to Oracle FLEXCUBE Universal Banking Solutions modules, which is part of the mock upgrade activity.

Chapter 5	*Cut-over Upgrade Activities* explains the activities that you need to carry out during cut-over.
Chapter 6	*Conversion Script Generation Tool* discusses the method to use the Dynamic Script Generation tool, which makes the target database compatible with the front end application deployed.
Chapter 7	Data Reconciliation explains the use of Upgrade Reconciliation Tool, which compares the data on the source and target versions after migration and after running a parallel EOD.
Chapter 8	*Annexure* provides the details of utility scripts and conversion scripts used in the upgrade process.
Chapter 9	*Glossary of Scripts* lists out the scripts and links to the places in the document wherever they are used.

1.6 Related Information Sources

For more information, refer to the following documents:

- Oracle FLEXCUBE Installation Manual
- Oracle FLEXCUBE User Manuals

2. Upgrade and Conversion Approach

2.1 Introduction

This chapter gives an outline of the upgrade and conversion approaches. This also provides a summary of all the activities involved in the entire upgrade process.

2.2 Approach - Data Import to Target Schema

In this approach, you will prepare the staging area only with the target version schema. This is the recommended approach.

The steps involved in this method are as follows:

1. Prepare the staging area with the target version schema.
2. Insert the E-data (factory shipped static data) alone in the target schema using the consolidated insert scripts. You can do this using the Oracle FLEXCUBE Universal Banking Installer. The basic setup step should not be done through the installer.
3. Disable the Not null and check constraints in the target schema. Disable all the triggers except the module specific triggers required to be enabled for the upgrade activity. *For the list of triggers to be enabled, See "Module Upgrade" on page 4-1.*
4. Perform data import from the export dump of the production data.
5. Apply the conversion scripts for the columns with the not null and check constraints in the target schema to populate them with proper business data.
6. Enable the constraints in the target schema.

The target schema will now act as the gold copy for the customer to resume the verification/ production activities.

2.2.1 Advantages

The advantages of this approach are as follows:

- Only one staging area prepared for the target version schema
- The production p-data (transaction data) is imported without loss of data as constraints are disabled during the import

2.2.2 Disadvantages

The disadvantages of this approach are as follows:

- While enabling the constraints which are disabled earlier during the process, there might be a few columns that violate the constraints. You need to manually handle this.

2.3 Upgrade Process Summary

A summary of the version upgrade process is provided below.

1. Complete the mock upgrade activity. The mock upgrade provides a safe platform for the actual production environment upgrade. Mock upgrade involves the following steps: (*See "Upgrading Database" on page 3-3.*)
   - Identify the source database schema. The source system should be at the stage of TI just marked on migration date.

- Setup the target database as per the installation manuals. Exit the installer immediately after loading the static data.
- Retrofit the customization changes, if any, into the target database. Apply the DB object changes, source file changes and static data changes.
- Take a full dump of the source schema.
- In the target schema, disable all the triggers and selective constraints.
- In the target schema, selectively enable the triggers as per the module wise approach.
- Import the table data alone into the target database from the full dump of source schema.
- In target schema, apply the dynamic conversion scripts. This makes the target database compatible with the front end application deployed. (*See "Conversion Script Generation Tool" on page 6-1.*)
- In the target schema, enable all the triggers and constraints.
- In target schema, proceed with LD to CL migration and then LM to ELCM migration. Complete the module wise migration and verification checks (*See "Module Upgrade" on page 4-1.*)
- In the target schema, complete the verification activities (*See "Verifying Data after Database Upgrade" on page 3-7.*)

2. Complete the cut-over upgrade activity. During this, you will upgrade the production environment database. (*See "Cut-over Upgrade Activities" on page 5-1.*)
   - During cut-over upgrade, if conversion script needs to be applied on any specific units, generate the required conversion script and apply it on the target schema. (*See "Conversion Script Generation Tool" on page 6-1.*)
3. Reconcile the data in the source and target databases using the data reconciliation tool. Run a parallel EOD and reconcile the data again. (*See "Data Reconciliation" on page 7-1.*)

3. Mock Upgrade

3.1 Introduction

This chapter discusses the prerequisites and guides you through the process of upgrading Oracle FLEXCUBE Universal Banking Solutions from a lower version to higher version.

The upgrade involves the following two activities:

- Mock upgrade activity
- Cut-over upgrade activity

The mock upgrade activity provides a safe platform for the actual production environment upgrade. Once the mock upgrade is completed, you will have a ready target database which is termed as the Gold Copy for setting up the upgraded production environment.

3.2 Prerequisites

Following are the prerequisites for the upgrade activity:

1. Prepare a copy of the production system covering all components of the product for mock upgrade.
2. Set up the Oracle DB parameters as per the FCUBS recommendations for the destination schema. It is ideal to follow the steps of installation of a new version.
3. Identify and list out the installed components of the production system.
4. List out new components available in the new version that the customer proposes to use.
5. Identify and list out the details missed out in the new version. Internally discuss and suggest the actions proposed to address them.
6. Update the customer about the proposed plan and get the customer's concurrence.
7. Following are the other interfacing teams to be involved in discussions for qualification:
   - For qualifying with the new version: Address all changes required for qualification with the new version.
   - For qualifying with the existing interfacing system: Identify and address the new interfacing requirements for the interfacing system to remain intact.
8. You need to understand the database upgrade strategy proposed below:
   - Identify and document the migration steps that are planned. Identify whether any module migrations are present and collate the migration scripts in the migration area (like module migration from LD to CL, LM to ELCM, etc.).
   - Set up the utilities for data comparison and data migration, if any, in the migration area.
   - Identify and document the verification strategy.
   - Prepare the staging area for both source schema and target schema (staging area for source schema is required only if the strategy followed mandates it; otherwise it is not required).
   - Identify the conversion scripts to be applied post-upgrade.
9. Prepare a plan with timeline considering all changes required for a smooth upgrade.

3.3 Mock Upgrade Activity

The mock upgrade activity provides a safe platform for the actual production environment upgrade. You need to prepare a test area where the mock activity can be carried out.

During mock activity, you need to perform user acceptance testing (UAT) for the new modules and the functionality that are added in the higher versions. While performing the actual migration, you need to take the maintenances and parameterizations done in UAT to the production environment.

The target database after the mock upgrade serves as a Gold Copy for you to set up the upgraded production environment.

You can truncate the p-Data tables from the Gold Copy and re-import from the production area. In the time between starting mock run activity and starting the actual production upgrade activity, if any of the static data is changed, then you need to handle such data manually.

Mock upgrade involves the following steps:

- Upgrade Oracle database to the new version
- Identify temp soft changes to and take appropriate action to setup target version with temp soft changes (temp soft changes refer to the customization changes and bug fixed on the source version)
- Setup target version of Oracle FLEXCUBE Universal banking Solutions
- Perform module specific changes if any
- Set up interfaces and adapters
- After upgrade, test the target version along with all interfaces and adapters
- Get sign off on the production environment upgrade

3.3.1 Applying Temp Soft Changes

Temp soft changes refer to the customization changes and bug fixes that are applied on the source version of the application used by the customer. You need to identify the temp soft changes that should be applied in the target version.

Source File Changes

You can use DIFF tools to compare the base version of the source application and the version used by the customer.

Static Data Changes

For identifying the differences in the factory shipped data, use utility/scripts mentioned in the Annexure.

3.3.2 Setting up Target Schema

You need to set up the target schema. For the purpose of illustration, let us consider a schema by name ‘DESTSCHEMA’.

You can use the Oracle FLEXCUBE Universal Banking Installer to set up the target schema. Follow the steps given below.

1. Create the target version database using the target version Installer. Refer to the installation manual of the required version for details on setting up database.
2. Load the static data using installer. Refer to the installation manual of the required version for details on loading static data.

3. Exit the installer immediately after loading the static data. The basic setup step should not be done through the installer.
4. At this point all data structures will be in place and static data tables will have the data populated as of the target version. But all schema objects like the source packages, triggers, procedures, functions, constraints, indexes, views, sequences, etc. would be available as of the base Kernel version.
5. If there are any customization changes that needs to retro-fitted in the target version schema, you may compile them now. You can also make the related static data changes. While doing the TEMPSOFT changes, you need to take care of the following:
   - If the source version had an additional column with data, you need to manually move the same as the import of data from production has already been done.
   - Apply the additional static data onto the upgraded schema.
6. You need to create a dummy schema in the same oracle instance as that of target schema. The dummy schema will have the same name as that of the source schema (from which dump was exported). Provide necessary grants for import/export. This is necessary to connect and import data later on from the dump.

