Oracle® Banking Enterprise Limits and Collateral Management

Development of Call Forms

Release 14.7.5.0.0
G25003-01
September 2024

Oracle Banking Enterprise Limits and Collateral Management Development of Call Forms, Release 14.7.5.0.0

G25003-01

Contents

1

Preface

1.1 Purpose

This guide is designed to help acquaint you with the Oracle Banking Enterprise Limits and Collateral Management (ELCM) application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

User can further obtain information specific to a particular field by placing the cursor on the relevant field and pressing <F1> on the keyboard.

1.2 Audience

This guide is intended for the following User/User Roles:

Table 1-1 Audience

Role	Function
Back office data entry clerk	Input functions for funds
Back office managers/officers	Authorization functions
Product Managers	Product definition and authorization
End of day operators	Processing during end of day / beginning of day

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.

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.

1.4 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.

1.5 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.

1.6 Basic Actions

Table 1-2 Basic Actions

Action	Description
Approve	Used to approve the initiated report. This button is displayed, once the user click **Authorize**.
Audit	Used to view the maker details, checker details, and report status.
Authorize	Used to authorize the report created. A maker of the screen is not allowed to authorize the report. Only a checker can authorize a report, created by a maker.
Close	Used to close a record. This action is available only when a record is created.
Confirm	Used to confirm the performed action.
Cancel	Used to cancel the performed action.
Compare	Used to view the comparison through the field values of old record and the current record. This button is displayed in the widget, once the user click **Authorize**.
Collapse All	Used to hide the details in the sections. This button is displayed, once the user click **Compare**.
Expand All	Used to expand and view all the details in the sections. This button is displayed, once the user click **Compare**.
New	Used to add a new record. When the user click **New**, the system displays a new record enabling to specify the required data.

Table 1-2 (Cont.) Basic Actions

Action	Description
OK	Used to confirm the details in the screen.
Save	Used to save the details entered or selected in the screen.
View	Used to view the report details in a particular modification stage. This button is displayed in the widget, once the user click **Authorize**.
View Difference only	Used to view a comparison through the field element values of old record and the current record, which has undergone changes. This button is displayed, once the user click **Compare**.
Unlock	Used to update the details of an existing record. System displays an existing record in editable mode.

1.7 Related Documents

For more information refer to the Oracle Banking manuals on:

- Development of Launch Forms and Others Screens
- Enterprise Collaterals User Guide
- Enterprise Limits and Collaterals Common User Guide

1.8 Conventions

The following text conventions are used in this document:

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.

1.9 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.

1.10 Acronyms and Abbreviations

The list of the acronyms and abbreviations used in this guide are as follows:

Table 1-3 Acronyms and Abbreviations

Acronyms	Abbreviations
CIF	Customer Information File

Table 1-3 (Cont.) Acronyms and Abbreviations

Acronyms	Abbreviations
CASA	Current Account and Savings Account
DDA	System that holds the CASA account and balances
ELCM	Enterprise Limits and Collateral Management
ECA	External Credit Approval
FCUBS	Oracle FLEXCUBE Universal Banking Solution
GW	Gateway
HTTP	Hyper Text Transfer Protocol
ID	Identification Number
Mark EOTI	Mark End of Transaction Input
Mark TI	Mark Transaction Input
OFSAA	Oracle Financial Services Analytical Applications
ORMD	Oracle Revenue and Billing Management
PK	Primary Key
RDBMS	Relational Data Base Management System
SMS	Security Services
UI	User Interface
VD	Value Date
XML	Extensible Mark-up Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

1.11 Symbols and Icons

The lists of symbols, buttons and shortcut key that are used in the application to perform various tasks are covered in this topic.

Table 1-4 Symbols and Icons

Icons	Function
🔍	Perform search
┘ └ ┐ ┌	Minimize
▶	Navigate to the next record
◀	Navigate to the previous record
○	Toggle OFF
●	Toggle ON
✕	Delete
+	Click this icon to add a new row.

Table 1-4 (Cont.) Symbols and Icons

Icons	Function
—	Click this icon to delete an existing row.
☰	List view
⛶	Maximize
|<	Navigate to the first record
>|	Navigate to the last record
🔍	Advance search
📝	Search record
💾	Save the record
↺	Reset the record
📋	Clear the record

Table 1-5 Symbols and Icons - Audit Details

Icons	Function
👤	A user
🏛	Branch details
📅	Date and Time

1.12 Prerequisite

Specify the **User ID** and **Password**, and login to **Home** screen.

2 Introduction

This provides introduction to the development of call form.

3
Overview of Call Form

This topic provides an overview of call form.

Call Forms are function Id’s (screens) which can be used for processing of a feature which is common across multiple function Ids. Call Forms can be attached to the main function Id for processing the common functionality. Call form screens cannot be launched independently.

Example: Tax Processing for a Contract

Tax Processing depends on common tax rules attached for the product/contract. Same processing can be used for various contract screens like Funds Transfer Input Screen, Letters Of Credit etc. Thus a common function id can be developed which can be attached to all the contract screens requiring tax processing.

