Oracle® Banking Corporate Lending

Upload Records From Upload Table

Release 14.8.2.0.0
G53393-01
April 2026

Oracle Banking Corporate Lending Upload Records From Upload Table, Release 14.8.2.0.0

G53393-01

Contents

Preface

This topic contains the following sub-topics:

Purpose

This manual is designed to help acquaint you with the standard framework in Oracle FLEXCUBE Universal Banking for uploading records from upload tables.

Acronyms and Abbreviations

Table 1 Acronyms and Abbreviations

Acronyms	Abbreviations
FCUBS	Oracle FLEXCUBE Universal Banking Solution
OBCL	Oracle Banking Corporate Lending
ODT	Oracle Development Tool

Audience

This document is intended for Oracle FLEXCUBE Universal Banking Application developers/ users that use Development Workbench to develop various Oracle FLEXCUBE Universal Banking components. To use this manual, the user needs a conceptual and working knowledge of the below:

Table 2 Proficiency Details

Proficiency	Resources
Oracle FLEXCUBE Universal Banking Technical Architecture	Training programs from Oracle Financial Software Services.
Working knowledge of Web based applications	Self Acquired
Working knowledge of Oracle Database	Oracle Documentations

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

Conventions

The following text conventions are used in this document:

Table 3 Conventions

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Resources

For more information on any related features, refer to the following documents:

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

Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes.

1

Overview of Bulk Upload of Records

This topic describes the bulk upload of records to Oracle FLEXCUBE Universal Banking.

Bulk upload of records to Oracle FLEXCUBE Universal Banking 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**
>
> The data population in upload tables should be taken care of by the custom team. Open Development Tool tool does not provide a feature for data population.

Thereafter upload routine is processed from the screen **CVDUPLOD** for the particular function ID. Upload routine processes for each record from upload tables. The status of processing will be updated in a process table for monitoring purposes.

The upload routine should follow the same flow as that of Gateway/Oracle FLEXCUBE Universal Banking 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 Open Development Tool which is described in the following topics:

- Upload Framework
  This topic describes a standard framework for uploading records.

1.1 Upload Framework

This topic describes a standard framework for uploading records.

Upload Framework supports the upload of both maintenance and transaction screens. Different steps involved in Bulk Upload are listed below:

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 the 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 processing the routine, the system would process all the unprocessed records from the process table for the particular function ID. This would be done using a cursor on the 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 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

This topic contains the following sub-topics:

1.1.1 Naming Convention

This topic describes a standard framework for uploading records.

The framework does not enforce a standard naming convention for upload tables. Existing upload tables can be re-used in this framework. If any new upload table is introduced, it is recommended to follow the naming convention as mentioned below.

> **Note**
>
> In the naming convention, the fourth letter of the base table is to be replaced with **U**.

For example,

- Base Table Name - **STTM_CUSTOMER**
- Upload Table Name - **STTU_CUSTOMER**

Recommended to follow the naming convention for consulting/client-developed Upload Table Name as **Table name _U_EXTGBL**.

1.1.2 Process Table

This topic describes process table for uploading records.

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**
- **STTB_UPLOAD_MASTER**

Table 1-1 CSTB_EXT_CONTRACT_STAT

Name	Type	Characters	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	NA	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 a 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. The upload routine can be invoked in multiple threads if multiple upload IDs are present.

Table 1-2 STTB_UPLOAD_MASTER

Name	Type	Characters	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	NA	Y
USER_ID	VARCHAR2	12 CHAR	Y
ACTION_CODE	VARCHAR2	15 CHAR	Y
SOURCE_SEQ_NO	NUMBER	NA	N
UPLOAD_ID	VARCHAR2	16 CHAR	Y

Here a particular record from upload tables would be picked by a 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. The upload routine can be invoked in multiple threads if multiple upload ids are present.

1.1.3 Upload Tables

This topic provides an overview of the Upload Table.

Each data source in the function ID must be mapped to corresponding Upload Tables in Open Development Tool.

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 the complexity of the upload table.