3.3.3 Upgrading Database

The activities involved in database upgrade are given in the table below:

Activity No.	Activity Details	Source/ Destination	Dependency
1	• For illustration purpose, consider that the name of the source schema used by the customer is 'SOURCESCHEMA'. This contains the production data of the bank and the complete set of DB objects including tables, constraints, index sequences, source packages, triggers, etc. • Disable the running Oracle jobs, if any. • Create a full schema dump using the 'expdp' utility in the SOURCESCHEMA. Name the export dump file as 'SITE_FULL_DUMP.DMP'. The parameter file 'Export_Site_FULL_Dump.par' can be used for this export (*See "Annexure" on page 8-1*).	Source	No Dependency. You can do this while the target schema is setup.
2	• Configure the TNS in source and destination database to create DB link.	Common	

Activity No.	Activity Details	Source/ Destination	Dependency
3	• Run the schema difference utility (*See "Annexure" on page 8-1*). This utility lists out the schema differences for the Tables and Columns. • Run 'Create_DB_Link.sql' in the destination schema. It will prompt for the site schema name, password and database name. Upon providing the details, MIG_DB database link will be created connecting source schema. • In case creating a DB link to the production schema is disallowed, a staging area can be created and the DB link can be created to point to the same. • Run 'TableDiff_Source_Dest.sql' utility to identify the table difference between the SOURCESCHEMA and DESTSCHEMA. Copy the results to an Excel file. • Run 'Existing_Table_Column_Diff.sql' to identify the Table Column difference between the SOURCESCHEMA and DESTSCHEMA. Copy the spooled result to Excel file. • This Excel file will act as a reference point of the schema differences between source DB and target DB. • This file has the column level information and details like whether null values are allowed or not. For all the 'not null' columns that are newly introduced in the target version, you need to handle the data import with special consideration because the import for these tables will fail if the records are present in the SOURCESCHMA for the same. • Based on the column differences, generate the scripts to disable the constraints for the new not null columns in the DESTSCHEMA, Along with this, generate the scripts to disable all the triggers. • Use the stub 'Constraint_Trigger_Disable_Script.sql' (*See "Annexure" on page 8-1*) to generate the following scripts. – ALTER_TRIGGER_DISABLE.sql - This sql contains the scripts to disable all the triggers. – ALTER_CONSTRAINTS_DISABLE.sql - This sql contains the script to disable only the not null, unique constraints and check constraints for a column without default value. • Execute the above two scripts before importing the table data from site dump to the DESTSCHEMA. • You may need to enable any specific triggers during import as a special case. Certain ELCM triggers need to be enabled during the data import process. *For details on enabling ELCM related triggers, See "Enabling Triggers" on page 4-13*.	Destination	Activity 1 and Activity 2

Activity No.	Activity Details	Source/ Destination	Dependency
4	• Note that we have already created a dummy schema with the same name as the source schema to facilitate impdp command, which is used in the below command. • Import the table data from the site dump using the par file given below: – Data pump import command: IMPDP source_schema_name/pwd@target_instance PARFILE=<parameter file name with path> • The parameter file 'Import_P-M_data.par' can be used to import P Data, M-Data and P-M Data into the DESTSCHEMA (*See "Annexure" on page 8-1*). • The parameter file 'Import_EM_data.par' can be used to import the E-M data into the DESTSCHEMA (*See "Annexure" on page 8-1*).Refer the import log to ensure that all the table data is imported without any error. • If there is any failure in the import, you need to analyse and handle it manually. • While comparing the SOURCESCHEMA and DESTSCHEMA the stub 'Drop_Sequence_Script.sql' generates the drop script for the common sequences. The drop script file name will be 'DROP_SEQUENCES.sql'. Execute this script to drop the common sequences from DESTSCHEMA. • After dropping the sequence, import the sequences from SITE_FULL_DUMP.DMP using the import par file 'Import_Sequence.par'.	Destination	Activity 1, Activity 2 and Activity 3
5	• Ensure that all the triggers and selected constraints are disabled as mentioned in Activity 3. • Generate and apply the module wise conversion scripts, EXCEPT for LD to CL and LM to ELCM migration. (For details on conversion script generation and application, *See "Conversion Script Generation Tool" on page 6-1*). • Enable all the triggers and constraints once the module wise conversion scripts are generated and applied. You need to manually handle the errors encountered while enabling the triggers and constraints. • Once the triggers and constraints are enabled, migrate LD module to CL module and LM module to EL module. • Note that conversion from LD module to CL module and LM module to EL module should be separately handled as they are re-vamped modules. You may carry out these two migration activities after enabling the triggers and constraints. (*See "Upgrade of Revamped Modules" on page 4-1*).	Destination	Activity 4

Activity No.	Activity Details	Source/ Destination	Dependency
6	• The target database is now ready. Carry out the post-import activities provided in the next section. Carry out the post upgrade verification activities. • You need to preserve the scripts applied while carrying out these activities to use them again if required.	Destination	Activity 5

3.3.3.1 Post Import Activities

Once the data import is completed, you need to perform the following post import activities:

- Recompile invalid objects

3.3.3.2 Issues in Data Import using IMPDP Utility

You may encounter any the following issues while importing data using IMPDP utility.

Issue	Problem	Cause	Resolution
Import options not recognized	Some of the import options may not be enabled in the server. One such example is the DATA_OPTIONS clause of the import, which is used in the E-M Data import par file.	Oracle parameter setup	DBA needs to enable the same
Data Import fails because of new indexes	If the value for a column is null in the imported data which is going to be part of an index in the target then the import fails.	The existing column would have been added as part of a newly created unique index in the DESTSCHEMA. So, if the data for this column contains null values then the uniqueness is violated.	Disable the index, do the import, supply values to this column
Data Import fails due to long columns	If a varchar2 column was changed to long column in the higher versions, then the import fails.	IMPDP does not support importing varchar2 columns into long columns. It is given in oracle documentation that long columns are deprecated and not recommended to create tables using long data type. Instead CLOB to be used.	As a workaround, instead of impdp utility, use the imp utility to import the tables affected by this issue.

3.3.4 Deploying Front End Application

For deploying the front end application, follow the steps below:

1. Refer to the installation manual of the required version of the application.
2. Apply the temp soft changes, if any.

3. Ensure that the deployed EAR points to the upgraded database.

3.3.5 Impact on Existing External System Interfaces

If the customer has any external interfaces maintained in the source application, you need to follow the steps below:

1. Communicate any format level changes (GI files, Gateway XSDs) in existing interfaces to the external systems.
2. Communicate the changes in queues configuration, file locations, etc to the external systems
3. Communicate the changes in the tag names of the XSD files which are shared with other systems to the respective external system owners

3.3.6 Verifying Data after Database Upgrade

Once the database is upgraded, you need to do the following verifications:

- System wide data verification of reports and other check points
- Interface testing to check the connectivity
- Module-wise data verification of reports and other check points
- Converted deals testing
- New deals testing
- New product maintenance testing
- Signoff

These verifications are explained in detail under the following headings.

3.3.6.1 System wide Data Verification

This verification includes the following steps.

Generic Checks

generic check includes the following:

- Check the main parameter table ‘cstb_param’ for the parameter values.
- Menu organization is as per the static factory shipped data and handled from the import. ELCM and CL modules are not handled by the script. You need to remove the LD and Limits menu functions and add the ELCM and CL menu functions.
- The bank may need to modify their user roles. You need to take care of the change of user roles for the new modules. A script will introduce all such roles into ALLROLES.
- Unlock and save all modules maintenance such as Products, ICCF Rule, ICCF Class, UDF etc.

EOD and Performance Testing

This verification includes the following:

- Configure the following as part of FCJ configuration:
  - Install Oracle Web cache which is present in the Sizing document
  - Change the Internet Explorer setting as per DBA server sizing document (recommended IE settings)
  - Ensure that onsite changes are not done to introduce ‘no-cache’ in the code
  - Configure realistic user roles against the usual ALLROLES
- Launch basic screens of high volume module sand process them onsite

- Record and review the EOD and EOM timelines before and after upgrade to check if there are any major variances
- Check if all the programmes maintained in the EOD window is run without being skipped
- Check if all aspects of the EOD, i.e. module functionality and reports generation are covered
- Test on a masked dump of the site if it is done offshore

3.3.6.2 Interface Testing to Check Connectivity

As part of this verification, you need to perform the following activities:

- Test incoming and outgoing interfaces, conversion of FLEXML formats to gateway, EMS to JEMS and ATM/POS using the SWIG interface
- Check for all the channels that receive information from Oracle FCUBS

3.3.6.3 Module-wise Data Verification of Reports and other Check Points

See "Module Upgrade" on page 4-1

3.3.6.4 Converted Deals Testing

You need to test the converted deals as follows:

- As part of the upgrade, the system will have new tables as well as new columns in the existing tables. Check the sanctity of the conversion utility and populate the additional fields and tables by testing the converted/migrated data.
- Perform basic life cycle testing for the converted contracts.
- Check the product maintenances and static maintenances for modifications.

3.3.6.5 New Deals Testing

You need to test the new deals as follows:

- Create new contracts on existing products and observe the validation of default values.
- Test the basic life cycle of new deals.

3.3.6.6 New Product Maintenance Testing

Once the upgrade is completed, create a new product in each module.

3.3.6.7 Signoff

Get the customer signoff to go ahead with the upgrade of production environment.

3.3.6.8 Gold Copy

Gold Copy - DB Schema Setup

Once the above activities are completed, you can use the DESTSCHEMA as the Gold Copy to set up the database during production environment upgrade.

Gold Copy - Front End Setup and Interface

Use the latest available executables to set up the various components for production upgrade.

All interface related changes available in various files need to be deployed.

4. Module Upgrade

4.1 Introduction

This chapter discusses the data migration methods specific to Oracle FLEXCUBE Universal Banking Solutions modules. This step is part of the mock upgrade activity. The following points are discussed in detail in this chapter:

- Scope
- Upgrade of revamped modules
- Upgrade of existing modules
- Module-wise verification checkpoints

4.2 Scope

The scope of the Oracle FCUBS module upgrade is discussed below.

Upgrade of Revamped Modules

The Oracle FCUBS modules such as LD and LM in the older versions are revamped and upgraded to CL and ELCM modules. This migration requires the bank's consent and such cases must be handled separately.

