Uploading Records from Upload Table
Oracle FLEXCUBE Universal Banking
Release 14.7.2.0.0
Part No. F87755-01
[November] [2023]

Contents

1. Preface

This document describes standard framework in FLEXCUBE for uploading records from upload tables.

1.1 Audience

This document is intended for FLEXCUBE Application developers/users that use development Workbench to develop various FLEXCUBE components.

To Use this manual, you need conceptual and working knowledge of the below:

Proficiency	*Resources*
FLEXCUBE Functional Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Screen Development	*Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development*
Working knowledge of Web based applications	Self Acquired
Working knowledge of Oracle Database	Oracle Documentations
Working knowledge of PLSQL & SQL Language	Self Acquired
Working knowledge of XML files	Self Acquired

1.2 Related Documents

Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development
Development Workbench - Screen Development II

2. Introduction

2.1 How to use this Guide

The information in this document includes:

3. Overview of Bulk Upload of Records

Bulk upload of Records to FLEXCUBE through upload tables is commonly used for uploading data from an external system periodically.
Data is populated in the upload tables through Macro Excel Upload or any other utility. (Note: Data Population in upload tables should be taken care by custom team .ODT tool does not provide feature for data population)
Thereafter upload routine is invoked from the Screen **CVDUPLOD** for the particular function id. Upload routine processes each record from upload tables. Status of processing will be updated in a **process table** for monitoring purpose.
Upload routine should follow the **same flow** as that of Gateway/FLEXCUBE User Interface to ensure integrity and consistency for records uploaded through different routines.
This necessitates the need for a standard framework for uploading records from upload tables.
A standard framework for the same has been developed using ODT which is described in the sections below.

4. Upload Framework

Upload Framework supports upload of both maintenance and transaction screens.

Different steps involved in Bulk Upload can be enlisted as:

1) Data is populated in the upload tables through Macro Excel Upload or any other utility.
2) ***Trigger on Master Upload Table*** would insert entries into a ***process table*** with Upload Status as 'U' (Unprocessed). One entry would be inserted into ***process table*** for each record. Function id would also be updated in the process table along with other information.
3) Upload routine is invoked for a particular function id by the user from **CVDUPLOD** screen/stub.
4) On invoking the routine, system would process all the unprocessed records from the ***process table*** for the particular function id. This would be done using a cursor on process table
5) An adapter package converts the upload table types to base table type data. Then it invokes the main package of the function id.

6) After processing of each record , ***process table*** columns for uploaded status, error code etc would be updated by the system

From the above steps, we can derive at the components required for a particular function id to be brought under this framework.

1) Process Table
2) Upload Tables
3) Trigger on Master Upload Table
4) Adapter package for Upload Routine
5) Wrapper code in CVDUPLOD screen processing logic to call the adapter package based on the function id

4.1 Naming Convention:

Framework does not enforce a standard naming convention for upload tables. Existing upload tables can be re-used in this framework.
But if any ***new*** upload table is introduced, it is recommended to follow naming convention as illustrated below:
Fourth Letter of base table to be replaced with U
Example:
Base Table Name: STTM_CUSTOMER
Upload Table Name: STTU_CUSTOMER

Recommended to follow naming convention for consulting / client developed
Upload Table Name: Table name _U_EXTGBL

4.2 Process Table

For Uploading, each record is processed from a cursor on process tables.
This is common across all function ids.
There are 2 process tables

- **CSTB_EXT_CONTRACT_STAT** :
  This is the process table for all transaction Function ids.

Name	Type	Nullable
BRANCH_CODE	VARCHAR2(3 CHAR)	N
SOURCE	VARCHAR2(20 CHAR)	N
PRODUCT_CODE	VARCHAR2(4 CHAR)	Y
COUNTERPARTY	VARCHAR2(35 CHAR)	Y
EXTERNAL_INIT_DATE	DATE	Y
MODULE	VARCHAR2(2 CHAR)	Y
EXTERNAL_REF_NO	VARCHAR2(20 CHAR)	N
IMPORT_STATUS	VARCHAR2(1 CHAR)	Y
CITICUBE_REF_NO	VARCHAR2(16 CHAR)	Y
POST_IMPORT_STATUS	CHAR(1 CHAR)	Y
EXPORT_STATUS	CHAR(1 CHAR)	Y
USER_ID	VARCHAR2(12 CHAR)	Y
JOBNO	NUMBER(2)	Y