On launching the call form screen from the main screen, the values will be picked up based on the data input in main screen. User will have the option to change the data in call form screen if desires so.

There are two types of Call forms they are:

1. Maintenance Call Forms
2. Transaction Call Forms

Maintenance Call forms can be attached to only maintenance function id’s while transaction call forms can be attached to transaction screens only.

4

Screen Development

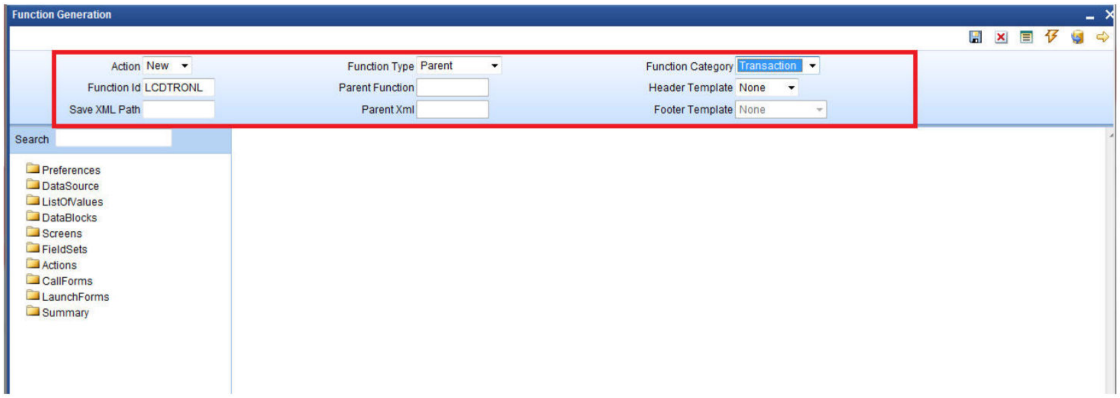
This topic describes about the development of screen.

Design and development of a Call Form function id is similar to any other function Ids. This section briefs the steps in designing a Call Form screen

For detailed explanation, refer the document: Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development

Header Information

Figure 4-1 Header Information



Action	New	Function Type	Parent	Function Category	Transaction
Function Id	LCDTRONL	Parent Function		Header Template	None
Save XML Path		Parent Xml		Footer Template	None

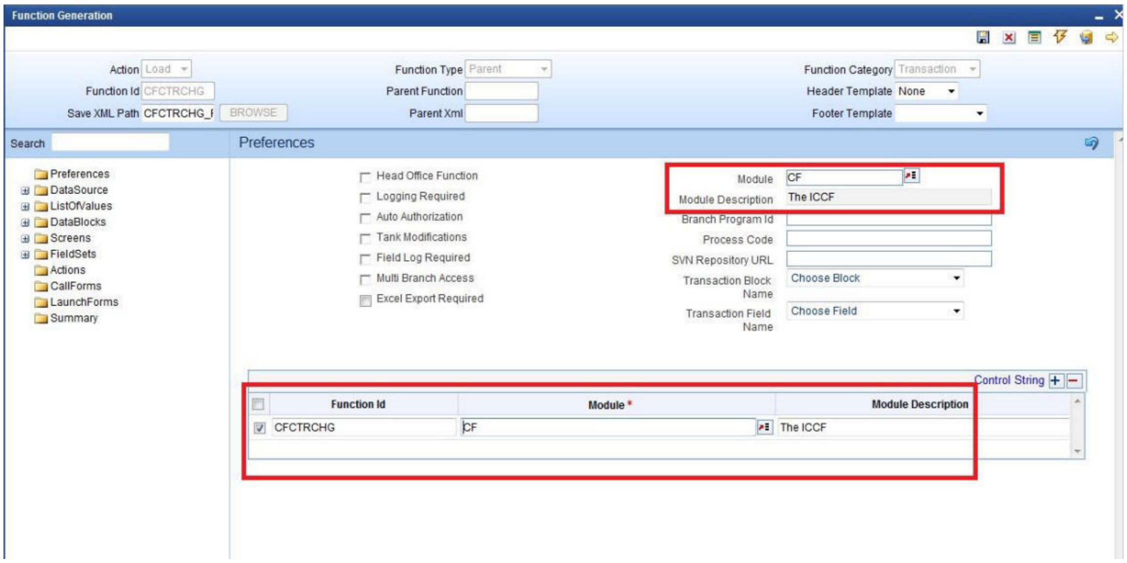
Note the following while providing header information

1. **Name of the Call form**: Call Form name has to have the third character as ‘C’. This is how system differentiates a call form from other screens. Ideally, the length of the name should be 8 characters.
   Example: CFCTRCHG, ISCTRSTL etc are valid call form names
2. **Call Form Category**: It has to be either Maintenance or Transaction depending on the functionality and the screens from which it will be invoked.
3. **Footer Template**: Footer template can be provided as required. Note for Transaction screens, footer template has to be selected as NONE unless it is a process screen.
4. **Function Type**: Parent and child functionality is supported for call forms.