Guidelines

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

- The Master Data Source of the function ID should always be mapped to an Upload Table (except in the case of some call forms where it is not feasible). This upload table would be referred to as Master Upload Table.
- Map the Upload Tables to only Normal Data Sources as per Open Development Tool configuration. For query, in only 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**
>
> All the base tables should have one-to-one relationships with each other in this scenario.

  For 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 (For example, **CSTB_CONTRACT**), try grouping it with any of its child tables, so that the master upload table is unique for the Function ID.
  For 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 columns and upload table columns. This avoids complexity for both the developer and the user.
- Apart from mapped columns from Base Table, Upload Table should have a standard set of columns as defined below:

Table 1-3 Standard Column Names

Column Name	Remarks
SOURCE_CODE	This field specifies the external source.
SOURCE_REF / MAINTENANCE_SEQ_NO	This field specifies the external reference number. **SOURCE_REF** is used for Contract upload tables while **MAINTENANCE_SEQ_NO** is for maintenance upload tables.

Table 1-3 (Cont.) Standard Column Names

Column Name	Remarks
SOURCE_SEQ_NO	This field specifies the source sequence number.
BRANCH_CODE	This field specifies the branch code.
FUNCTION_ID	This column is required only in the Upload Master table. This is mandatory if the same upload master tables are used for multiple function IDs. For 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 the 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 the upload routine in multiple threads. This is an optional column, mostly used in transaction screens.
UPLOAD_STATUS	Required only if Upload Table and Master Table are the same. For example, PC contract, This is used mostly in transaction screens. This field is optional.
MODULE	This field displays the module code of the function ID. Used mostly in transaction screens. This field is optional.
SOURCE_OPERATION	This field specifies the Source Operation Code. This needs to be present in only the master upload table if any. If not present, then the system would try to derive the default **SOURCE_OPERATION** for a particular action code. This field is optional.

SOURCE_CODE, **SOURCE_REF / MAINTENANCE_SEQ_NO**, **SOURCE_SEQ_NO**, and **BRANCH_CODE** form the composite primary key for any master upload table. For detailed upload tables, the 4 columns mentioned above along with a unique identifier for the record, if any, form 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, the **EXTERNAL_REF_NO** of the upload table has to be mandatorily mapped to a base table column. This is required to derive the reference number in case of any modified 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 the 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 tables. Upload where clause for both these data sources should be such that the adapter picks proper data to base table data types. Note that in this scenario, both data sources should not be directly related to each other. A difference has to be noted between this scenario and the case where 2 data sources with a one-to-one relationship are mapped to the same upload table.
- For call form function IDs, the master data source would often be a view for propagating the record key to the call form. In such instances, master data sources should not have any upload table mapped to them. For example, **CSTBS_CONTRACT__ADV** is the master data source for the Advice Call form but data is uploaded only in **CSTB_CONTRACT_EVENT_ADVICE**. Hence upload table should not be mapped to **CSTBS_CONTRACT__ADV**.

Refer to the following sample Master Upload Table definitions:

- Mainteneace_Master_Upload_Table.sql

- TXN_Master_Upload_Table.sql
- Detail_Upload_Table.sql
- Callform_Upload_table.sql

1.1.4 Trigger on Upload Table

This topic describes an overview to trigger the 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.

Guidelines

- If the column for **Action** code is not present in the master upload table, then the **Action** code column in process table would be updated as **NEW**.
- If the upload routine is present for parent and child function IDs, then the master upload table would be the same. In such cases, the **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 do not exist independently.

Refer to the sample Upload Table Trigger documents as follows:

- Maintenance_Upload_Trigger.sql
- Txn_Upload_Trigger.sql

1.1.5 Upload Adapter Package

This topic describes overview on upload adapter package for uploading records.

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

- Naming Convention - **Modulepks_FunctionID_Ext_Upload**
  For example, **fxpks_fxfdtronl_ext_upload**

Based on structure, upload packages can be broadly classified as follows:

Table 1-4 Upload Packages

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

2

Open Development Tool Capabilities

This topic describes the Open Development Tool capabilities for an extensible upload framework.

Open Development Tool supports an extensible upload framework. The upload framework components can be generated by the tool through configurations.