CONTRACT_REF_NO	VARCHAR2(16 CHAR)	Y
ERR_CODE	VARCHAR2(11 CHAR)	Y
ERR_MESSAGE	VARCHAR2(255 CHAR)	Y
ACTION_CODE	VARCHAR2(10 CHAR)	Y
FUNCTION_ID	VARCHAR2(8 CHAR)	Y
EXTERNAL_SEQ_NO	NUMBER(22)	N
UPLOAD_ID	VARCHAR2(16 CHAR)	Y

Here a particular record from upload Tables would be picked by combination of EXTERNAL_REF_NO, EXTERNAL_SEQ_NO, BRANCH_CODE and SOURCE .
Columns like EXPORT_STATUS, CONTRACT_REF_NO, ERR_CODE and ERR_MESSAGE would be updated by the system after processing.
UPLOAD_ID signifies the thread of execution. Upload routine can be invoked in multiple threads if multiple upload ids are present.

- **STTB_UPLOAD_MASTER** :
  This is the process table for all maintenance Function ids.

Name	Type	Nullable
MAINTENANCE_SEQ_NO	VARCHAR2(16 CHAR)	N
BRANCH_CODE	VARCHAR2(3 CHAR)	N
SOURCE_CODE	VARCHAR2(15 CHAR)	N
MAINTENANCE_TYPE	VARCHAR2(15 CHAR)	Y
UPLOAD_STATUS	CHAR(1 CHAR)	Y
UPLOAD_INITIATION_DATE	DATE	Y
USER_ID	VARCHAR2(12 CHAR)	Y
ACTION_CODE	VARCHAR2(15 CHAR)	Y
SOURCE_SEQ_NO	NUMBER	N
UPLOAD_ID	VARCHAR2(16 CHAR)	Y

Here a particular record from upload Tables would be picked by combination of MAINTENANCE_SEQ_NO, SOURCE_SEQ_NO, BRANCH_CODE and SOURCE_CODE.
UPLOAD_STATUS would be updated by the system after processing a record.
UPLOAD_ID signifies the thread of execution. Upload routine can be invoked in multiple threads if multiple upload ids are present

4.3 Upload Tables

Each Data source in the function id, if required, should be mapped to corresponding Upload Tables in ODT.

- ***Data Source Column Mapping***
  Mapping of Upload Table Columns to Base Table Columns has to be done after proper analysis. Avoid including internal processing columns/invisible field columns etc to upload Table. This will reduce complexity of upload table.

4.3.1 Guide Lines

Some guidelines for mapping upload table/columns with base table/columns are enlisted below:

- Master Data Source of the Function Id should always be a mapped to an Upload Table (except in case of some call forms where it is not feasible). This upload table would be referred to as **Master Upload Table**
- Upload Tables should be mapped only to **Normal** Data Sources as per ODT configuration. For query, in only and summary data sources; upload tables are not required
- More than one data source in the Function Id can be mapped to a single upload table. Note that all the base tables should one to one relationship with each other in this scenario.
  Example: Both CSTB_CONTRACT and FXTB_CONTRACT_MASTER can be mapped to the same upload table, say, FXTB_UPLOAD_MASTER.
- If the master data source is a common table used across many functions (*Example: CSTB_CONTRACT*), try grouping it with any of its child table; so that the master upload table is unique for the function id.
  Example: Both CSTB_CONTRACT and FXTB_CONTRACT_MASTER can be mapped to the same upload table, say, FXTB_UPLOAD_MASTER.
- It is recommended to provide the same column names to both base table column and upload table column. This avoids complexity to both developer and user.
- Apart from Mapped Columns from Base Table, Upload Table should have a standard set of columns as defined below

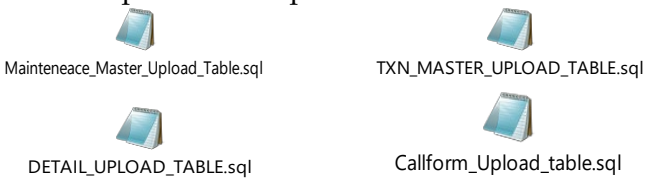
COLUMN NAME	Remarks
SOURCE_CODE	Specifies External Source
SOURCE_REF / MAINTENANCE_SEQ_NO	Specifies External Reference Number. SOURCE_REF is used for Contract upload tables while MAINTENANCE_SEQ_NO for maintenance upload tables
SOURCE_SEQ_NO	Specifies Source Sequence Number.
BRANCH_CODE	Specifies branch Code
FUNCTION_ID	This column is required only in Upload Master table. This is mandatory if same upload master tables are used for multiple function ids. *Example : Parent and Child Functions*
ACTION_CODE	This is required only in Upload Master Table. If not present , then only NEW Operation would be supported by upload framework for the Function Id.
UPLOAD_ID	This is required only in Upload Master Table. Different values can be inserted for this column in batches to process upload routine in multiple threads. This is an optional column; mostly used in transaction screens