Preferences

Note the following while providing Preferences for Call Forms.

Figure 4-2 Preferences



Action: Load
Function Id: CFCTRCHG
Save XML Path: CFCTRCHG_I
Function Type: Parent
Parent Function:
Parent Xml:
Function Category: Transaction
Header Template: None
Footer Template:

Preferences

□ Head Office Function
□ Logging Required
□ Auto Authorization
□ Tank Modifications
□ Field Log Required
□ Multi Branch Access
□ Excel Export Required

Module: CF
Module Description: The ICCF
Branch Program Id:
Process Code:
SVN Repository URL:
Transaction Block Name: Choose Block
Transaction Field Name: Choose Field

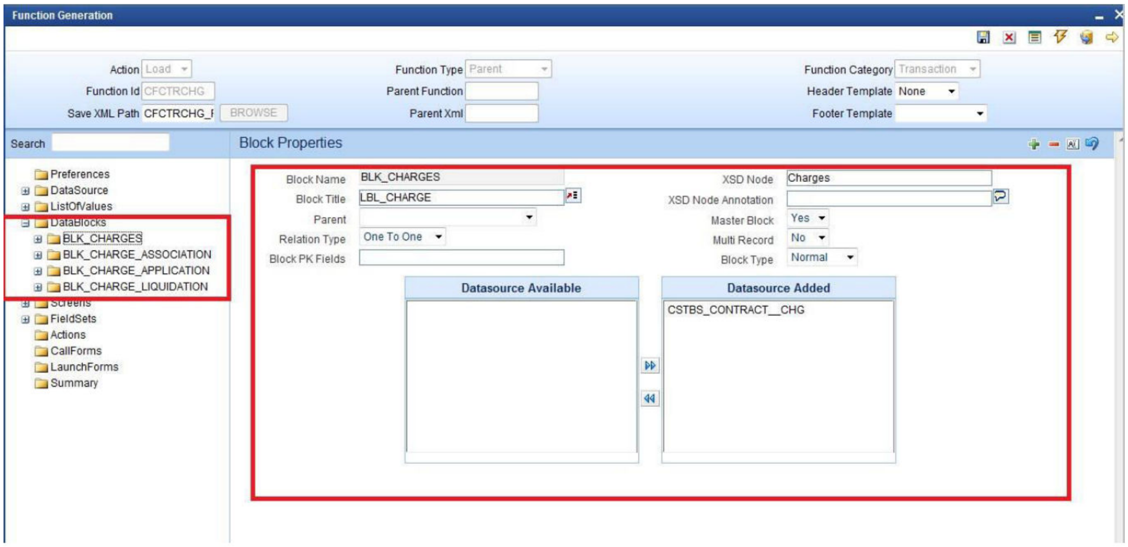
□	Function Id	Module *	Module Description
☑	CFCTRCHG	CF	The ICCF

1. **Module name**: Module name is a mandatory field and has to be provided. It is recommended that the first two letters of the function id is kept as same as the module name. Naming of the generated package will be derived from the module code maintained.
2. Of the menu details inc generated, only script for SMTB_MENU and SMTB_FCC_FCJ_MAPPING is required for Call Forms
3. **Browser menu options**: Call Forms cannot be launched independently .Hence browser menu labels need not be maintained. Script for smtb_function_description is not required for call forms

Data Sources

Identify the tables/views for the call form. Define data sources and add data source fields as required.

Figure 4-3 Data Sources



Action: Load
Function Id: CFCTRCHG
Save XML Path: CFCTRCHG_I
Function Type: Parent
Parent Function:
Parent Xml:
Function Category: Transaction
Header Template: None
Footer Template:

Block Properties

Block Name: BLK_CHARGES
Block Title: LBL_CHARGE
Parent:
Relation Type: One To One
Block PK Fields:
XSD Node: Charges
XSD Node Annotation:
Master Block: Yes
Multi Record: No
Block Type: Normal