As the target version application may refer to a new set of tables, you need to move the data from the old set of tables to the new set.

Upgrade of Existing Modules

In the target version, new tables and columns may have been added as part of functional enhancements. You need to use the module wise conversion scripts to migrate the data from such modules.

You need to separately handle the customization changes done at site. Involve the bank authorities in the discussions and decide whether data migration is required for the tables added as part of customization. Identify such requirements and document that as an addendum to this guide.

For the existing tables, if data conversion scripts are not provided for newly added columns or existing columns, you need to analyze and handle them manually.

4.3 Upgrade of Revamped Modules

Data migration scripts are provided for the following operations:

- Migration from Loans and Deposits (LD) module to Consumer Lending (CL) module: The CL module was introduced instead of LD in the Kernel version FCUBS 7.1.0.0.0.0.2. The module architecture has changed during this.
- Migration from LM module to ELCM module: The ELCM module was introduced instead of LM in Kernel version FCUBS 11.0.0. The module architecture has changed during this.
- Migration from WebBranch module to Retail Teller: Retail Teller module was introduced instead of Web Branch in Kernel version FCUBS 10.3.0.

4.4 Migrating Data from Loans and Deposits to Consumer Lending Module

The loans portion of LD module has been replaced with CL module in the versions later than FCUBS 7.1.0.0.0.0.2. You can use the migration scripts to upload all active loans from LD module to CL module.

This involves the following steps:

- Migrate LD products to CL module
- Migrate LD contracts to CL module
- Understand the migration strategy
- Migrate commitments

You must migrate commitments before loans. For ease of reference, this document first discusses the loans migration strategy and then the commitment migration strategy.

4.4.1 Migrating Products from LD to CL

You cannot migrate the product data from LD module to CL module. You must create the products in CL manually.

4.4.2 Migrating Contracts from LD to CL

The method of migrating LD contracts to CL module is explained below.

Prerequisites for Source

The below set of instructions are applicable for the source environment

You need to first identify the LD contracts which are not eligible for migration. You can identify such contracts using the script ‘Prelim_src_data_chk.sql’. This script is available in the location ‘SOFT\TOOLS\Upgradeto2olkit\Soft\Migration\LD-CL\SCRIPTS\SOURCE’.

Execute the script ‘Prelim_src_data_chk.sql’ to populate the LD contracts which are not eligible for migration.

The following objects needs to be compiled in the source environment for successful execution of this script.

- DDL
- CSTB_LD_CONTRACT_CHECK
- FNC
- FN_LD_CONTRACT_CHECK.FNC

Once you execute the script ‘Prelim_src_data_chk.sql’ in the source environment, it will print the following output:

querying from CSTB_LD_CONTRACT_CHECK...

COUNT(1)

0

query should return ZERO rows....

If the count is greater than zero, then the details of the contracts that are not eligible for migration will be populated in 'CSTB_LD_CONTRACT_CHECK'. You need to scrutinize them.

Note

If you are not able to correct the non-eligible contracts, you need o manually reverse such LD contracts.

Prerequisites for Target

For target version, you need to do the following maintenances:

- Create conversion GLs
- Maintain CL branch parameters and bank parameters
- Do the mapping between LD product code and CL product code
- Do the mapping between LD component name and CL component name
- Replicate the status codes maintained for LD module in CL module
- Temporarily make the UDFs non mandatory before migration
- Temporarily waive the charges
- Temporarily suppress the advices for Book, Init and Disbursement events
- Uncheck the flag 'Liquidated Back Value Schedules' before migration and revert the same to its original value after migration
- Maintain the interest calculation method
- Maintain the Holiday details in line with the source environment for all the branches
- Change 'Manual Disbursement' to 'Auto Disbursement' before migration and revert the same to its original value after migration

Once the above maintenances are done, you can execute the scripts for migration. These scripts will migrate the eligible LD contracts to CL module.

You must execute the following scripts in the order given below:

1. 1_pre-migration-upd.sql
2. 2_ld-comt-extraction.sql
3. 3_commitment-upload.sql
4. 4_ld-loan-extraction.sql
5. postextraction_check_html.sql
6. 5_cl-upload.sql
7. 5.1.post-migration-checks.sql
8. 6_limits-update.sql
9. 7_bills-update.sql
10. 8_post_migration_updates.sql

When contracts should be migrated again, you need to execute the following scripts in the order given below.

1. cl_account_creation_rollback.sql
2. ld_extraction_rollback.sql

Note

Note the following:

- You need to create the CL products manually through front end.
- Characteristics of LD module like GL mapping, tenor, transaction code, holiday period, exchange rate code etc. must be the maintained AS IS in CL. There should not be any deviation.
- Once the migration is completed, irrespective of the Interest period basis for contracts in LD, the corresponding CL accounts will be based on 'Include from' and 'Exclude to'.
- Schedules pertaining to capitalized LD contract in CL would not be reflected after migration. Nevertheless, capitalization will be handled on the schedule due date.

4.4.2.1 General Migration Strategy

The strategy for migration of LD contracts to CL module is given below.

1. LD loans portfolio is migrated to CL module.
2. Only active LD contracts are considered for migration. Inactive, liquidated or reversed contracts will not be migrated.
3. Contract Reference Number in LD module is stored as Alternate Account Number in CL module. This Alternate Account Number can be used for future references.
4. The status of the LD contracts migrated to CL module will be updated as 'M' to indicate that they were migrated contracts. Changes will be done in the batches to ignore all contracts with status 'M'.
5. All the loans migrated to CL module will start with version 1 and event DSBR.
6. The value date of the LD contracts will be the original start date of the CL accounts. Incase the original start date is available in the LD contract, then it will be taken as the original start date of the CL account.
7. Value date of the migrated CL accounts will be determined from the Last Fully Paid Schedule's Due Date (LFPSD) irrespective of the schedule type, payment method, status and interest rates (fixed / floating). This LFPSD will be the Last fully Paid Schedule for Principal/Interest component.

   Example - Case 1

   In the below example, Principal and Interest components are fully paid on April 15, 2012 and Interest is fully paid on May 16, 2012. whereas Principal component for May 16, 2012 is Outstanding. So, the LFPS would be on April 15, 2012.

Contract Reference Number	Component	Due Date	Amount Due	Amount Settled
406LILP10040000Z	INTEREST	1/17/2012	120	120
406LILP10040000Z	PRINCIPAL	1/17/2012	1000	1000
406LILP10040000Z	INTEREST	2/15/2012	120	120
406LILP10040000Z	PRINCIPAL	2/15/2012	1000	1000
406LILP10040000Z	INTEREST	3/15/2012	110	110
406LILP10040000Z	PRINCIPAL	3/15/2012	1000	1000

Contract Reference Number	Component	Due Date	Amount Due	Amount Settled
406LILP10040000Z	INTEREST	4/15/2012	110	110
406LILP10040000Z	PRINCIPAL	4/15/2012	1000	1000
406LILP10040000Z	INTEREST	5/16/2012	100	100
406LILP10040000Z	PRINCIPAL	5/16/2012	1000	0
406LILP10040000Z	INTEREST	6/15/2012	100	
406LILP10040000Z	PRINCIPAL	6/15/2012	1000	0
406LILP10040000Z	INTEREST	7/15/2012	100	
406LILP10040000Z	PRINCIPAL	7/15/2012	1000	0

Example - Case 2

In this example, the Principal component is fully paid on March 30, 2012 and the Interest component is fully paid on April 15, 2012. LFPS in this case would be March 30, 2012.

Contract Reference Number	Component	Due Date	Amount Due	Amount Settled
406LILP10040000Z	INTEREST	1/17/2012	120	120
406LILP10040000Z	INTEREST	2/15/2012	120	120
406LILP10040000Z	INTEREST	3/15/2012	110	110
406LILP10040000Z	PRINCIPAL	3/30/2012	1000	1000
406LILP10040000Z	INTEREST	4/15/2012	110	110
406LILP10040000Z	INTEREST	5/16/2012	100	50
406LILP10040000Z	INTEREST	6/15/2012	100	0
406LILP10040000Z	PRINCIPAL	6/30/2012	1000	0

8. Migration date plays a vital role in this strategy. No operations (payments, amendment, rollover, reversals) are allowed on migrated loans prior to the migration date.
9. The LD outstanding principal amount as on the migration date is taken as ‘Amount Disbursed’ and ‘Amount Financed’ in CL account. Original loan amount has to be stored as a UDF at CL contract level. Contracts with fully paid principal amounts will be migrated with 0.01 as ‘Amount Disbursed’ / ‘Amount Financed’.

   Example - Case 1

   The principal outstanding from the last fully paid schedule (No partial payment after LFPS).