UPLOAD_STATUS	Optional. Required only if Upload Table and Master Table are the same.eg: PC contract. This is used mostly in transaction screens
MODULE	Module Code of the Function Id. Optional; used mostly in transaction screens
SOURCE_OPERATION	Optional. Specifies the Source Operation code. This needs to present in only master upload table, if any. If not present , then system would try to derive the default SOURCE_OPERATION for particular action code

SOURCE_CODE, SOURCE_REF / MAINTENANCE_SEQ_NO, SOURCE_SEQ_NO and BRANCH_CODE form the composite primary key for any master upload table. For detail upload tables, the 4 columns mentioned above along with unique identifier for the record, if any, forms the primary key
FUNCTION_ID, ACTION_CODE, UPLOAD_ID, UPLOAD_STATUS, MODULE and SOURCE_OPERATION are optional columns in Master Upload Table

- For Transaction screens, EXTERNAL_REF_NO of upload table has to be mandatorily mapped to a base table column. This is required to derive the reference number in case of any modify operation. Most often, this column can be found in CSTB_CONTRACT.
- More than one data source can be mapped to the same upload table differentiated by ***upload table where clause***.
  For Instance; if two different legs of a transaction (buy and sell) of a deal are captured by two data sources in function id (same table with different aliases); then one upload table can be used for both the tables. Upload where clause for both these data source should be such that the adapter picks proper data to base table data types
  Note that in this scenario, both the data sources should not be directly related to each other. Difference has to be noted between this scenario and the case where 2 data sources with one to one relationship is mapped to same upload table.
- For call form function ids, master data source would often be a view for propagating record key to the call form. In such instances master data sources should not have any upload table mapped to it.
  Example: CSTBS_CONTRACT__ADV is the master data source for Advice Call form but data is uploaded only in CSTB_CONTRACT_EVENT_ADVICE. Hence upload table should not be mapped to CSTBS_CONTRACT__ADV

Sample Master Upload Table definitions are attached:



4.4 Trigger on Upload Table

Triggers would be created on Master Upload Table to insert records into Upload Process Tables on insert of records in upload tables. For Uploading, each record is processed from a cursor on process tables.

4.4.1 Guide Lines

- If column for action code is not present in the master upload table; then Action code column in process table would be updated as NEW
- If upload routine is present for parent and child function ids; then the master upload table would be the same. ***In such cases, FUNCTION_ID column should be present in the master upload table*** and the same would be inserted into the process table. Hence the same trigger would hold good for all the child screens.
- There would be no separate trigger for any call form function ids as call form records does not exist independently.

Sample Upload Table Trigger is attached

Maintenance_Upload_Trigger.sql

Txn_Upload_Trigger.sql

4.5 Upload Adapter Package

Upload Packages would handle type conversions and processing records after conversion.

- **Naming Convention**
  Module||'**_pks**_'||FunctionId||'**_Ext_Upload'**
  Example : fxpks_fxfdtronl_ext_upload

Based on structure, upload packages can be broadly classified as

1) **Transaction Upload Adapter**
   Records will be processed based on cursor on CSTB_EXT_CONTRACT_STAT.
   Code to handle ***SUBSYSTAT*** will be present.
2) **Maintenance Upload Adapter**
   Records will be processed based on cursor on STTB_UPLOAD_MASTER
3) **Transaction Call forms Upload Adapter**
   It will be called from Transaction Upload Package.
   Code to update ***SUBSYSSTAT*** will be present
4) **Maintenance Call Forms Upload Adapter**
   It will be called from Maintenance Upload Packages

5. ODT Capabilities

ODT supports extensible upload framework.
Upload Framework components can be generated by the tool through configurations.

5.1 Configuration of Upload Table Details in RADXML

Upload Table :

In data source definition screen, upload table name for the data source has to be specified.