This topic contains the following sub-topics:

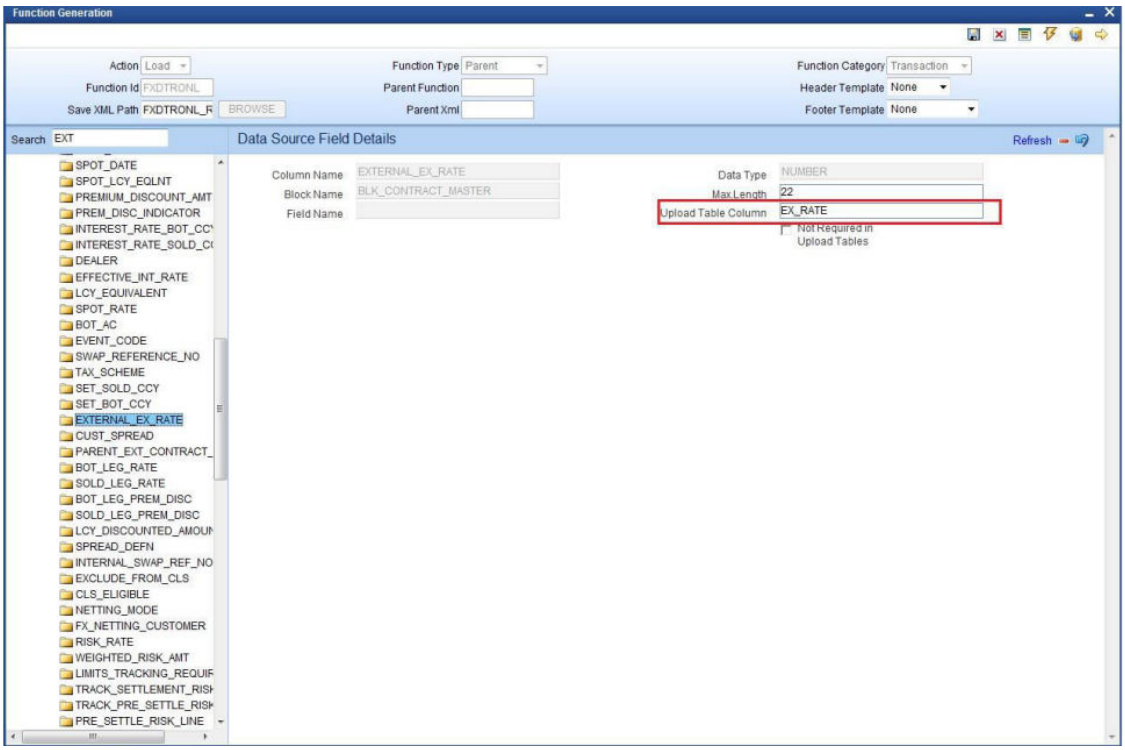
- Configuration of Upload Table Details in RADXML
  This topic describes the Configuration of Upload Table Details in RADXML.
- Generated Units
  This topic describes the detailed information on generated units.
- Upgrade Capabilities
  This topic describes the detailed information on upgrade capabilities.

2.1 Configuration of Upload Table Details in RADXML

This topic describes the Configuration of Upload Table Details in RADXML.

Upload Table - In the data source definition screen, specify the upload table name for the data source.

Figure 2-1 Data Source Field Details – Upload Table Column



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

Upload Table Standard Columns - Refer to the previous section for the standard columns which are part of the upload table. Default column names are provided on the screen. Source Operation would be not present in the table by default. The developer can change the column names of the standard columns as desired. This could be useful if existing upload tables are re-used. For example, Name of the column for external reference number can be changed from **SOURCE_REF** to **EXT_REF_NO**.