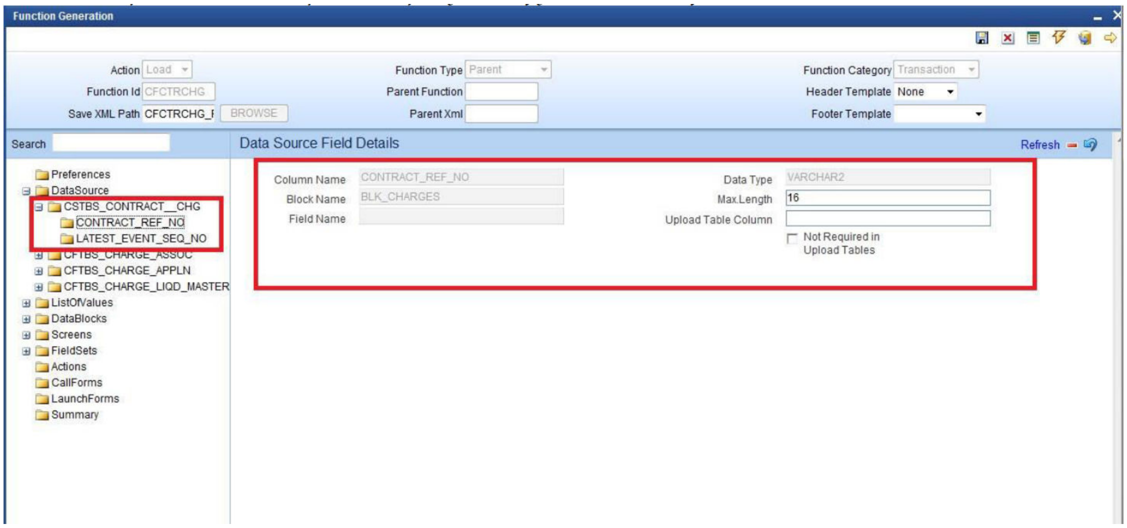
Datasource Added: CSTBS_CONTRACT__CHG

Note the following while creating data sources

1. Master Data Source has to be a single entry data source.
2. Logical Relationships has to be maintained for all data sources except the parent.
3. Provide PK Cols and PK types for all data sources.

If data source is a multi record block, then make sure it has at least one more pk than its parent which helps to uniquely identify each record of multi record block. Max length of the data source field can be modified as per requirement.

Figure 4-4 Data Sources



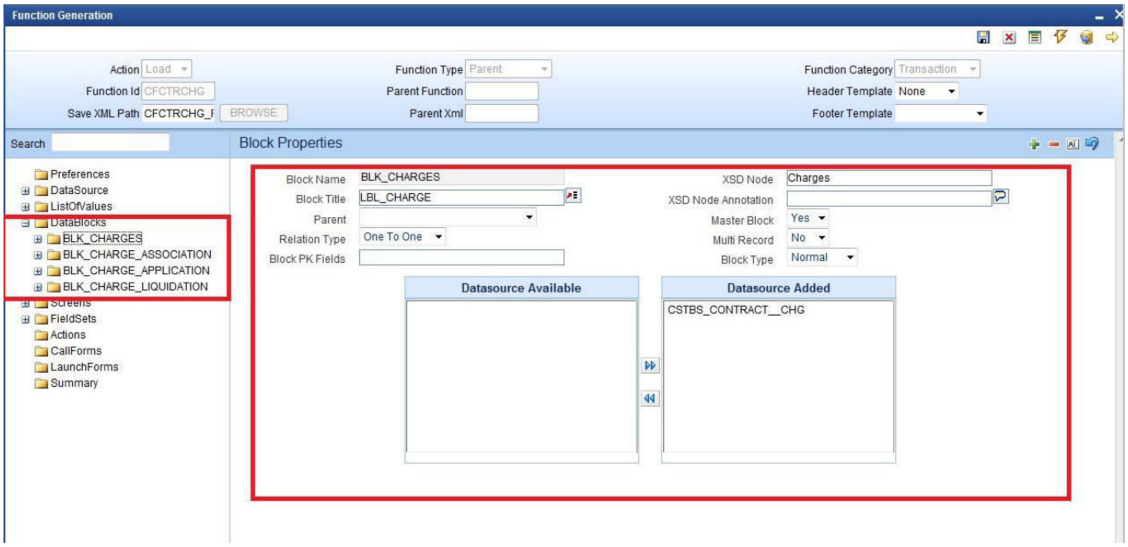
Action Load
Function Id CFCTRCHG
Save XML Path CFCTRCHG_I BROWSE
Function Type Parent
Parent Function
Parent Xml
Function Category Transaction
Header Template None
Footer Template

Column Name CONTRACT_REF_NO
Block Name BLK_CHARGES
Field Name
Data Type VARCHAR2
Max Length 16
Upload Table Column
□ Not Required in Upload Tables

Data Blocks

Determine the block structure for the function id .Define Data Blocks as per the design.

Figure 4-5 Data Blocks



Action Load
Function Id CFCTRCHG
Save XML Path CFCTRCHG_I BROWSE
Function Type Parent
Parent Function
Parent Xml
Function Category Transaction
Header Template None
Footer Template