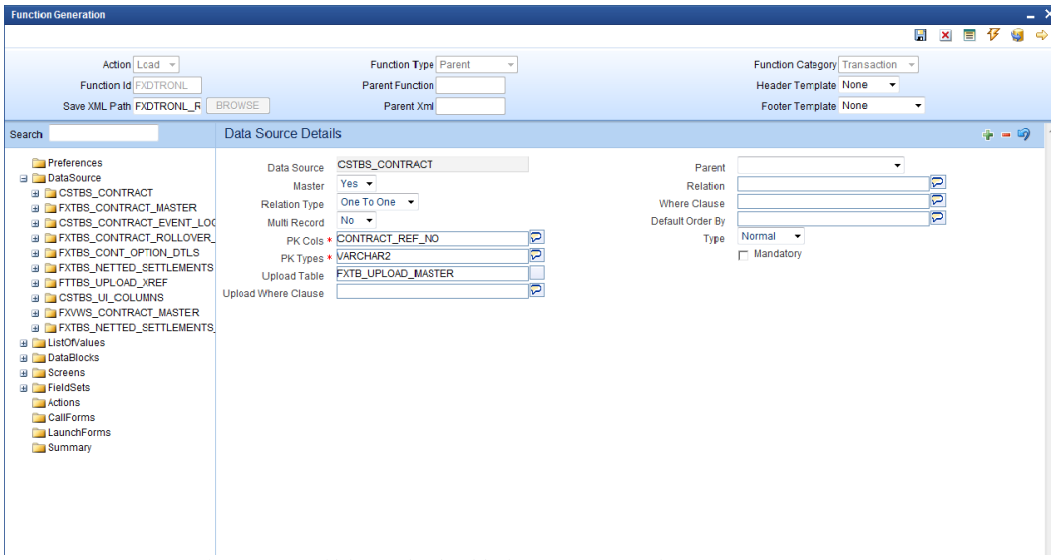


Figure 1: Specifying Upload Table for a Data Source in ODT

- Avoid providing Synonyms in Upload Table field.
- Upload Tables should be mapped only to **Normal** Data Sources
- The standard set of columns for the upload table can be viewed by clicking on the button next to Upload table.
- SOURCE_CODE, SOURCE_REF, SOURCE_SEQ_NO and BRANCH_CODE will be assumed as part of primary key of any upload table.
  Make note of the guide lines explained in previous section while providing Upload Table Name

Upload Table Standard Columns:

Refer previous section for the standard columns which are part of the upload table.

Default column names are provided in the screen. Source Operation would be not present in the table by default.

Developer can change the column names of the standard columns as desired. This could be useful if existing upload tables are re-used.

Example : Name of the column for External reference number can be changed from SOURCE_REF to EXT_REF_NO