Figure 2-2 Table Standard Columns – Transaction Screen

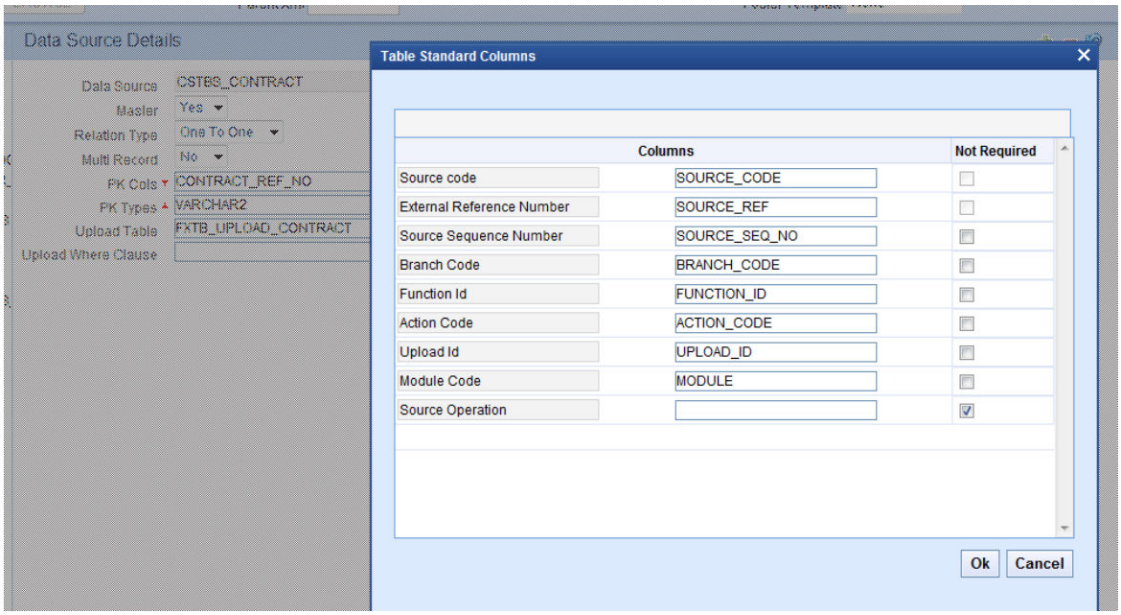


Figure 2-3 Table Standard Columns – Maintenance Screen

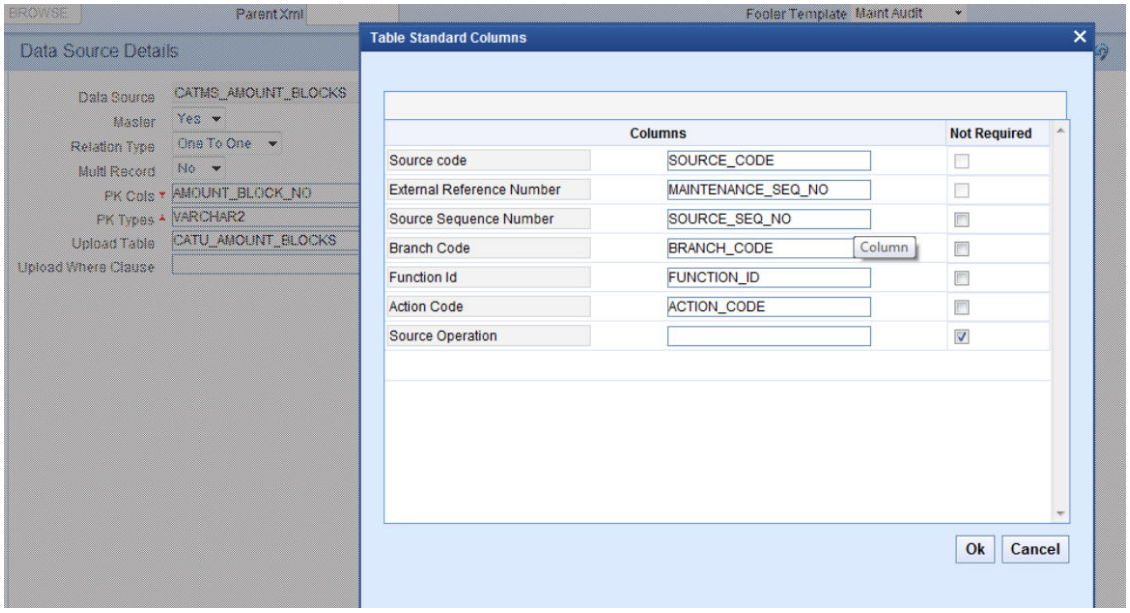
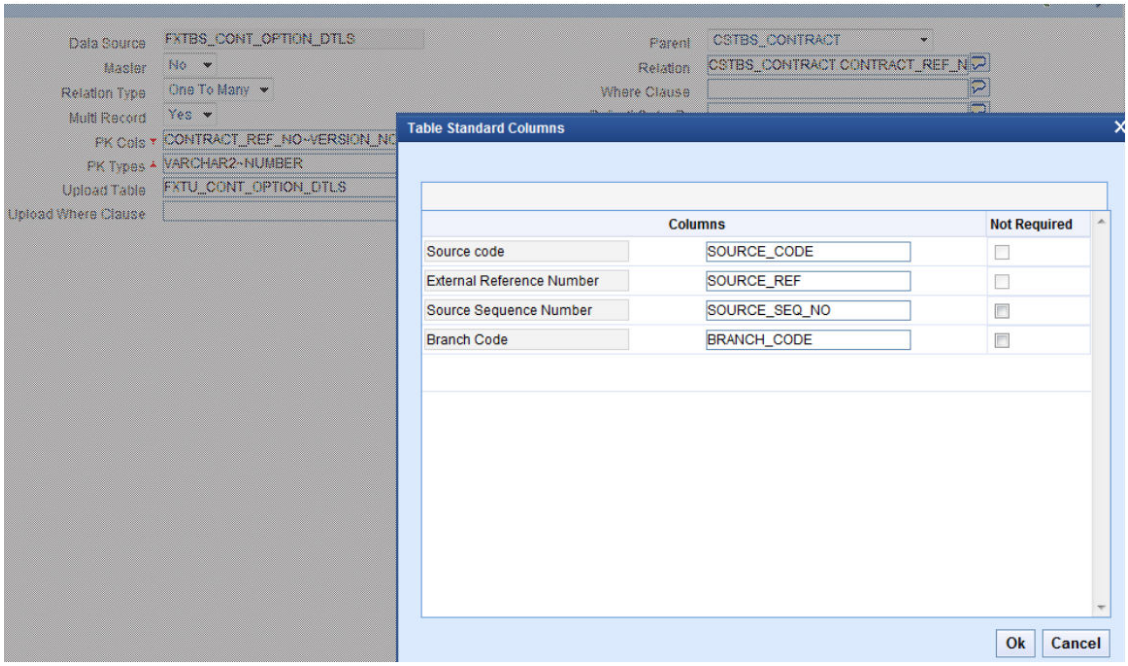
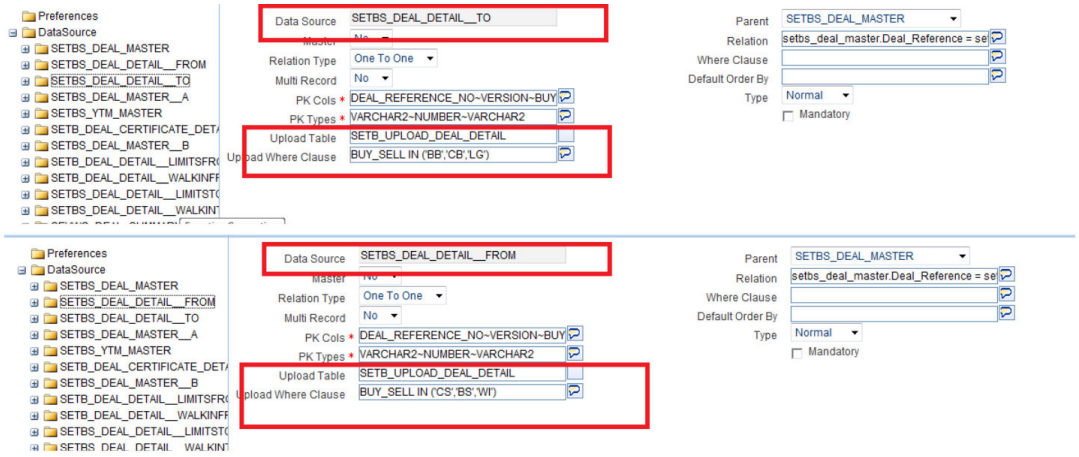


Figure 2-4 Table Standard Columns – Detail Upload Table



Upload Where Clause - If all the records in an Upload Table are not mapped to a particular data source, then the Upload Where Clause can be specified to filter the records. This is applicable only when the data sources involved are not directly related to each other.