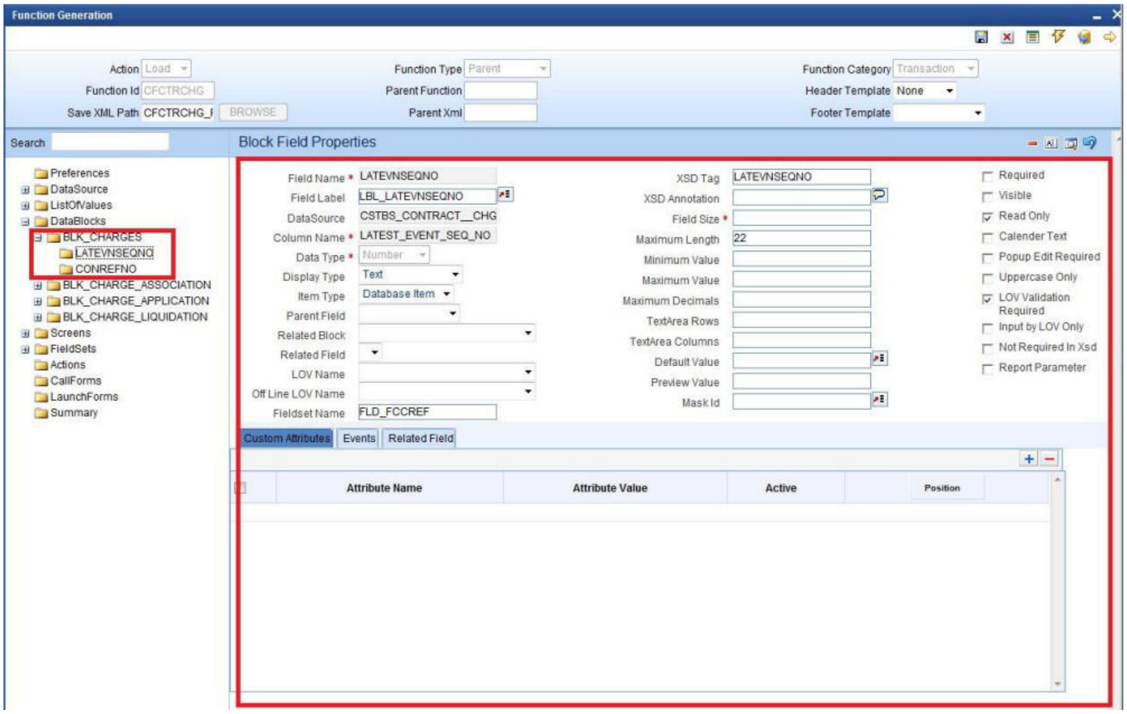
Block Name BLK_CHARGES
Block Title LBL_CHARGE
Parent
Relation Type One To One
Block PK Fields
XSD Node Charges
XSD Node Annotation
Master Block Yes
Multi Record No
Block Type Normal

Datasource Added: CSTBS_CONTRACT__CHG

Note the following while creating data blocks.

1. Master Data Source has to be a single entry data source.
2. Logical Relationships with the parent has to be maintained for all data sources.
3. Provide PK Cols and PK types for all data sources.
   If data source is a multi record block, then make sure it has at least one more pk than its parent which helps to uniquely identify each record of multi record block.
4. Provide Xsd node name if the block is normal and is required in gateway request

Figure 4-6   Data Blocks



Function Generation

Action: Load
Function Id: CFCTRCHG
Save XML Path: CFCTRCHG_I
Function Type: Parent
Parent Function:
Parent Xml:
Function Category: Transaction
Header Template: None
Footer Template:

Block Field Properties

Field Name *: LATEVNSEQNO
Field Label: LBL_LATEVNSEQNO
DataSource: CSTBS_CONTRACT__CHG
Column Name *: LATEST_EVENT_SEQ_NO
Data Type *: Number
Display Type: Text
Item Type: Database Item
Parent Field:
Related Block:
Related Field:
LOV Name:
Off Line LOV Name:
Fieldset Name: FLD_FCCREF

XSD Tag: LATEVNSEQNO
XSD Annotation:
Field Size *:
Maximum Length: 22
Minimum Value:
Maximum Value:
Maximum Decimals:
TextArea Rows:
TextArea Columns:
Default Value:
Preview Value:
Mask Id:

□ Required
□ Visible
☑ Read Only
□ Calender Text
□ Popup Edit Required
□ Uppercase Only
☑ LOV Validation Required
□ Input by LOV Only
□ Not Required In Xsd
□ Report Parameter