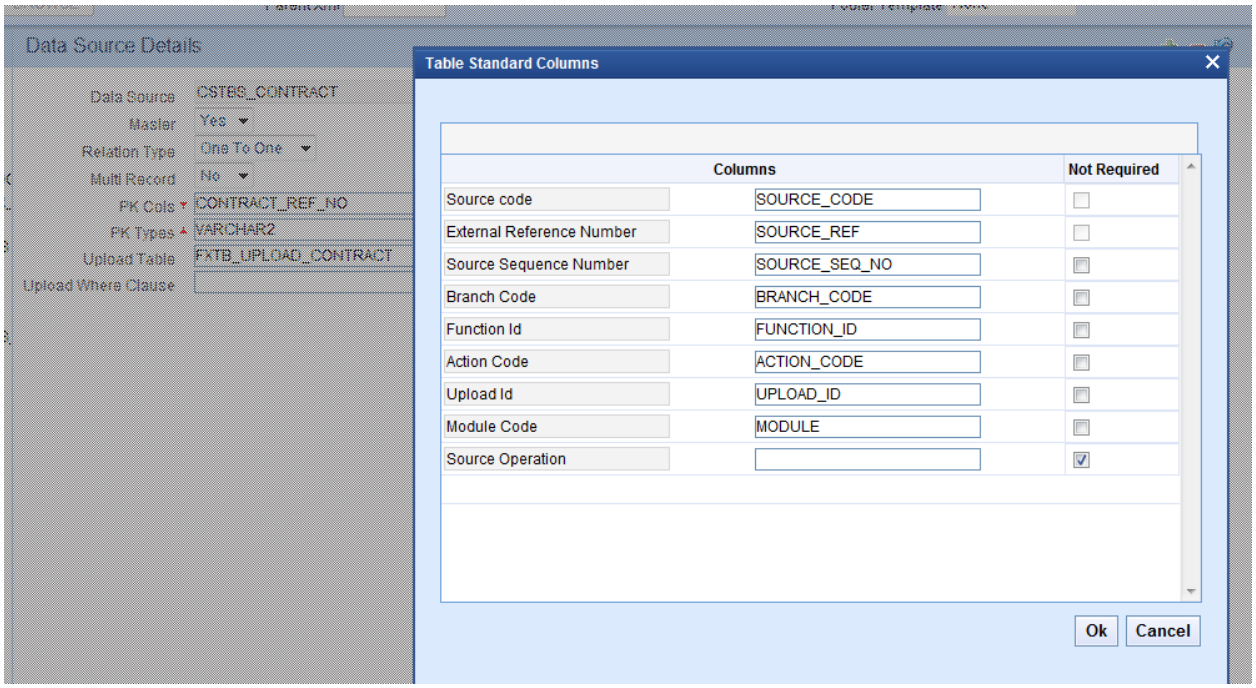


Figure 2: Standard Columns for Master Upload Table of Transaction Screen

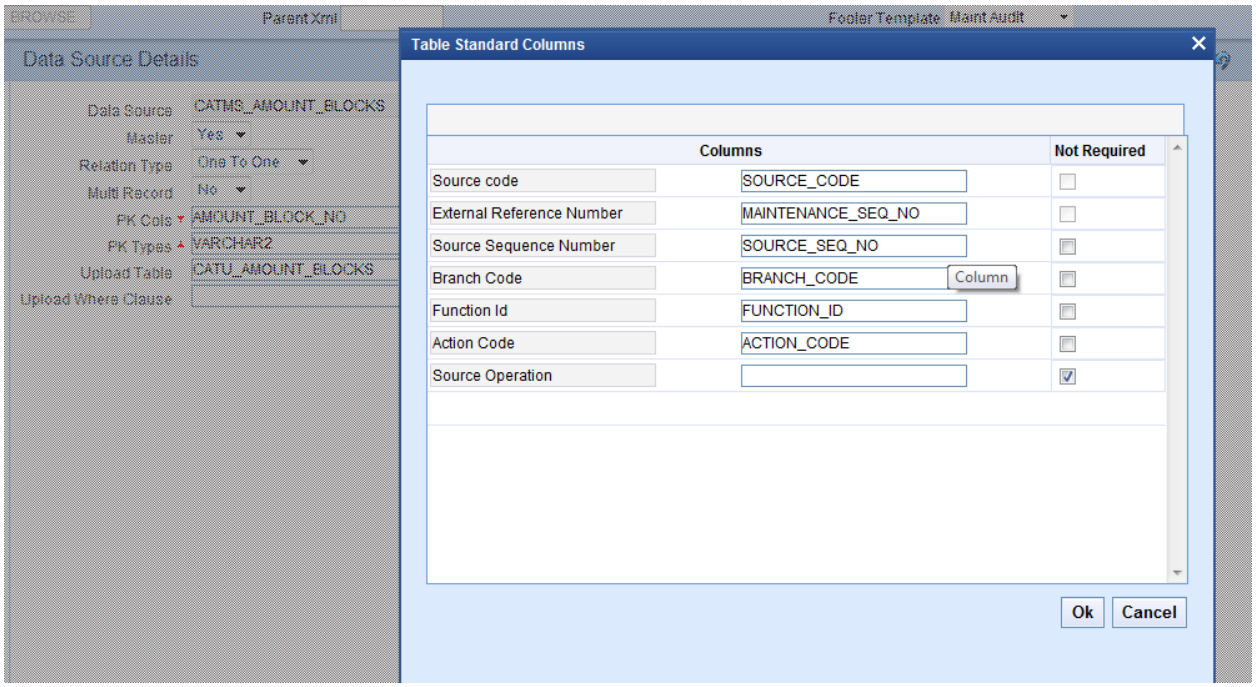


Figure 3: Standard Columns for Master Upload Table of Maintenance Screen

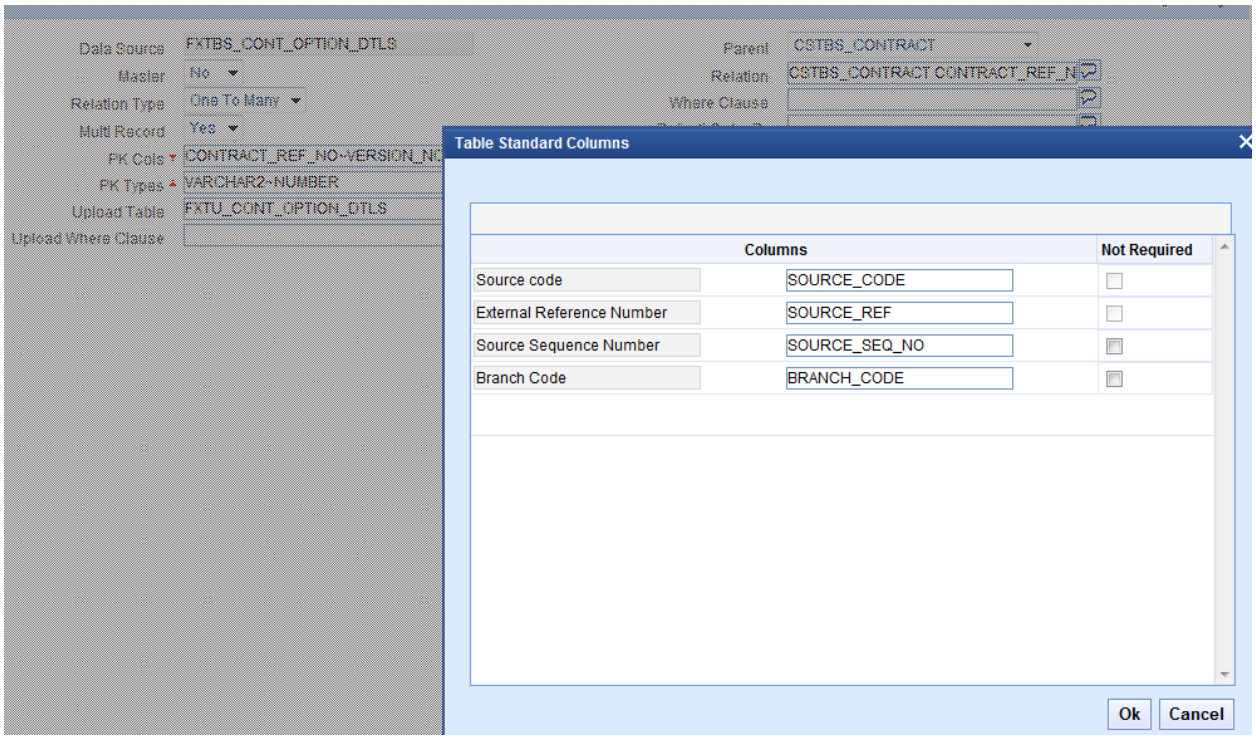


Figure 3: Standard Columns for Detail Upload Table

Upload Table Where Clause:

If all the records in an upload table is not be mapped to a particular data source; then upload table where clause can be specified to filter the records.

This is applicable only when the data sources involved are not directly related with each other.