   Loan Date: December 15, 2011

   LFPS: April 15, 2012

Migration Date: June 30, 2012

Contract Reference Number	Component	Due Date	Amount Due	Amount Settled
406LILP10040000X	PRINCIPAL	1/17/2012	6734.310	6734.310
406LILP10040000X	PRINCIPAL	2/15/2012	7394.010	7394.010
406LILP10040000X	PRINCIPAL	3/15/2012	7571.560	7571.560
406LILP10040000X	PRINCIPAL	4/15/2012	7112.080	7112.080 (LFPS)
406LILP10040000X	PRINCIPAL	5/16/2012	7130.450	0.000
406LILP10040000X	PRINCIPAL	6/15/2012	7306.740	0.000
406LILP10040000X	PRINCIPAL	7/15/2012	7325.000	0.000
406LILP10040000X	PRINCIPAL	8/15/2012	7186.670	0.000
406LILP10040000X	PRINCIPAL	9/15/2012	7205.240	0.000
406LILP10040000X	PRINCIPAL	10/17/2012	7068.410	0.000
406LILP10040000X	PRINCIPAL	11/15/2012	7551.820	0.000
406LILP10040000X	PRINCIPAL	12/15/2012	7413.71	0.000 - Maturity
TOTAL			**87000**	**28811.96**

Principal Outstanding = 87000 - 28811.96 = 58188.04

In the above case, since the principal outstanding is 58188.04 on the migration date -

CL Account Disbursement Amount = 58188.04

Amount Financed = 58188.04

Original disbursement amount 87000 has to be stored as UDF at the CL account Level.

To summarize, the least of the last fully paid schedule among the components pertaining to a loan will be considered as the value date of the account.

10. During migration from LD to CL module, the fully paid schedules are not considered. Only the partially paid and unpaid schedules are migrated to CL module. History of the fully paid schedules have to be viewed from LD screens.
11. All overdue schedules (partial/full overdue) for all components are moved to CL with the current outstanding amount as it is without any further calculations. This table is stored in a table and contains the data as is from LD module. This table contains data for all schedules between LFPSD and migration date. Current running schedule will also be present in this table. This will be applicable to outstanding interest, penalty, unpaid principal amount and up-front fees collected.
12. All the future schedules from migration date onwards are calculated automatically in CL module based on the schedule definition at the contract level.
13. When the minimum tenor of a CL product is higher than the tenor of the active pending loans of the corresponding LD contracts, the product definition of CL product has to be modified manually for the migration purpose. You can later update it to the actual tenor. During migration, the minimum tenor will be updated as one day and once the contract

migration is over, the CL products should be updated to its original minimum tenor as in the LD product.

14. Contracts having only bullet schedule for principal and interest components are migrated with the original value date.
15. For a given contract, the following values are migrated from LD to CL module for a given contract

LD (Loans)	CL (Loans)
Contract Reference Number	Alternate Account Number
Last Fully Paid Schedule's Due Date (LFPSD)	Value Date
Book Date	Book date will be the Migration Date
Maturity Date	Maturity Date
Outstanding principal amount as on migration date	Loan Amount
Original Start Date	Original Start Date
Value Date	Original Start Date (if original start date is null in LD)
Number of Schedules	Derived (LD original schedules as on LFPSD)
Frequency	Same as in LD from LFPSD
No of Units	Derived from LFPSD

16. For a given contract, the following details are migrated from LD to CL module:

LD (Loans)	CL (Loans)
Contract	CL Account Upload
Parties	CL Account Parties Upload
Components	CL Component Upload
Schedules	CL Component Schedule Upload
Interest Rates	CL Account UDE Upload
Linkages	CL Linkages Upload
Settlement Details	CL Account Settlement Upload
MIS Details	Migrated as is from LD contract

4.4.2.2 Accounting Strategy

During the migration process, accounting entries will not be passed in CL module during. All the relevant tables, especially GL balances, will be populated from LD to CL module.

GL mapping between CL tables and LD tables should be the same. Otherwise you may notice inconsistencies in the GL balances.

4.4.2.3 Interest and Accrual

The accrued interest is populated in one table which will have CL replica of LD from last fully paid schedule due of LD till the migration date including the current running schedule. This is applicable to outstanding interest, penalty, unpaid principal amount and up-front fees collected.

During the migration process, accounting entries will not be passed in CL module during. All the relevant tables will be populated as they are from LD module starting from the last fully paid schedule till the migration date including the current running schedule. Accruals from the migration date onwards will continue in CL module.

Example

Consider the following details:

Loan Contract: 550AMN208086xxxx

Loan Book Date: January, 25 2012

LFPSD: May 26, 2012

Migration Date: July 20, 2012

Contract Reference Number	Component	Due Date	Amount Due	Accrued Amount	Amount Settled
550AMN208086xxxx	AMN2-INTER	1/26/2012	56.020	56.020	56.020
550AMN208086xxxx	AMN2-INTER	2/28/2012	52.960	52.960	52.960
550AMN208086xxxx	AMN2-INTER	3/28/2012	37.570	37.570	37.570
550AMN208086xxxx	AMN2-INTER	4/26/2012	31.170	31.170	31.170
550AMN208086xxxx	AMN2-INTER	5/26/2012	24.170	24.170	(LFPS) 24.170
550AMN208086xxxx	AMN2-INTER	6/27/2012	17.120	17.120	0.000
550AMN208086xxxx	AMN2-INTER	7/26/2012	55.860	44.56	0.000
Total			**274.870**	**263.57**	**201.890**

In the above example since till 26 May, 2012 the interest has been fully paid by the customer. Accrued interest in above example will be 61.68.

At the time of migration, the system will migrate schedules which are partially paid/unpaid. LD partially paid/unpaid schedule will be migrated and populated in the CL table ‘cltb_mig_account_schedule’ as given below.

Contract Reference Number	Component	Due Date	Amount Due	Accrued Amount	Amount Settled
550AMN208086xxxx	AMN2-INTER	6/27/2011	17.120	17.120	0.000
550AMN208086xxxx	AMN2-INTER	7/26/2012	55.860	44.56	0.000
Total			**72.980**	**61.68**	**0.000**

Note

From the migration date onwards, accrual will continue as per the existing CL calculation.

4.4.2.4 Status Change Rules

Status change rules are applicable to automatic as well as manual status changes.

Automatic Status Change Rules

After migration, irrespective of the status in LD module, all the CL accounts have 'NORM' status.

During migration when the CL account upload happens, the system processes status change, by suppressing the CL status. This action will move the status depending on the status change rule maintained for the CL products after the CL account upload. The status change rule should to be maintained with the same conditions as in the LD module to ensure that the account status will change from 'NORM' to the new status. The status of the LD contract is matched against the changed status of the CL account, to ensure that the status has not changed after migration.

Note

During migration, the status change of CL accounts is strictly based on the rules defined at product level. Statuses that are supposed to be manual must be maintained as 'Manual' at CL Product level in order to avoid incorrect status derivation.

Manual Status Change Rules

There may be loan contracts in the LD modules whose status has been manually changed.Such contracts will be migrated to CL module with the status 'NORM'. Once the migration is completed, you need to manually change the status of these accounts to their respective statuses in LD module. You also need to manually reverse the accounting entries passed during manual status change.

4.4.2.5 Limits Utilization

During CL account migration process, the limit utilization will be disabled. After migration, the ELCM tables will be updated with the new CL account number in place of loans reference number.

After the migration process, the component wise balances, interest rates, schedule definition, UDF fields, calculation method, date fields etc. have to be compared and verified with LD module contracts.

4.4.2.6 Commitments Linkage

You need to update the commitment linkages separately after the migration of both commitment and loans contracts.

4.4.2.7 MIS

During the CL account migration process, the MIS update is disabled. After the migration, the MIS tables will be updated with the new CL account number in place of loans reference number.

4.4.2.8 UDF

You need to associate the user defined fields defined at the LD product level with the CL products. Account level UDFs attached to the LD contracts will be automatically migrated to CL accounts based on the product mapping done in CL. New UDFs has to be created for storing the original loan amount.

4.4.2.9 SMS

The customer needs to define the SMS roles for new function IDs. Since the LD - CL function ID mapping is not always one-one, the bank needs to manually do the new role configuration.

4.4.2.10 Exceptions in Migration Strategy

The exceptions in the migration strategy are as follows:

- If a loan is of type Rule-78, you need to migrate it with its original value date.
- Holiday treatment in CL module is only at the product level. For CL accounts, the holiday treatment is defaulted from the corresponding CL product. For the LD contracts, if the holiday treatment is different from the corresponding CL product, there may be parameter differences in the schedule definition.
- Acquiring unamortized portion of the discounted/premium on charge or fees component in CL side is not supported by automated migration.

4.4.3 Migrating Commitments

Commitment migration strategy is similar to the loans migration strategy except for the derivation of commitment amount and value date.

See "Migrating Data from Loans and Deposits to Consumer Lending Module" on page 4-2 for the migration flow of loans. You can follow the same method for commitment migration except for derivation of commitment amount and value date.

Derivation logic for commitment amount and value date is given below.

Type	Value Date	Amount	Maturity Date	Linking Strategy
Non-revolving	Min value date (active loans outstanding)	Sum (active loans outstanding) + Unutilized as on migration date	Same as original commitment	During CL migrations, utilize the commitment
Revolving	Min value date (active loans outstanding)	Original commitment amount	Same as original commitment	During CL migrations, utilize the commitment

Example

This example explains the derivation of parameters for non-revolving and revolving commitments. Assume that the commitment does not have VAMI events and the loans have simple BOOK and LIQD events.

Non-revolving:

Commitment	Event Date	Utilized	Un-utilized	LD Contract	Event	Amount
C1 (1000)	01-Jan-12	0	1000			
	01-Feb-12	100	900	L1	BOOK	100
	01-Mar-12	300	700	L2	BOOK	200
	01-May-12	300	700	L1	LIQD**	100
	01-Jun-12	150	550	L3	BOOK	150

Contract L1 is liquidated on 01 May, 2012. Hence L1 is not migrated.

C1 is migrated as CX1, L2 is migrated as LX2 and L3 is migrated as LX3.