	Attribute Name	Attribute Value	Active	Position

Note the following while attaching block fields to data blocks

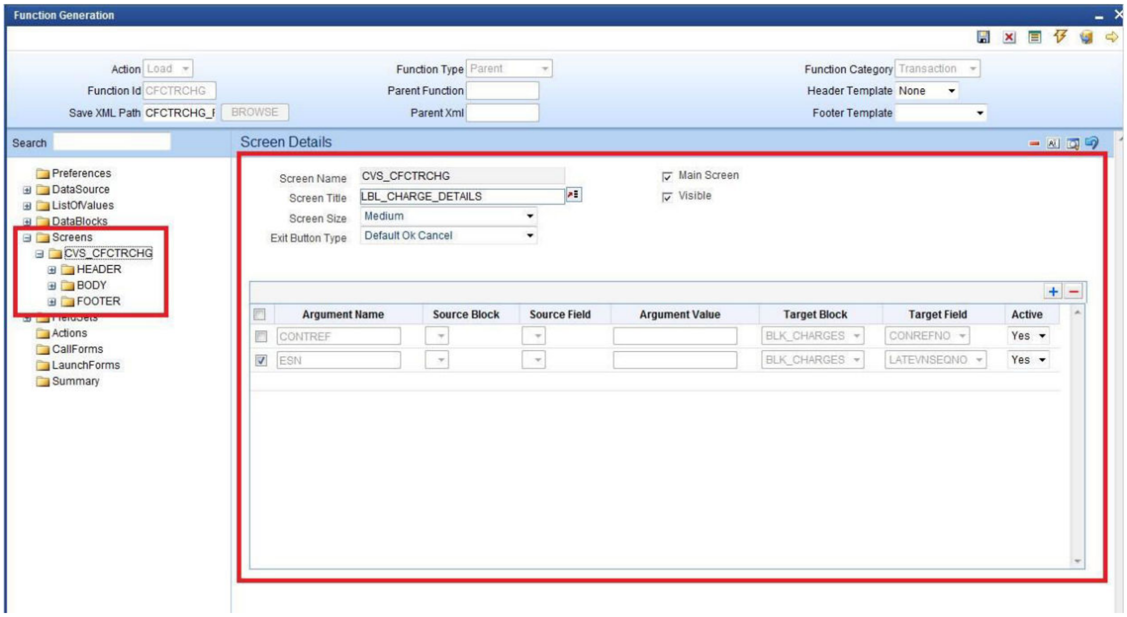
1. In case the field is not required in XSD, check not Required XSD.
2. Ensure that Related Block and Field are given for Amount Fields.
3. Minimize the use of query data sources by using DESC fields wherever possible.

> **Note:**
>
> Query data sources is rarely required for a Call Form screen; as launch form can be used for query only screens.

Screens

Design the screen layout based on the requirement

Figure 4-7 Screens



Action: Load
Function Id: CFCTRCHG
Save XML Path: CFCTRCHG_I
Function Type: Parent
Parent Function:
Parent Xml:
Function Category: Transaction
Header Template: None
Footer Template:

Screen Name: CVS_CFCTRCHG
Screen Title: LBL_CHARGE_DETAILS
Screen Size: Medium
Exit Button Type: Default Ok Cancel
☑ Main Screen
☑ Visible

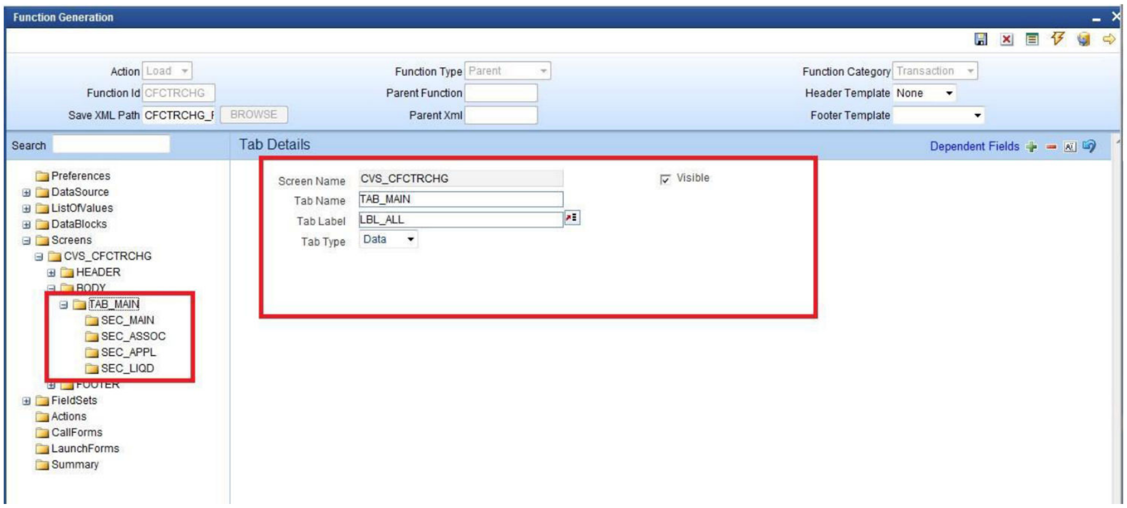
□	Argument Name	Source Block	Source Field	Argument Value	Target Block	Target Field	Active
□	CONTREF				BLK_CHARGES	CONREFNO	Yes
☑	ESN				BLK_CHARGES	LATEVNSEQNO	Yes

Note the following while creating screens

1. One Screen should be identified as the main screen; if multiple screens present.
2. In the function id ,where the call form is called is for the button (which launches call form) events, the main screen of the call form has to be mentioned.
3. **Screen Arguments** :
   Screen Arguments has to be provided for the main screen. Any field which has to be populated based on the data from the calling Function id can be provided as the target block and target field.

   Normally values for the pk fields of the master data source can be retrieved from the screen arguments .Relationship between the calling function id and the call form will also be based on the pk columns of master data source.

