

Development of Maintenance Form
Oracle FLEXCUBE Investor Servicing
Release 14.7.0.1.0
Part No. F71715-01
[May] [2023]



Table of Contents

1. Preface	3
1.1 Audience	3
1.2 Related Documents	3
2. Introduction	4
2.1 How to use this Guide	4
3. Overview of Maintenance Screen	5
4. Screen Development	6
4.1 Header Information	6
4.2 Preferences	8
4.3 Data Sources	9
4.4 Data Blocks	12
4.5 Screens	15
4.6 Field Sets	17
4.7 LOV	20
4.8 Attaching Call forms	24
4.9 Adding Summary	27
4.10 Amendable fields Maintenance	29
5. Generation and Deployment of files	31
6. Generated Units	33
6.1 Front End Units	33
6.1.1 Language xml	33
6.1.2 SYS JavaScript File	33
6.1.3 Release Type Specific JavaScript File	33
6.2 Data Base Units	33
6.2.1 Static Scripts	33
6.2.2 System Packages	34
6.2.3 Hook Packages	34
6.3 Other Units	35
6.3.1 Xsd	35
7. Extensible Development	36
7.1 Extensibility in JavaScript Coding	36
7.2 Extensibility in Backend Coding	36
7.2.1 Functions in Hook Packages	36
7.2.2 Flow of control through Hook packages	37
7.2.3 By passing Base Release Functionality	38

1. Preface

This document describes Maintenance Screens in FLEXCUBE and the process of designing a simple Maintenance form using Oracle FLEXCUBE Development Workbench for Universal Banking

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:

<i>Proficiency</i>	<i>Resources</i>
FLEXCUBE Functional Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Screen Development	<i>04-Development_WorkBench_Screen_Development-I.docx</i>
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

[*04-Development_WorkBench_Screen_Development-I.pdf*](#)

[*05-Development_WorkBench_Screen_Development-II.pdf*](#)

2. Introduction

2.1 How to use this Guide

The information in this document includes:

- [Chapter 2 , "Introduction"](#)
- [Chapter 3 , "Overview of Call Form"](#)
- [Chapter 4 , "Screen Development"](#)
- [Chapter 5 , "Generated Units"](#)
- [Chapter 5 , "Extensible Development"](#)

3. Overview of Maintenance Screen

Maintenance Function Id's are used for storing maintenance data which are required for processing of any contracts, batches or for any other maintenance which are dependent on this

Example: Customer maintenance screen

If any customer wants to use the service of a bank, details about the customer will have to be maintained in the system .This will be maintenance data which will be required for other maintenances (creating account for the customer) as well as for transaction processing (debiting of customer account)

Business logic for a maintenance function id would be provided by the Development Workbench generated files .Most of the cases, system provided logic would be sufficient .Extra validations can be coded in the hook packages by the developer.

4. Screen Development

Design and development of a Maintenance function id is similar to any other function Ids. This section briefs the steps in designing a Maintenance screen. UTFDNDRL is sample function id used for demonstration in this document

For detailed explanation, refer the document:

[04-Development WorkBench Screen Development-I.pdf](#)

4.1 Header Information

Provide the header information as shown in the figure.