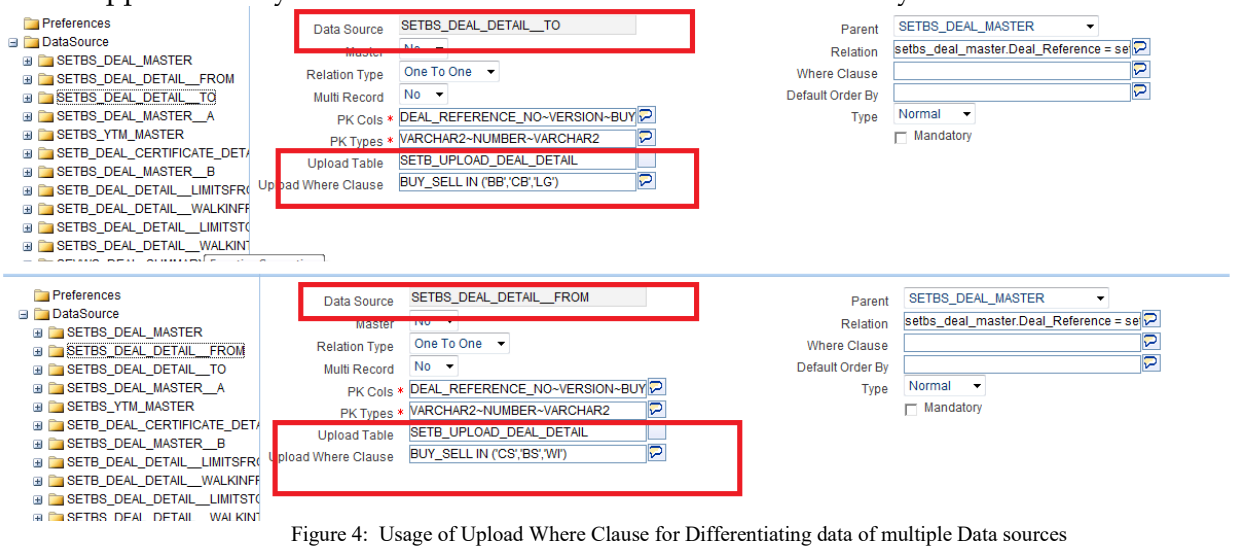


Figure 4: Usage of Upload Where Clause for Differentiating data of multiple Data sources

Upload Table Where clause would be applied on the upload table; hence upload table columns should be used in the clause

Upload Table Column:

All the data source columns which are included in the RADXML would be assumed to be part of the upload table.

- By default, name of the Upload Table column would be assumed to be same as that of the base table name
- If the name of the upload table column has to be different from base table column; then the same has to be explicitly mentioned in Upload Table Column Field

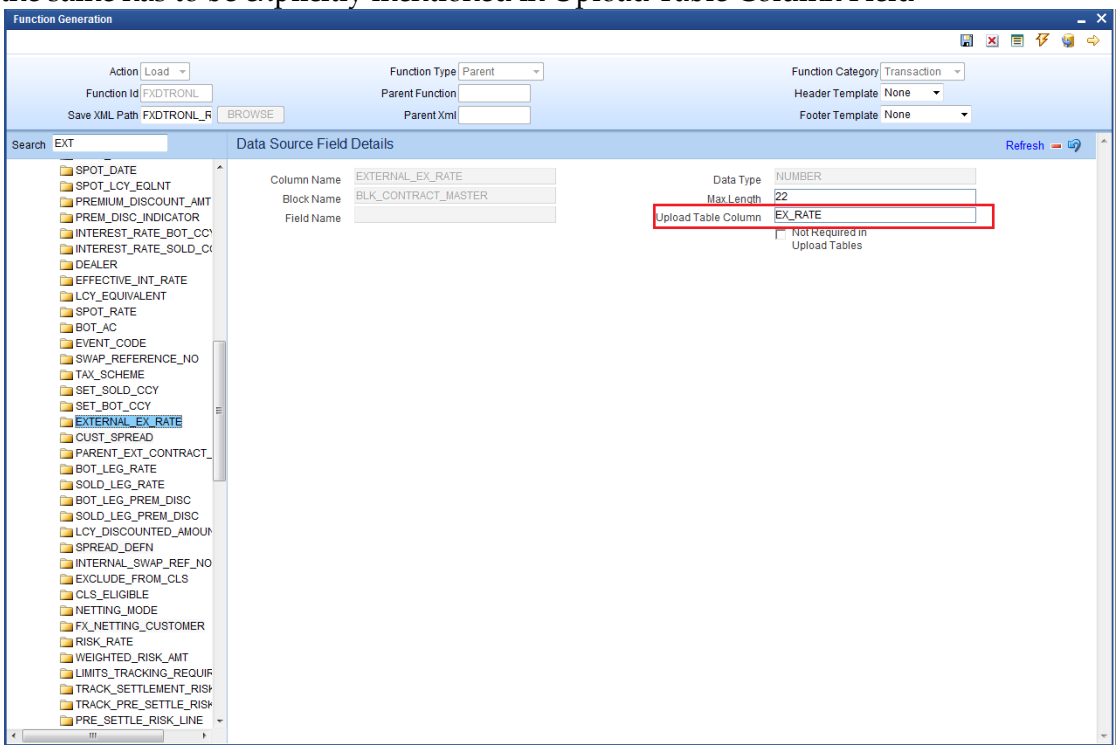


Figure 5: Changing Column Name for Standard Columns in Upload Tables

- If any of the columns included in the data source in RADXML is not required in Upload table; then the same has to specified by selecting the checkbox **Not Required in Upload Tables**