Figure 2-5 Upload Where Clause – Multiple Data sources

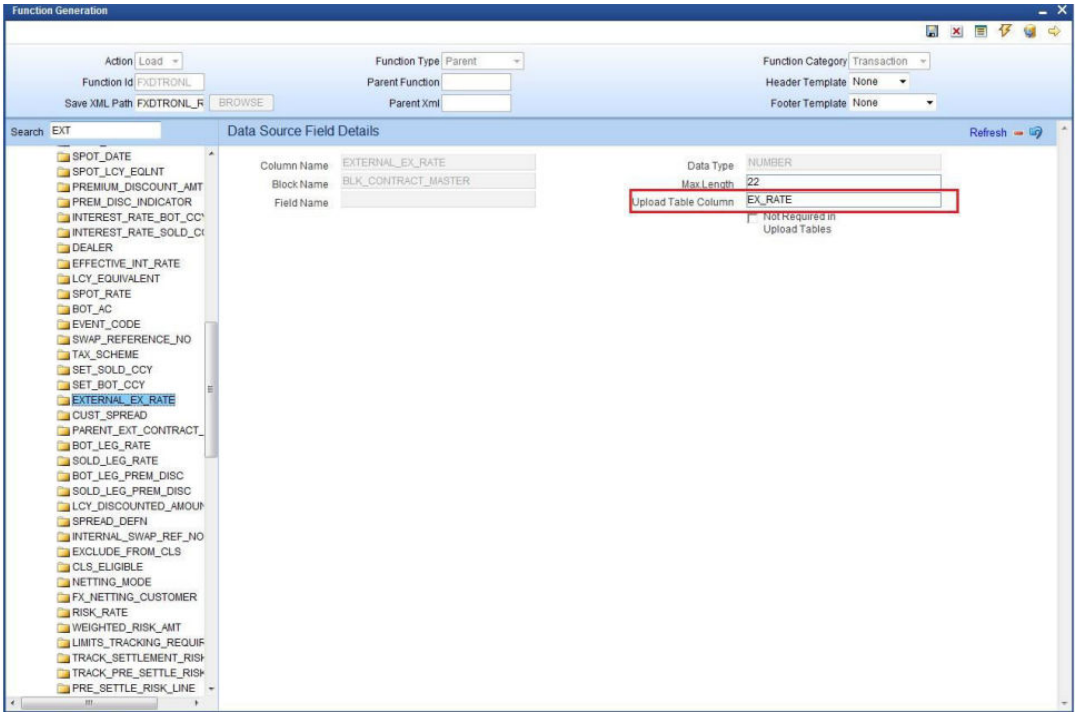


Upload Table Where Clause would be applied on the upload table, hence upload table columns should be used in the clause.