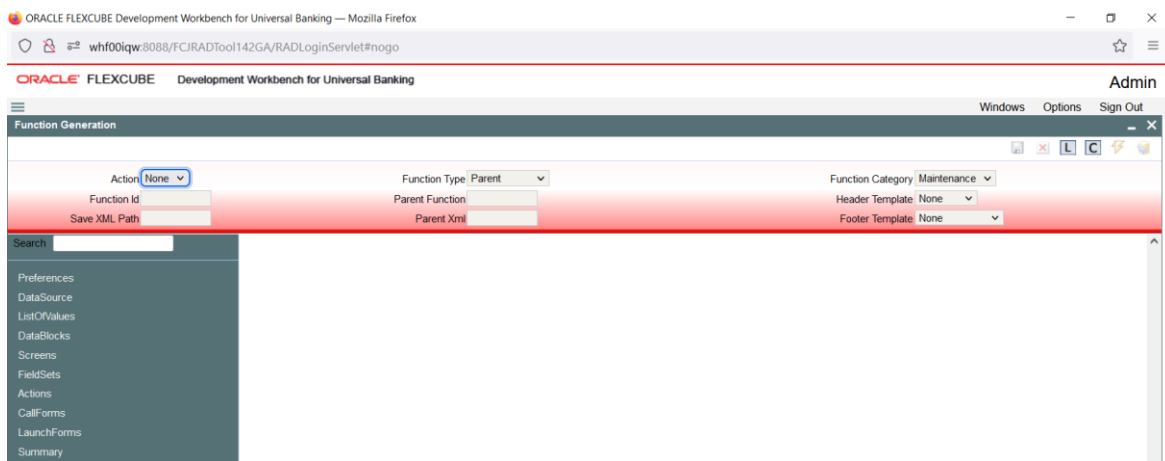


Fig: Providing Header Information for Maintenance Screen

- For new screen select action As New.
- Enter Function ID → UTFDNDRL
- Function Type → Parent
- Function Category → Maintenance
- Parent Function Id → None
- Parent Xml → None
- Header Template → None (Only for Process flow screens)
- Footer Template → Maint Audit

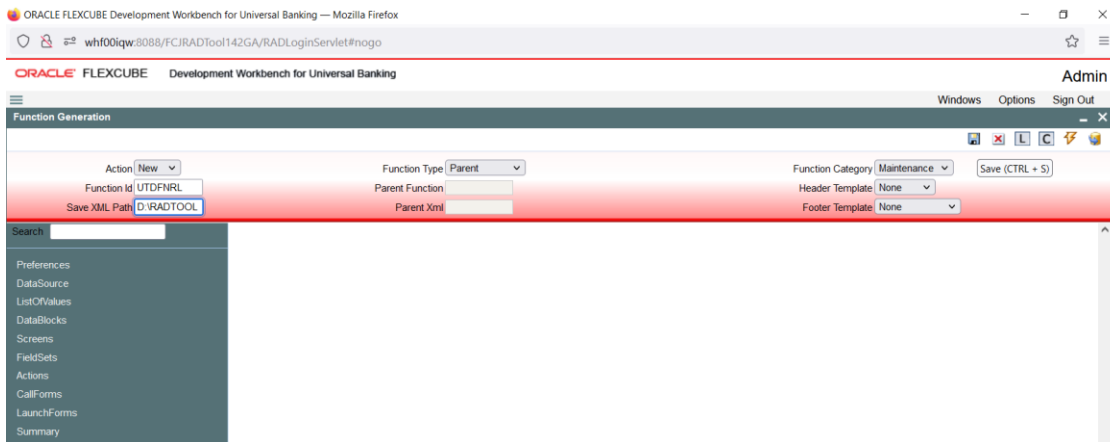


Fig: Save icon used for saving the radxml

User can save work at any point in time. Click the save icon on top right for the same .In order to work again with it select action as Load and load radxml from the hard disk path