Figure 4-8 Screens



Action: Load
Function Id: CFCTRCHG
Save XML Path: CFCTRCHG_I
Function Type: Parent
Parent Function:
Parent Xml:
Function Category: Transaction
Header Template: None
Footer Template:

Screen Name: CVS_CFCTRCHG
Tab Name: TAB_MAIN
Tab Label: LBL_ALL
Tab Type: Data
☑ Visible

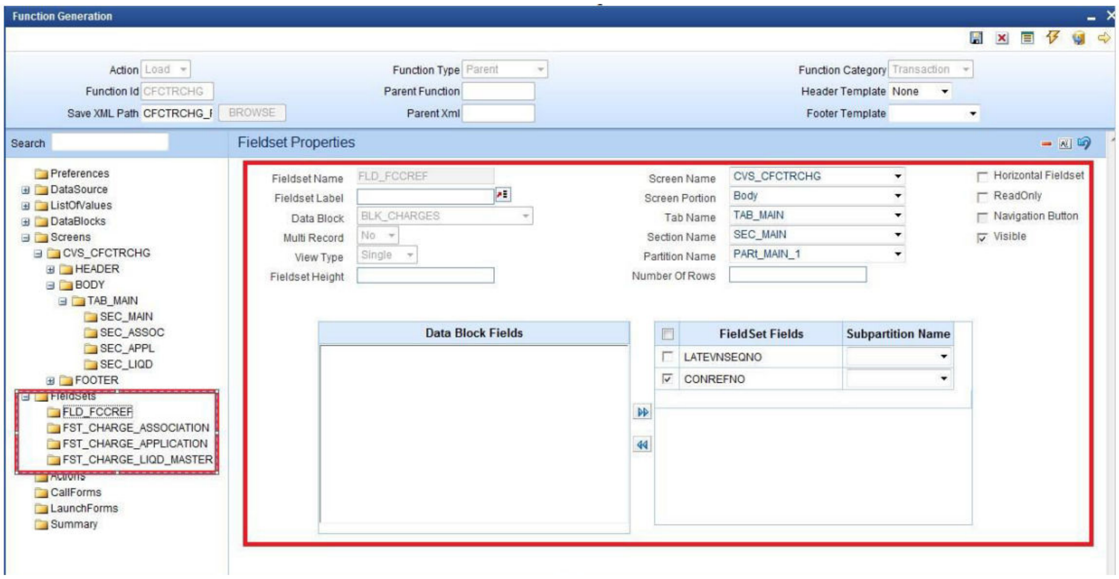
Note the following when creating tabs and sections for the screen

1. If the screen does not have multiple tabs, then only the TAB_MAIN needs to be used.
2. TAB_HEADER should not contain any sections in this scenario.

Field Sets

Not the following when attaching field to a field set

Figure 4-9 Field Sets



Function Generation

Action: Load
Function Id: CFCTRCHG
Save XML Path: CFCTRCHG_I
Function Type: Parent
Parent Function:
Parent Xml:
Function Category: Transaction
Header Template: None
Footer Template:

Fieldset Properties

Fieldset Name: FLD_FCCREF
Fieldset Label:
Data Block: BLK_CHARGES
Multi Record: No
View Type: Single
Fieldset Height:
Screen Name: CVS_CFCTRCHG
Screen Portion: Body
Tab Name: TAB_MAIN
Section Name: SEC_MAIN
Partition Name: PARt_MAIN_1
Number Of Rows:
□ Horizontal Fieldset
□ ReadOnly
□ Navigation Button
☑ Visible

Data Block Fields

□	FieldSet Fields	Subpartition Name
□	LATEVNSEQNO	
☑	CONREFNO	

1. If a field value is passed as screen argument ,but is not required to be shown in the screen, The field has to be made invisible and attached to a field set.
2. If it is not attached to any fields set, the screen html won’t contain the field and may result in script error while loading

Actions

Mention the web service and amendable information in Actions Screen

Note the following while maintaining web services and amendable information

1. Call forms will generate only Type XSD. Operation specific message xsd’s will not be generated. Call form Type will be part of the main function Id xsd; hence separate message xsd is not required for call form ‘**Subsys**’ will be added to the name of call form type xsd.
   Example: for the example given in the figure, name of the xsd generated will be SubSysTxnChgDtls-Types.xsd
2. Operation Id and Operation Code need not be maintained for the above mentioned reason
3. Amendable information has to be maintained similar to any other function id’s.

Launch Forms

Launch Forms can be attached to a Call form screen. Though it is technically supported, practical scenarios where launch form is part of a call form is very rare. Process to attach launch forms is similar to any other function Id’s.

Call Forms

Call forms can themselves be attached to a call form. This scenario also is practically very rarely used. Processing logic (sub system pickup) for the attached cal forms has to be called from the main call form

Summary

Summary screens are not required for Call Form screens. Since a Call Form screen cannot be launched independently in FLEXCUBE, it doesn’t require a summary screen

Preview

Generate the units for call form and deploy them in the FLEXCUBE server for unit testing.

5

Attach Call Form to Main Function Id

This topic describes about the attach call form to main Function Id.

Call Forms cannot be launched independently. It has to be called from a main function id. Refer Call Forms section in Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development for detailed explanation

> **Note:**
>
> Scripts for CSTB_CALL_FORM_NODES and SMTB_MENU tables generated by Call Form screen has to be deployed in FLEXCUBE schema before attaching Call form to the main function Id.