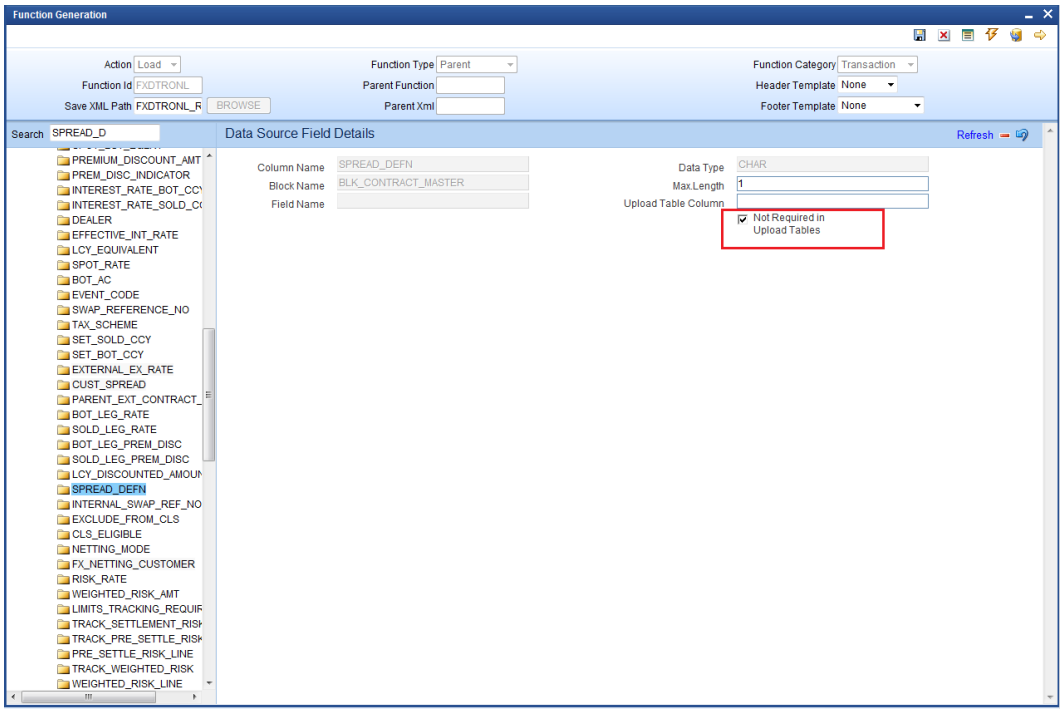


Figure 6: Data Source Column not required in upload Table. Explicitly specifying the same

5.2 Generated Units

1) Upload Adapter Package
   Naming Convention : Module||'**_pks**_'||FunctionId||'**_Ext_Upload'**
2) Triggers on Upload table
3) Upload Table DDL
   If existing upload tables are being used, DDL scripts can be ignored.
   Drop scripts for the table would be generated in a separate file.

5.3 Upgrade Capabilities

Normal ODT upgrade feature is supported in Upload table configurations as well.
Customizations can be done on the configuration maintained in ODT
Customizations can be done to:

i) Change the Upload Tables mapped.
   Map new Upload tables/Remove existing table mapping
ii) Modify/Remove/Add Upload table column names
iii) Configure upload tables for entire screen if it is not provided by engineering and if bank needs the same

Changes done as part of customizations would be retained during ODT Refresh.
Any new mappings done by the engineering team would reflect after Refresh.
After Refresh all the artifacts has to be regenerated including upload table definitions.

6. Miscellaneous

6.1 Appending Data

In certain scenarios, only the data which has to be appended would be uploaded to the upload tables. Requirement would be to append this data to the existing data in the table. This feature is not supported by the standard FLEXCUBE framework

Example : Upload of Floating Rates for a Currency

FLEXCUBE Framework :

In FLEXCUBE, if any multi Record Block has to be modified then the complete data of the block has to be sent. FLEXCUBE will derive the records modified, deleted or added in the block according to the data sent and updates the tables accordingly. ***Hence if only the data to be appended to the block is sent, then the existing records would be treated as deleted and hence deleted from the tables.***

Solution:

To handle this scenario, the following approach is recommended:

Handle this case based on the source operation parameter in the custom package.

Developer can either

1) ***skip all the system functions and write code to upload data in custom package for particular source operation OR***
2) ***append the existing data to screen object instance before start of processing (preferably in pre_check_mandatory) for the particular source operation***

Uploading Records from Upload Tables
[November] [2023]
Version 14.7.2.0.0

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/