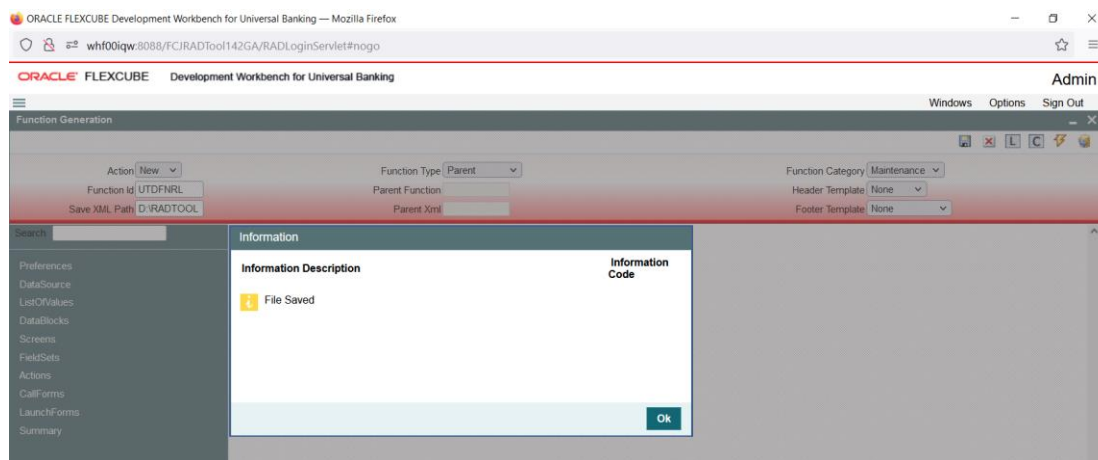


Fig: Saved File Information page

Note the following while providing header information for Maintenance screen

i) **Naming Convention:**

The third letter of the function id has to be D. Ideally the function id name should have 8 characters.

ii) **Footer Template**

Make sure that the master data source has the audit columns if footer template is provided as Maint log.

Refer [04-Development WorkBench Screen Development-I.docx](#) for detailed explanation

4.2 Preferences

- Details entered in Preferences are used in generating INCS for SMTB_MENU, SMTB_FUNCTION_DESCRIPTION and SMTB_ROLE_DETAILS.
- Control String** → Developer needs to select the actions which should be available for this screen in FLEXCUBE.