Migration Date: 15 June, 2012

Commitment Value Date: 01 May, 2012 (Minimum of value date of active loan (L2,L3))

Commitment amount: 350 + 550 = 900 (sum of outstanding loans (L2,L3) + Un-utilized)

Utilization is as follows:

Commitment	Event Date	Utilized	Un-utilized	LD Contract	Event	Amount
CX1 (900)						
	01-Mar-09	0	900			
	01-Mar-09	200	700	LX2	BOOK	200
	01-Jun-09	350	550	LX3	BOOK	150

Revolving:

Commitment	Event Date	Utilized	Un-utilized	LD Contract	Event	Amount
C2 (1000)	01-Jan-12	0	1000			
	01-Feb-12	100	900	L11	BOOK	100
	01-Mar-12	70	930	L11	PAYM	30
	01-Mar-12	270	730	L12	BOOK	200
	01-Apr-12	240	760	L11	PAYM	30
	01-Apr-12	190	810	L12	PAYM	50
	01-May-12	150	850	L11	LIQD**	40
	01-Jun-12	300	700	L13	BOOK	150

Contract L1 is liquidated on 01 May, 2012. Hence L11 is not migrated.

C2 is migrated as CX2, L12 is migrated as LX12 and L13 migrated as LX13.

Migration Date: 15 June, 2012

Commitment Value Date: 01 March, 2012 (Minimum of value date of active loan (L12, L13))

Commitment amount: 1000 (Original amount)

Utilization is as follows:

Commitment	Event Date	Utilized	Un-utilized	LD Contract	Event	Amount
CX2 (1000)						
	01-Mar-12	0	1000			
	01-Mar-12	150	850	LX12	BOOK	150
	01-Jun-12	300	700	LX13	BOOK	150

4.5 Migrating Data from LM Module to ELCM Module

The LM module in the source version may have been revamped as ELCM module in the target version. You need to migrate the LM data to the ELCM module. This involves the following steps:

- Understand the migration approach
- Ensure that the prerequisites are met
- Table mapping
- Enable EL specific triggers
- Migrate data to the target version
- Truncate database, if required

4.5.1 Migration Approach

The migration approach is as follows:

- Before starting LM to ELCM data migration, LD to CL migration must be completed.
- The migration of data from old set of tables to new set of tables is effected through a package. The DDLs corresponding to the package are also available in the shipment media.
- All maintenance data such as liabilities, collateral etc. are migrated by a simple table-to-table movement of data.
- The utilizations migration involves calls to the underlying limits processing packages.
- LM to ELCM migration is done as a last step in the module wise conversion application process.

4.5.2 Prerequisites

The prerequisites for this migration are as follows:

- Complete the ELCM setup in the migration environment. Refer to the installation manuals of the target version for details on this setup.

- Ensure that the CSTB_PARAM values for ELCM related parameters are properly set.
- Compile the POJO jars and ELCM java files as per the ELCM setup document for target version.
- Provide the Java grants in the schema as per ELCM setup document for target version.
- Verify all java objects in the database and ensure that they having valid status.

Note

The prerequisites given above are a few basic checks to be completed. For complete details, refer to the Installation Manuals of the target version.

4.5.3 Enabling Triggers

Before you import the data from LM to ELCM in the target schema, you need to enable certain ELCM related triggers. You can enable the triggers using the script 'ELCM-TriggersEnable.sql'.

See "Annexure" on page 8-1 for details on the location and usage of the SQL file.

As per the migration strategy, by default all the triggers are disabled before the data import. However, the triggers mentioned in this section must be enabled as exceptional cases.

The list of triggers is given below:

Sl. No.	Trigger Name	Type	Event	On Table
1	ELTR_ACC_-CLASS	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	STTM_AC-COUNT_CLASS
2	ELTR_CLT-M_PRODUCT	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	CLTM_PROD-UCT
3	ELTR_CLT-M_PRO-DUCT_UDE	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	CLTM_PRO-DUCT_UDE
4	ELTR_GETM_LIA-B_CUST_01	BEFORE EACH ROW	INSERT OR UPDATE	GETM_LIA-B_CUST
5	ELTR_LDTB_-CON-TRACT_MASTER	AFTER EACH ROW	INSERT OR UPDATE	CSTB_CON-TRACT
6	ELTR_PRODUCT	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	CSTM_PROD-UCT
7	ELTR_STT-M_CUSTOMER	BEFORE EACH ROW	INSERT OR UPDATE	STTM_CUS-TOMER
8	ELTR_STT-M_CUST_AC	AFTER EACH ROW	INSERT OR UPDATE	STT-M_CUST_AC-COUNT

4.5.4 Migrating Data

You need to migrate the data from LM to ELCM module in the target version. Follow the steps given below:

1. Ensure that the triggers related to ELCM are enabled.
2. Use the package 'ELPKS_LM_REPLICATION' to migrate LM data to GE (ELCM module) tables. You can use the stub 'LM_EL_MIG_Stub.sql' to call the package.

   See "Annexure" on page 8-1 for details on the location and usage of the LM_EL_MIG_STUB.sql.
3. Use the function 'fn_process' for migrating maintenance entities and utilizations. You may migrate the entities either one-by-one or all in one stretch. The parameters to the function are as follows:
   - p_user_id IN VARCHAR2,
   - p_ref_no IN OUT VARCHAR2,
   - p_function_id IN VARCHAR2,
   - p_process_no IN NUMBER,
   - p_remarks IN VARCHAR2,
   - p_errs IN OUT VARCHAR2,
   - p_prms IN OUT VARCHAR2
4. In the above list, the parameter 'p_function_id' is a mandatory parameter. You may leave the other parameters blank. Remarks, if passed, will be used for updating the log tables.
5. The tables 'eltb_migration_log' and 'eltb_mig_exception_log' are the log tables populated during the migration process.
6. You can control the behaviour of the migration using function ID input parameter. If you pass the value 'ALL' as the input, then all liabilities, maintenance entities and utilizations will be migrated in one step. Or we can pass individual function IDs to migrate each entity one-by-one. The functions IDs are listed below:
   - GEDMLIAB - Liabilities Maintenance
   - GEDCULIK - Liability-Customer Link
   - GEDCOLTY - Collateral Types Maintenance
   - GEDSTYPE - Static Types Maintenance
   - GEDCOLCA - Collateral Categories Maintenance
   - GEDISSUR - Issuer codes Maintenance
   - GEDSECTY - Securities Maintenance
   - GEDCOLLT - Collaterals Maintenance
   - GEDCOLTD - TD Collaterals
   - GEDMPOOL - Pool Collateral Linkages
   - GEDFACLT - Facilities Maintenance
   - GEDTRANS - Facilities Transger
   - GEDBLOCK - Facilities Block
   - GEDTRKEXP - Country wise limits
   - GEDUTILUPD - Utilization Upload
   - GEDUTILMIG - Utilization Migration

4.5.5 Truncating Database

During upgrade, you may need to iterate the migration of LM to ELCM module. Before an iterative migration, the EL tables in target database can be truncated to re-run the whole process.

You can use the script 'ELCM_TRUNCATE.sql' to truncate.

See "Annexure" on page 8-1 for details on the location and usage of ELCM_TRUNCATE.sql.

Note

Partial truncation or partial re-migration is not supported.

4.6 Migrating Data from Branch to Retail Teller

In Oracle FCUBS 10.0 version, the branch related data has been moved to new set of tables. Conversion or upgrade scripts are not provided for migration of branch data into retail teller.

At the time of upgrade, the implementation team needs to ensure that outstanding transactions are not pending to be posted to the account at the time of cut-over in any of the web-branches.

The clearing checks which are yet to be paid will come as part of the host-data-migration. That will be available in the upgraded system.

Amendment or reversal an old transaction, which was entered before upgrade, is not supported in the new system after upgrade.

You can setup the new Retail-Branch.

For details, refer to the chapter 'Data Replication' in 'Savings' user manual.

Note

Note the following:

- The table 'FBTB_TCDENM' will be replicated when you maintain the data in the screen 'Confirm Receipts' (IVDCONFR).
- The table 'FBTB_TCDENM_DESC' will be replicated through the screen 'Denomination Details' (CSDDEMAN).

4.7 ATM/POS Modules Impact

For handling ATM/POS transactions in Oracle FCUBS, 'Switch' module was introduced in version 10.3. From this version a totally new set of tables are used. Migration scripts are not provided for the upgrade.

At the time of cut-over, all the transactions should have been posted to the accounts from the Switches. You need to ensure that there is no pending transaction in the Switches.

Reversal of a transaction entered in the source version is not supported in the new system after upgrade.

Maintenance table of ATM/POS terminals may have huge data. The bank may want to migrate such data to the new system.

4.8 Upgrading Existing Modules

You need to upgrade the modules that are not revamped. The conversion scripts for new tables, columns and functional enhancements are available in the 'Conversion Script Repository' (*See "Conversion Script Generation Tool" on page 6-1*).

Scripts are available for the following modules for upgrade from a lower versions to a higher version.

- Core
  - ST
  - CIF
  - CASA
  - IC
  - IS
  - MS
- SMS
- BC
- FA
- FT
- LC
- CD
- MM
- SI

4.8.1 Generic Conversion Methods

The generic conversion methods are discussed under the following headings.

Note

If the source data has account numbers which are in mixed case alphabet or contain characters that are considered to be invalid in target version, then you need to change the respective RAD XML property. You must manually uncheck the uppercase property in RAD.