6
Generated Units

This topic describes about the generated units.

The following units will be generated for a call form screen.

Refer document on generated units on detailed explanation on the same

7

Front End Units

This topic describes about the front end units

Language xml

This file is an XML markup of presentation details, for the designed Call Form specific to a language.

SYS JavaScript File

This JavaScript file mainly contains a list of declared variables required for the functioning of the screen

Release Type Specific JavaScript File

This file won’t be generated by the Tool. It has to be manually written by the developer if he has to write any code specific in that release

8

Data Base Units

This topic describes about the data base units.

Static Scripts

The following static scripts generated are required for the proper functioning of a Call Form screen. Refer document on generated units for detailed explanation

1. Menu Details
   Scripts for SMTB_MENU and SMTB_FCC_FCJ_MAPPING are required for the functioning of Call Form screen
2. Call Form details
   Script for CSTB_CALL_FORM_NODES is required for attaching the call forms to the main function id. This has to be compiled in the schema before attaching the Call form to the main function Id
3. Lov Details
4. Amendable Details
5. Label details
6. Screen Details
7. Block details
8. Data Source Details

System Packages

Main package would be generated by the Tool and should not be modified by the developer.

There is small change in the structure of the package depending on the type of the call form (Maintenance or Transaction).

Unlike normal maintenance function ids, call form packages does not have any call to the business logic within itself (similar to transaction function id). If developer wishes to uses any functions within the main package , call has to be made from the release specific package.

Main package contains functions for :

- Converting Ts to PL/SQL Composite Type
- Calling fn_main.
- Mandatory checks (fn_check_mandatory).
- Default and validation(fn_default_and_validate)
- Querying(fn_query)
- Converting the Modified Composite Type again to TS

Except the functions for type conversions, others functions calls the respective hook functions in hook packages of the call forms. Thus no processing logic within the main package is used It is to be noted that each of these functions are called from the main package of the main function id (where this call form is used) during respective stages.

But the package contains many other system generated functions for operations like

- Mandatory checks(fn_sys_check_mandatory)
- Default and validation(fn_sys_default_and_validate)
- Uploading to DB(fn_sys_upload_db)
- Query operation (fn_sys_query) etc

These functions are not called anywhere in the package. These functions if required can be called by the developer from the release specific package. Otherwise developer can write his own logic for the same in the Hook Packages

Hook Packages

Release specific packages will be generated based on the release type (KERNEL.CLUSTER or CUSTOM). The structure of the package depends on the type of call form (Maintenance or Transaction). Developer can add his code in the release specific hook package.

9 Other Units

This topic describes about the other units used in the module.

Xsd

Only Type XSD will be generated for a Call Form function Id. Subscript **Subys** will be added before XSD Type identifier in the name of the generated xsd.

This type xsd will be used in the type xsd of any function which uses the particular call form

10 Extensible Development

This topic describes about the Extensible Development

Developer can add his code in hook packages and release specific JavaScript file.

Extensibility in JavaScript Coding

For release specific JavaScript coding, code has to be written in release specific JavaScript file. It follows the naming convention as : (Function Id)_(Release Type).js

Example: Code in CFCTRCHG_CLUSTER.js is exclusive to cluster release

This JavaScript file allows developer to add functional code and is specific to release. The functions in this file are generally triggered by screen events. A developer working in cluster release would add functions based on two categories:

- Functions triggered by screen loading events
  Example: fnPreLoad_CLUSTER(), fnPostLoad_CLUSTER()
- Functions triggered by screen action events
  Example: fnPreNew_ CLUSTER (), fnPostNew_ CLUSTER ()

Extensibility in Backend Coding

Release specific code has to be written in the Hook Packages generated. Structure of a Maintenance and Transaction Call Form hook packages are almost the same

> **Note:**
>
> though structure is almost the same ,arguments differ in transaction and maintenance call forms .Hence Transaction Call Form can be attached only with Transaction screen and similarly for Maintenance screens

Different functions available in the Hook Package of a Call Form are:

1. Skip Handler : Pr_Skip_Handler
   This can be used to skip the logic written in another release.

   Example: logic written in KERNEL release can be skipped in CLUSTER release
2. Fn Main
   This is called form the fn_main in main package.
3. Fn_pre_query
4. Fn_post_query
   Any specific logic while querying can be written in these functions. It is called from fn_query of the main package
5. Fn_pre_upload_db
6. Fn_post_upload_db
   Any logic while uploading data to tables can be written here.

7. Fn_pre_default_and_validate
8. Fn_post_default_and_validate
   Any release specific logic for defaulting and validation can be written here . It is called from the fn_default_and_validate in the main package
9. Fn_pre_check_mandatory
10. Fn_post_check_mandatory
    Any mandatory checks can be validated here.
11. Fn_pre_process
12. Fn_post_process
    These hook functions are specific to transaction call form screens. These are called from fn_process of the main package which in turn is called from fn_process of the calling function id

Refer maintenance and Transaction Screen development document for further explanation