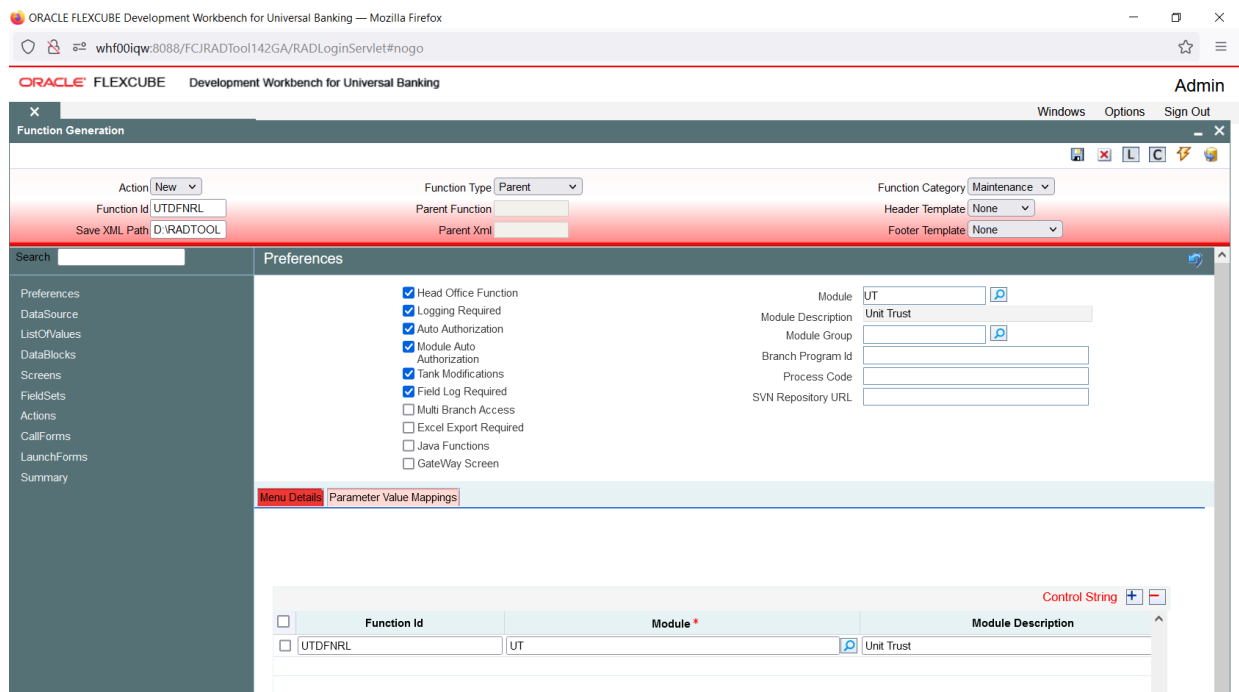


Fig: Providing Preferences for Maintenance Screen

Note the following points while providing details in Preferences screen

i) Control String

REVERSE, ROLLOVER, CONFIRM, LIQUIDATE, HOLD operations are not applicable for maintenance screens.

i) Defining Browser Menu Tree

Browser menu tree will be defined in the script generated for **smtb_function_description**.

The following labels has to be maintained for generation of proper script

Main Menu: LBL_{function id}_MAIN_MENU

Sub Menu 1: LBL_{function id}_SUB_MENU_1

Sub Menu 2: LBL_{function id}_SUB_MENU_2

Description: LBL_{function id}_DESC

Example: For UTDFNDRL, following labels has to be maintained

LBL_UTDFNDRL_MAIN_MENU, LBL_UTDFNDRL_SUB_MENU_1,

LBL_UTDFNDRL_SUB_MENU_2, LBL_UTDFNDRL_DESC

Refer [Development WorkBench Screen Development-I.docx](#) for detailed explanation on preferences.

4.3 Data Sources

- Right Click on Data Sources; click on Add. Add table window gets opened.
- If user knows the exact table name, he can enter name directly; else go to List Of values to get the list of tables available. Select the required table from the list.