4.8.1.1 Node Update

Node field is updated with the new database name in different tables across modules. The script for node update is available in the conversion script repository.

4.8.1.2 Basic Parameters Setup

As per the upgrade strategy, E-Data tables will not get imported into target schema. The basic application level parameters (CSTB_PARAM table) is of type E-Data. It cannot be imported to the target version.

You can setup the basic parameters through the following methods:

- Directly through Oracle FLEXCUBE Universal Banking Installer interface - at the time of installation, you can setup the parameters directly in the user interface screen popped up by the Installer
- Load static data - you can use ‘Load Static Data’ option provided by the Installer
- Manual execution - you can use the INC script available for the table CSTB_PARAM in the VERCON area

The implementation team should verify the parameter values wherever parameters are set up.

4.8.2 Upgrading Core Module

Conversion scripts are provided for various tables related to GL, ST, CIF, CASA, IC, IS and MS modules.

See “Conversion Script Generation Tool” on page 6-1.

Import the source data into target schema and apply the conversion scripts referred above. Once this step is completed, you need to do some maintenances for the imported data to work in the target environment.

In case of CIF screen, note that the data in the maintenance tables of the LOV fields (option lists) like ‘Prefix’, ‘Customer Category’ etc. may have mixed case text. But in the new setup of Oracle FCUBS (12.0 and higher), such data is expected to be created in upper case. FCUBS 12.0.2.0.0 accepts mixed case characters during modification of the records.

4.8.2.1 Limitations

Following are the limitations with regard to account creation:

- Location type was free text in older version, but is an LOV field in the later versions. It has to be populated before saving the customer information.
- Invalid characters check for external account had been introduced, which might prevent the account being saved.

Conversion scripts are not provided for the above two cases.

4.8.3 Upgrading SMS Module

4.8.3.1 Password History

During migration, if the source version is older than Oracle FCUBS 11.3, you should not move the password history from source to target.

You can control this by updating the column ‘DATA_IMPORT_REQD’ in ‘CVTM_TABLE_TYPES’ to 'N' where TABLE_NAME ='SMTB_PASSWORD_HISTORY'.

4.8.3.2 Password Reset

Once the data upgrade activities are completed and the front end application is setup, you may login to the new system. However, you need to reset the password.

Using the Oracle FLEXCUBE Universal Banking Installer, you can create two users with password.

Refer to the Installation Manual ‘User Creation Utility’ for details.

You can login to new system with the user IDs created. Once logged in, you can reset the password for any or all the users. You can use the 'User Credentials Change' (SMDCHPWD) screen to reset the password.

You can directly specify the password in the field only if it is enabled. Otherwise, you will notice the option 'Reset Password' as enabled.

Refer to the user manual 'Security Management System' for details on this screen.

4.8.4 Upgrading Deposits Module

The existing deposits in the source version will be created based on the 'Deposit' type of products as part of LD (Loans and Deposits) module. These deposits will be migrated as 'Corporate Deposits' and can be handled through the CD (Corporate Deposits) module.

If required, you need to configure new TD products in the target version (12.0.0 or higher) to make use of the TD functionality.

4.8.5 Dynamic Package Generation for IC Rule

A stub is provided to generate the rule based package dynamically. For each rule in IC module, a package, which is necessary for the functioning of IC module, is generated.

The package 'icpks_gen_new.PR_GEN_RULE' is called for each IC rule defined.

4.8.6 Dynamic Package Generation for Products in CD/MM

A stub is provided to generate a dynamic package for each of the migrated products in CD and MM modules.

The package 'Ldpkss_Status_Rule_Gen.Fn_Gen_Rule' is called for all CD and MM products.

4.8.7 Upgrading PC Module

4.8.7.1 Dynamic Package Generation for Products in PC Module

A stub is provided to generate a dynamic package for each of the migrated products in PC module. This package will help in resolving the rule set at product level for charges calculation.

The package 'PCPKS_CHG_CALC.Fn_Create_Body' is called for all PC products.

4.8.7.2 Bank Clearing Network Maintenance

In lower versions of Oracle FLEXCUBE the bank network clearing maintenance was not mandatory. However, it is mandatory in the higher versions.

Once the migration process is complete, before you start with the operations, maintenances need to be done for the required 'Bank Code-Network ID' combinations.

4.8.8 LC Module - Tracers Generation

If the batch 'LCTRACER' is maintained as part of EOD/BOD batch, make sure that the advices are properly maintained for TRGN event of the product. Otherwise, LC tracer will fail due to non-maintenance of advices for TRGN event.

4.8.9 Upgrading CASA Module - Lower Case Alphabets in Account Number

In the higher versions of Oracle FCUBS, only upper case alphabets are allowed in account numbers. However, in lower versions which needs to be upgraded, in the existing database the account number fields may have alphabets in both upper and lower cases. You need to handle such cases separately.

To handle such situations, after upgrade, you can correct the front end UI file to accept lower case input in the below fields:

- Account Number
- Alternate account number
- Clearing Account Number.
- Master Account Number
- ATM Account Number

You should not set the 'Upper Case' property and 'Restricted Text' property in the RAD XML for the above fields. The RAD XML for these fields is 'STDCUSAC_RAD.XML'.

Once the changes are effected, you need to deploy the UI files and related back end packages. Refer to the Installation Manuals of the target version for details.

4.9 Module Wise Verification Check Points

You need verify the modules to ensure the following:

- Module-wise maintenances - Check whether the module wise maintenances required for the module to function are completed or not. You may unlock or save the products or do backend updates.
- Correctness of sequences
- Correctness of parameter values
- Count of entities before and after migration - Reconciliation scripts are provided for each module to verify the counts of different entities in the database before and after migration.

  See "Data Reconciliation" on page 7-1 for details on reconciliation scripts.

5. Cut-over Upgrade Activities

5.1 Introduction

The upgrade activities that you need to carry out during cut-over are as follows.

- Activities in production environment
- Database upgrade in production environment
- Installation of other components

5.2 Activities in Production Environment

On the cut-over date, the following activities required to be done at the production environment level.

- Run EOC operations and bring the system to TI (Transaction Input) stage of migration date
- Ensure that there are no unauthorized transactions in any module
- Switch off the SWITCH interface, SWIFT and ATM
- Bring down the application server with due notification
- Bring down the Gateway server

5.3 Database Upgrade in Production Environment

In the Gold Copy, truncate the transaction data tables and re-import the same from the latest production data dump. You can use the script files 'Truncate-pData.sql' and 'Export-Source-PData.par' (*See "Annexure" on page 8-1.*) for this.

Ideally the transaction data that has gone into the production data from the time of starting mock run to till time would be the data level change in the source. You need to handled this. However, during the mock run and verification activities the transaction data (p-Data) might have undergone changes. So you must truncate the p-Data tables using the scripts in the Gold Copy.

You need to repeat the database upgrade activity performed during the mock upgrade.

See "Upgrading Database" on page 3-3.

You need not do any static data comparison at this point. The implementation team takes care not to do any static data changes in the production environment.

Selectively apply the post upgrade check points in the upgraded production area.

See "Verifying Data after Database Upgrade" on page 3-7.

In order to make the database consistent and up-to-date, you can reapply the scripts discussed in chapter 'Mock Upgrade Activity'.

See "Verifying Data after Database Upgrade" on page 3-7.

See "Post Import Activities" on page 3-6.

5.4 Installation of Other Components

The Gold Copy should be used for setup in production environment for all applicable components and various files.

6. Conversion Script Generation Tool

6.1 Introduction

You need to apply a set of conversion scripts on the target schema to upgrade the data after the production data is imported. This is necessary to make the target database compatible with the front end application deployed.

The new features introduced in the target version application may necessitate application of some data conversion/upgrade scripts. Apart from this the schematic differences in the database and constraints would necessitate certain scripts to be run in the back end before the application is opened to the Bank's users.

The conversion utility is a set of scripts that includes repository of data upgrade scripts and PL/SQL utility to generate the scripts dynamically to address functional enhancements and the schema differences.

This chapter discusses the method to use the Dynamic Script Generation tool.

6.2 Generating and Executing Scripts

Following steps are involved in the generation and of execution of scripts.

- Setup parameter
- Generate the dynamic scripts and spool the module wise spool files/control file
- Generate the dynamic script for a specific modules
- Generate the dynamic script for a specific script identifier
- Generate the dynamic script for aborted script identifiers
- Spool the module wise spool files and control file for a run number

6.2.1 Setting up Parameters

Set the appropriate values for the parameters in the table ‘CVTB_PARAM’ before generating dynamic scripts. You need to set the following parameters.

- SITE_VERSION: This refers to the Oracle FLEXCUBE version installed at the customer site. The scripts for data migration are picked up based on this parameter as the repository master ‘CVTM_REPOS_MASTER’ contains all the scripts.
- MIGRATION_TYPE: You can generate spool files for different table types depending on the time of data migration. For example, conversion scripts should not be applied for P-data during live cut-over.
- PARSE_STATEMENT: You can have 'Y' or 'N' as the value. If the value is ‘Y’, you can check the correctness of syntax of the dynamic scripts by parsing the statement.
- WORK_AREA: This is the valid folder which is accessible and provided ‘Write’ permission. The module wise spool of the conversion scripts and the control file is generated in this folder.
- RUN_NUMBER: It is used to identify the execution of the scripts end-to-end. if the scripts are already executed and applied onto the database, then you need to change the run_number. Otherwise, the same run_number can be retained.
- GENERATE_SPOOL_FILES: You can generate the scripts and spool the module files and control file by setting the parameter GENERATE_SPOOL_FILES to 'Y'. If it is set to 'N', the scripts are generated, but not spooled.