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

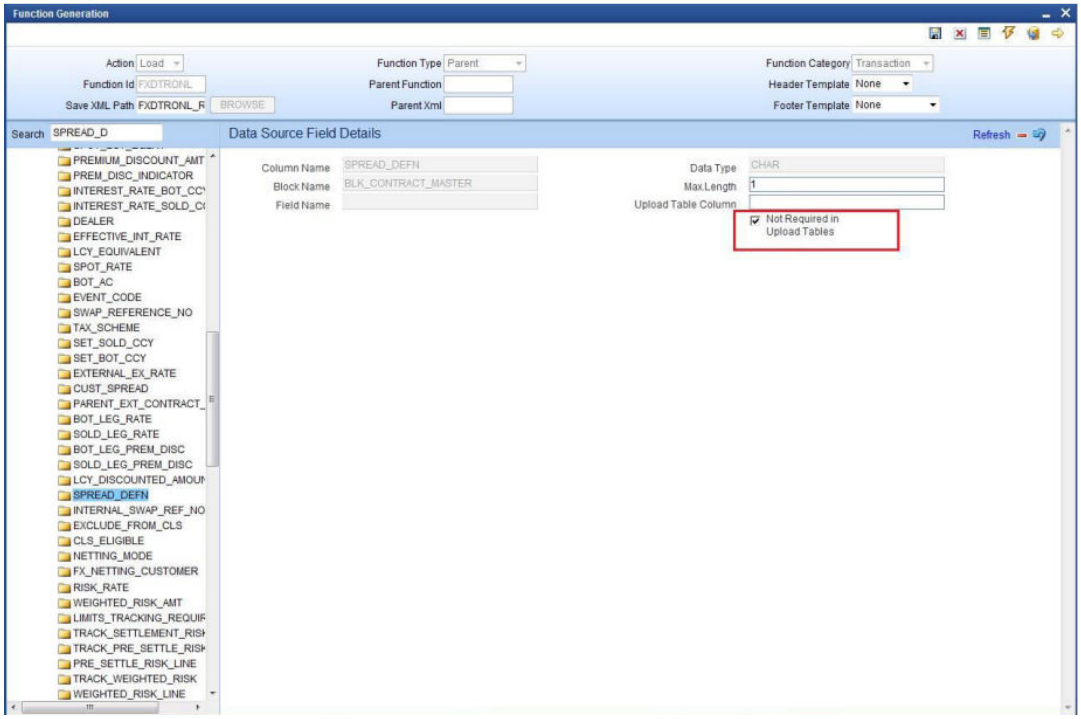
- By default, the name of the **Upload Table Column** would be assumed to be the same as that of the base table name.
- If the name of the upload table column has to be different from the base table column, then the same has to be explicitly mentioned in **Upload Table Column** field.

Figure 2-6 Data Source Field Details – Upload Table Column



- Check the **Not Required in Upload Tables** box to specify if any of the columns included in the data source in RADXML is not required in the upload table.

Figure 2-7 Data Source Field Details – Not Required in Upload Tables



2.2 Generated Units

This topic describes the detailed information on generated units.

1. **Upload Adapter Package** - Naming convention as **Modulepks_FunctionID_Ext_Upload**
2. **Triggers on Upload Table**
3. **Upload Table DDL**

> **Note**
>
> If existing upload tables are being used, DDL scripts can be ignored. Drop scripts for the table would be generated in a separate file.

2.3 Upgrade Capabilities

This topic describes the detailed information on upgrade capabilities.

The normal Open Development Tool upgrade feature is supported in upload table configurations as well. The customizations can be done on the configuration maintained in Open Development Tool. the customization can be done to:

1. Change the Upload Tables Mapped - Map new upload tables or remove existing table mapping
2. Modify/Remove/Add Upload Table Column Names
3. Configure upload tables for the entire screen if it is not provided by the engineering and if the bank needs the same.

Changes are done as part of customizations would be retained during the Open Development Tool refresh. Any new mappings done by the engineering team would reflect after the refresh. After refresh, all the artifacts have to be regenerated including upload table definitions.

A
Miscellaneous

This topic describes appending data in the standard Oracle FLEXCUBE Universal Banking framework.

Table A-1 Appending Data

Appending Data and Framework Details	Description
Appending Data	In certain scenarios, only the data which has to be appended would be uploaded to the upload tables. The requirement would be to append this data to the existing data in the table. This feature is not supported by the standard Oracle FLEXCUBE Universal Banking framework. For example, Upload of Floating Rates for a Currency
Oracle FLEXCUBE Universal Banking Framework	In Oracle FLEXCUBE Universal Banking, if any multi-record block has to be modified then the complete data of the block has to be sent. Oracle FLEXCUBE Universal Banking derives 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	The recommended approach is to handle this case based on the source operation parameter in the custom package.

The developer can either skip all the system functions and write code to upload data in a custom package for a particular source operation or Append the existing data to screen the object instance before the start of processing (preferably in **pre_check_mandatory**) for the particular source operation.