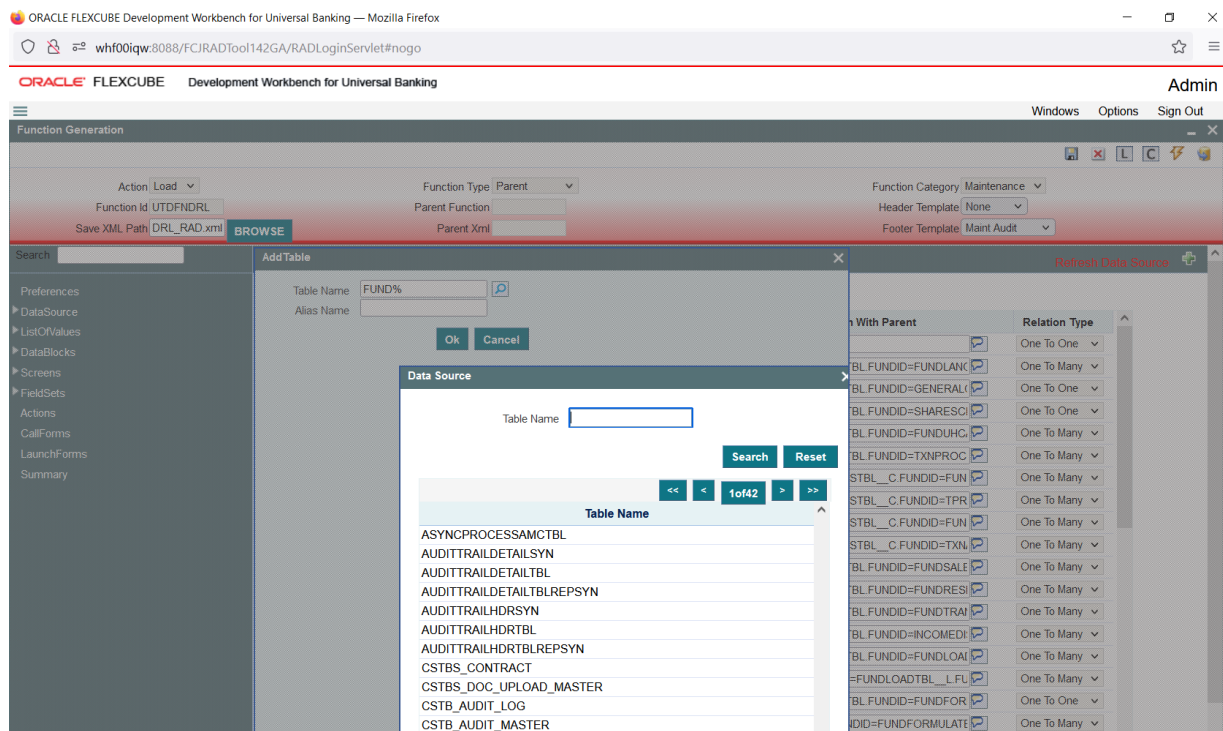


Fig: Adding Data Sources for the Function id

- Select Master as Yes if added data source is Master Data Source for the screen. Every function id should have one master data source.
- **Primary Key columns** (i.e. Pk Cols) and **Primary Types** (i.e. Pk Types) are mandatory. If it is already maintained in user schema in STTB_PK_COLS it will populated automatically otherwise user needs to enter values without fail. If user misses Pk cols and Pk Types package generation will fail.

Note: Master Data Source cannot have any parent.

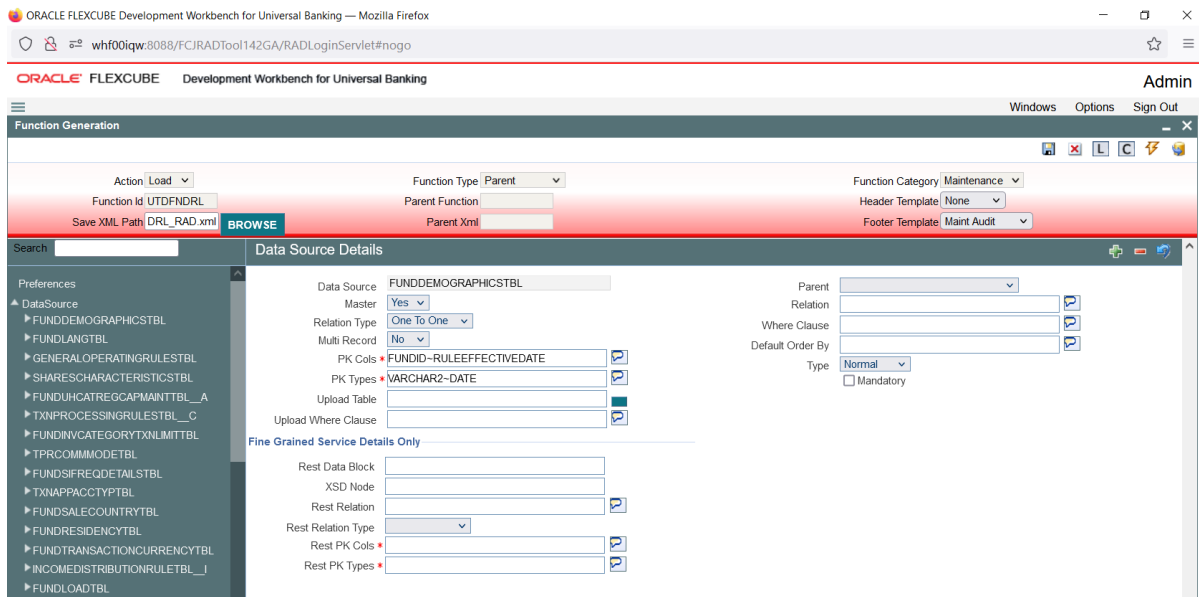


Fig: Providing master Data Source Properties

- Right Click on Added Table (STTM_CUSTOMER) to add fields to the table. Popup window gets opened with available columns in data source. Select the required fields and click ok. Selected will get added to the Data Source Tree.