For scripts to be spooled later, see "Spooling Module-wise Spool Files and Control File for a Run Number" on page 6-3.

6.2.2 Generating Dynamic Scripts and Spooling Files

Before you generate the dynamic scripts, ensure that the data in the parameter table 'cvtb_param' is set as per the requirement.

In order to generate and spool the scripts, execute the stub 'call_cvpks_full_generation.sql' in the SQL prompt. The stub will generate the code for the script_identifier and spool the module wise script files/control file in the folder specified in the 'WORK_AREA' parameter.

6.2.3 Generating Dynamic Script for Specific Modules

You can generate scripts for all the script_identifiers of a one or more specific modules. In order to generate the scripts for specific modules, you need to execute the stub 'call_cvpks_specific_generation.sql'.

The parameters for this file are 'FLEXCUBE', 'M' and the list of modules. The parameter 'M' denotes that it is module specific. You need to provide the list of modules separated by comma as the third argument. The modules will be validated against the maintenance in 'cvtm_module_seq'.

Example 1

The SQL call to generate scripts for the modules BC and SI is as follows:

```
cvpks_dynamic_script_gen.pr_generate_scripts ('FLEXCUBE', 'M',
'BC, SI');
```

In order to execute it in SQL prompt, you need to use the command EXECUTE. If the prompt is SQL>, then the screen will have the following text:

```
SQL> Execute cvpks_dynamic_script_gen.pr_generate_scripts ('FLEX-
CUBE', 'M', 'BC, SI');
```

You can execute the same statement as a PL/SQL block within begin/end as follows:

```
Begin
cvpks_dynamic_script_gen.pr_generate_scripts ('FLEXCUBE', 'M',
'BC, SI');
Exception
When others then
Dbms_output.put_line ('Error :' ||sqlerrm);
End;
```

Example 2

The SQL call to generate scripts for the module CA is as follows:

```
cvpks_dynamic_script_gen.pr_generate_scripts ('FLEXCUBE', 'M',
'CA');
```

For generating spool files, see "Spooling Module-wise Spool Files and Control File for a Run Number" on page 6-3.

6.2.4 Generating Dynamic Script for Specific script_identifier

You can generate scripts for specific script_identifiers. This is done by executing the stub 'CALL_CVPKS_SPECIFIC_GENERATION.SQL'.

The parameters for this file are ‘FLEXCUBE’, 'S' and the list of script identifiers. The parameter ‘S’ denotes that it is script_identifier specific. You need to provide the list of script identifiers separated by comma as the third argument.

Example 1

The SQL call to generate scripts for the script identifiers CA_007, LD_001 and SI_009 is as follows:

```
cvpks_dynamic_script_gen.pr_generate_scripts ('FLEXCUBE', 'S',
'CA_007, LD_001, SI_009');
```

Example 2

The SQL call to generate scripts for MS_009 is as follows:

```
cvpks_dynamic_script_gen.pr_generate_scripts ('FLEXCUBE', 'S',
'MS_009');
```

For generating spool files, see “Spooling Module-wise Spool Files and Control File for a Run Number” on page 6-3.

6.2.5 Generating Dynamic Script for Aborted Script Identifiers

You can regenerate all the scripts that were aborted for a specific run_number. You can do this by executing the stub ‘call_cvpks_specific_generation.sql’.

The parameters for this file are ‘FLEXCUBE’, 'A' and the run number. The parameter ‘A’ denotes that it is for the aborted script identifiers. The run number is the third argument.

Example 1

The SQL call to generate the scripts for aborted script identifiers is as follows:

```
cvpks_dynamic_script_gen.pr_generate_scripts ('FLEXCUBE', 'A',
2);
```

For generating spool files, see “Spooling Module-wise Spool Files and Control File for a Run Number” on page 6-3.

6.2.6 Spooling Module-wise Spool Files and Control File for a Run Number

You can generate module-wise spool files and control file by executing the stub call_cvpks_generate_spools.sql’. This stub creates the spool files for the code blocks that are already generated. The files would be generated in the path maintained in CVTB_PARAM.

Yo can generate the scripts for different run numbers by specifying it in the stub itself.

The input to this stub is source_code (FLEXCUBE by default) and then the run_number.

For every run_number, the stub generates the module-wise spool files and control files separately.

7. Data Reconciliation

7.1 Introduction

Once the data has been migrated from the source version to the target version, you need to reconcile the data. You can use the Upgrade Reconciliation Tool to compare the data on the source and target versions after migration and after running a parallel EOD. After data reconciliation, you can generate the reconciliation reports.

This chapter discusses the method of using upgrade data reconciliation tool. The following points are discussed in connection with reconciliation tool in this chapter:

- Setting up a new environment
- Releasing additional units (delta release)
- Changing the source and target schema in the existing system
- Extraction and report generation
  - Generating migration reconciliation report
  - Generating adhoc reconciliation report
  - Generating parallel run reconciliation reports
- Moving extraction data into history table

7.2 Setting Up New Environment

When reconciliation tool is setup in a fresh environment, you need to follow the steps given below.

1. Run the recon tool sources (dll, inc, spc,s ql, vw) in the source schema and target schema.
2. In the source schema, complete the following activities:
   - Update the source schema name, password and SID name in the script '1a_db_link_src.sql' and run the script
   - Run '1b_tab_recon_script_gen_src.sql'
   - Run '2a-validate_recon_scripts_src.sql'
   - Run '2b-update_inv_scripts_src.sql'
   - Run '1d_populate_mapping_tables_scr.sql'
3. In the target schema, complete the following activities:
   - Update the target schema name, password and SID name in the script '1a_db_link_tar.sql' and run the script
   - Run '1b_tab_recon_script_gen_tar.sql'
   - Run '2a-validate_recon_scripts_tar.sql'
   - Run '2b-update_inv_scripts_tar.sql'
   - Run '1d_populate_mapping_tables_tar.sql'
4. Check the following parameters in the 'CVTB_PARAM TABLE':

Parameter	Description
RECON_MODULE_LIST	Module list in Tilda separated values. This list will be taken, if the module code has been passed as 'ALL' during data extraction.

Parameter	Description
RECON_REPORT_PATH	Path where the Recon reports needs to be generated.
TARGET_LM_INSTALLED	Target LM module installed (it can be LM or EL)
SOURCE_LM_INSTALLED	Source LM module installed (it can be LM or EL)
TARGET_LOAN_INSTALLED	Target Loans module installed (it can be LD or CL)
SOURCE_LOAN_INSTALLED	Source Loans module installed (it can be LD or CL)
RECON_ENVIRONMENT	This is the environment Name. It will be appended as part of the report file name.

7.3 Releasing Additional Units - Delta Release

In case DDL files are released, it needs to be applied in both source schema and target schema.

In case the INC files are released, you need to perform the following activities:

1. Apply ‘CVTB_PARAM’ table related INCs in both the source schema and the target schema
2. Apply ‘CVTM_RECON_DYN_SCRIPTS’ table related INCs only in the target schema
3. Apply ‘CVTM_RECON_REPORTS’ table related INCs only in the target schema
4. In the source schema, complete the following activities:
   - Update the source schema name, password and SID name in the script ‘1a_db_link_src.sql’ and run the script.
   - Run ‘1b_tab_recon_script_gen_src.sql’
   - Run ‘2a-validate_recon_scripts_src.sql’
   - Run ‘2b-update_inv_scripts_src.sql’
5. Follow the below steps in Target System:
   - Update the target schema name, password and SID name in the script ‘1a_db_link_tar.sql’ and run the script
   - Run ‘1b_tab_recon_script_gen_tar.sql’
   - Run ‘2a-validate_recon_scripts_tar.sql’
   - Run ‘2b-update_inv_scripts_tar.sql’

In case spc, sql, vw (cvpks_recon_extract.sql, cvpks_recon_extract.spc) are released, you need to execute them in both the source schema and the target schema.

7.4 Changing Source and Target Schema in Existing System

When source schema and target schema in the existing system are changed, you need to follow the steps given below.

1. In the source schema, complete the following activities:
   - Update the source schema name, password and SID name in the script ‘1a_db_link_src.sql’ and run the script
   - Run ‘1b_tab_recon_script_gen_src.sql’

- Run ‘2a-validate_recon_scripts_src.sql’
- Run ‘2b-update_inv_scripts_src.sql’
- Run ‘1d_populate_mapping_tables_scr.sql’

2. In the target system, complete the following activities:
   - Update the target schema name, password and SID name in the script ‘1a_db_link_tar.sql’ and run the script
   - Run ‘1b_tab_recon_script_gen_tar.sql’
   - Run ‘2a-validate_recon_scripts_tar.sql’
   - Run ‘2b-update_inv_scripts_tar.sql’
   - Run ‘1d_populate_mapping_tables_tar.sql’
3. Check the following parameters in the ‘CVTB_PARAM TABLE’:
4. RECON_MODULE_LIST Module list in Tilda separated values. This list will be taken, if the module code has been passed as ‘ALL’ during data extraction)

Parameter	Remarks
RECON_REPORT_PATH	Path where the Recon reports needs to be generated.
TARGET_LM_INSTALLED	Target LM module installed (it can be LM or EL)
SOURCE_LM_INSTALLED	Source LM module installed (it can be LM or EL)
TARGET_LOAN_INSTALLED	Target Loans module installed (it can be LD or CL)
SOURCE_LOAN_INSTALLED	Source Loans module installed (it can be LD or CL)
RECON_ENVIRONMENT	This is the environment name. It will be appended as part of the report file name.

7.5 Generating Reports

You can generate various reports related to reconciliation tool. This section discusses the methods to generate the following reports:

- Migration Reconciliation Report
- Adhoc Reconciliation Report
- Parallel Run Reconciliation Report

7.5.1 Generating Migration Reconciliation Report

Once the source data is migrated to the target version environment, you can generate the migration reconciliation report. This is a complete reconciliation report and covers all the entities that need to be reconciled. For generating the migration recon report, you need to follow the steps given below:

1. Check the following details:
   - Head office branch for the source schema and target schema must be the same
   - The ‘Today’ column in ‘sttm_dates’ table should be same for all the branches in the source schema and the target schema
   - The report generation path available in ‘CSTB_PARAM TABLE’, PARAM_NAME: ‘RECON_REPORT_PATH’.
   - The recon extraction modules available in ‘CSTB_PARAM TABLE’, PARAM_NAME: ‘RECON_MODULE_LIST’.

2. In the source schema, complete the following activities:
   - Run ‘1b_tab_recon_script_gen_src.sql’
   - Run ‘2a-validate_recon_scripts_src.sql’
   - Run ‘2b-update_inv_scripts_src.sql’
3. Follow the below steps in Target System:
   - Run ‘1b_tab_recon_script_gen_tar.sql’
   - Run ‘2a-validate_recon_scripts_tar.sql’
   - Run ‘2b-update_inv_scripts_tar.sql’
4. In the source schema, complete the following activity:
   - Run ‘3_recon-migrt_src.sql’ with parameter ‘BRANCH_CODE’ as head office branch. In normal cases it is CHO.
5. In the target schema, complete the following activities:
   - Run ‘3_recon-migrt_tar.sql’ with parameter BRANCH_CODE as head office branch. In normal cases it is CHO.
   - Run ‘5_recon-reportgen_migrt.sql’ with parameter BRANCH_CODE as head office branch. In normal cases it is CHO.

7.5.2 Generating Adhoc Reconciliation Report

You can generate the adhoc reconciliation report for individual entities that you need to verify. For generating this report, you need to follow the steps given below:

1. Before you start the report generation, check the following:
   - Head office branch for the source schema and the target schema are the same.
   - The ‘Today’ column in ‘sttm_dates’ table should be the same for all the branches in source and target system.
   - The report generation path available in ‘CSTB_PARAM TABLE’, PARAM_NAME: ‘RECON_REPORT_PATH’.
   - The recon extraction modules available in ‘CSTB_PARAM TABLE’, PARAM_NAME: ‘RECON_MODULE_LIST’.
   - Get the list of entities which needs to be part of the adhoc report generation (module code, entities) and prepare the below insert statement.

     insert into cvtb_recon_adhoc_entity(module_code,entity) values ('PC','PC_PERIODIC_INSTRUCTIONS');
2. In the target schema, complete the following activities:
   - Run insert statements prepared in the previous step
   - Run commit
3. In the source schema, complete the following activity:
   - Run ‘4c_recon-parl_adhoc_src.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.
4. Follow the below steps in Target System:

Run ‘4c_recon-parl_adhoc_tar.sql’ with Parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.

Run ‘6e_recon-reportgen_parl_adhoc.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.

7.5.3 Generating Parallel Run Reconciliation Report

Once the data is migrated, you need to run EOD batch on both the source and the target environments at the same time. You can check specific entities and mark for parallel run. The parallel run reconciliation report provides the details of data reconciliation after the parallel EOD batch.

For generating this report, you need to follow the steps given below:

1. Check the following:
   - Head office branch for the source schema and the target schema are the same.
   - The branch for which Recon is planned to be executed
   - The ‘Today’ column in ‘sttm_dates’ table should be the same for the branch in source and target system, for which the report is generated.
   - The stage during which the recon is planned to be executed. It can be ‘MarkEOTI’ or ‘PostBOD’
   - The Oracle FLEXCUBE logical stage is the same for source schema and target schema
   - The report generation path available in ‘CSTB_PARAM TABLE’, PARAM_NAME: ‘RECON_REPORT_PATH’
   - The recon extraction modules available in ‘CSTB_PARAM TABLE’, PARAM_NAME: ‘RECON_MODULE_LIST’
2. In the source schema, complete the following activities:
   - Run ‘1b_tab_recon_script_gen_src.sql’
   - Run ‘2a-validate_recon_scripts_src.sql’
   - Run ‘2b-update_inv_scripts_src.sql’
3. In the target schema, complete the following activities:
   - Run ‘1b_tab_recon_script_gen_tar.sql’
   - Run ‘2a-validate_recon_scripts_tar.sql’
   - Run ‘2b-update_inv_scripts_tar.sql’

At this stage, you need to consider two instances of parallel run at MarkEOTI stage and PostBOD stage.

Case A: Parallel Run at MarkEOTI Stage

4. In the source schema, complete the following activity:
   - Run ‘4a_recon-parl_preod_src.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.
5. In the target schema, complete the following activities:
   - Run ‘4a_recon-parl_preod_tar.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.
   - Run ‘46a_recon-reportgen_parl_preod.sql’ with parameter BRANCH_CODE as ‘ALL’ For all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.

Case B: Parallel Run at PostBOD Stage

4. In the source schema, complete the following activity:
   - Run ‘4b_recon-parl_pseod_src.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.
5. In the target schema, complete the following activities:
   - Run ‘4b_recon-parl_pseod_tar.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.
   - Run ‘6b_recon-reportgen_parl_pseod.sql’ with parameter BRANCH_CODE as ‘ALL’ for all branch extraction. For a specific branch, the BRANCH_CODE parameter needs to be the specific branch itself.
6. Check the following parameters in the CVTB_PARAM TABLE:

Parameter	Remarks
RECON_MODULE_LIST	Module list in Tilda separated values. This list will be taken if the module code has been passed as ‘ALL’ during data extraction.
RECON_REPORT_PATH	Path where the Recon reports needs to be generated.
TARGET_LM_INSTALLED	Target LM module installed (it can be LM or EL).
SOURCE_LM_INSTALLED	Source LM module installed (it can be LM or EL).
TARGET_LOAN_INSTALLED	Target Loans module installed (it can be LD or CL).
SOURCE_LOAN_INSTALLED	Source Loan module installed (it can be LD or CL).
RECON_ENVIRONMENT	This is the environment name. It will be appended as part of the report file name.

7.5.4 Moving Extraction Data into History Table

For moving the extraction data into the history tables, you need to follow the steps given below.

1. Check the following:
   - Head office branch for the source schema and the target schema are the same.
   - The ‘Today’ column in ‘sttm_dates’ table should be the same for all the branches in source and target system.
   - Collect the branch code, stage and extraction date for the extraction data which is being moved into the history table.
2. In the source schema, complete the following activity:
   - Run ‘1c_move_to_history_src.sql’ with parameter branch code, stage and extraction date which has been collected in the previous step.
3. In the target schema, complete the following activity:
   - Run ‘1c_move_to_history_tar.sql’ with parameter branch code, stage and extraction date which has been collected in step 1.

8. Annexure

8.1 Utility Scripts

The utility scripts are given in the following table.

Script Name	Location	Remarks
Export_Site_FULL_Dump.par	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Param files to export the production schema at site. In this par file, replace the word 'SOURCESCHEMA' with the actual schema name to be exported. Also change the 'DIRECTORY', 'DUMPFILE', 'LOGFILE' names as per the actual names used. This is applicable to all the par files supplied in the document.
Create_DB_Link.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Script to create database link. Before creating the DB link, configure the TNS connection between the source and destination database.
TableDiff_Source_Dest.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Scripts to list out the differences between the source schema and target schema.
Existing_Table_Column_Diff.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Script to list out the differences in tables which are existing in both the schemas, but the columns are different.
Constraint_Trigger_Disable_Script.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Script to disable constraints and triggers.
Constraint_Trigger_Enable_Script.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Script to enable constraints and triggers.

Script Name	Location	Remarks
Drop_Sequence_Script.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Script to drop the sequence.
Import_P-M_data.par	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Param files to import static data.
Import_EM_data.par	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Param files to import non-static data.
Import_Sequence.par	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Par file to import sequence.
Truncate_PData.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Scripts to truncate p-Data.
Export_Source_PData.par	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Param file to export required p_data tables.
ELCM_Triggers_Enable.sql	/SOFT/TOOLS/ Upgradetoolkit/ Soft/ImpExp/ Scripts	Script to enable ELCM related tables.
ELCM-TriggersEnable.sql	SOFT/TOOLS/ Upgradetoolkit/ Soft/Migration/LM-EL/Scripts	Script to enable ELCM related triggers.
ELCM_TRUNCATE.sql	SOFT/TOOLS/ Upgradetoolkit/ Soft/Migration/LM-EL/Scripts	Script to truncate the ELCM database
LM_EL_MIG_STUB.sql	SOFT/TOOLS/ Upgradetoolkit/ Soft/Migration/LM-EL/Scripts	Script to call the package 'ELPKS_LM_REPLICA-TION' which migrates LM data to GE.