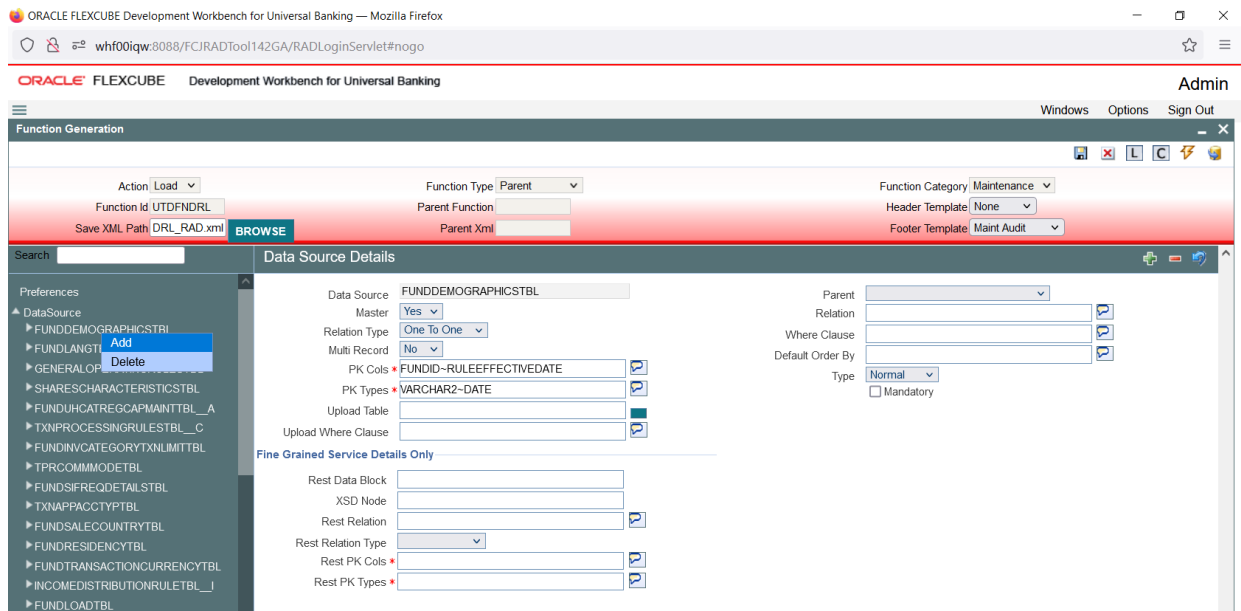


Fig: Including Data Source Fields for the Data Source

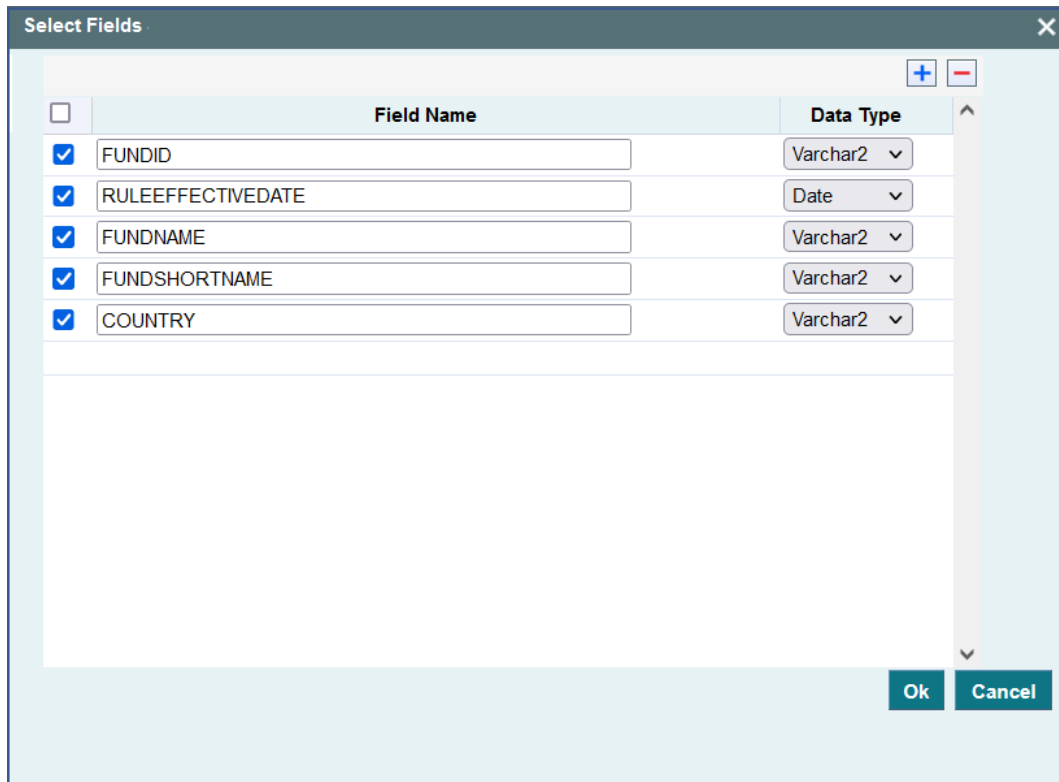


Fig: Selecting Data Source Fields for the Data Source

Data Source Field Properties:

Only max length can be modified by the developer in data source field properties. Rest will be defaulted from table definition

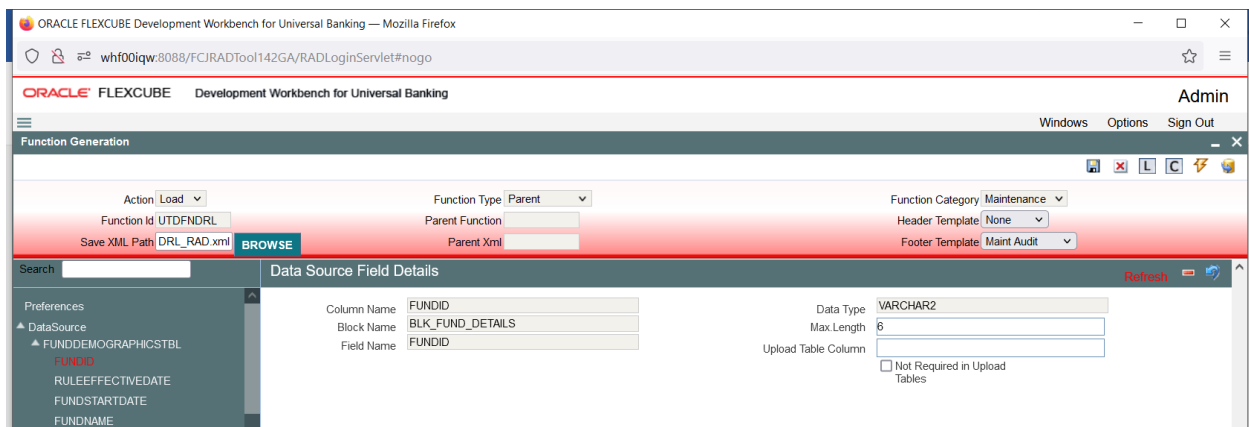


Fig: Providing properties for Data Source Fields

Data model of a single function id would include multiple tables. All the tables need to add in the function id. Note the following while adding child data sources

Adding Child Data Source:

- Select Multi Record value as Yes if child data source is Multi record table.
- Child Data Source should always be associated with a parent.
- Relation is mandatory between parent and child. While giving relation, parent data source should come in left side of the relation.

Fig: Providing properties for Child Data Source

Note: A data source cannot be parent to itself.

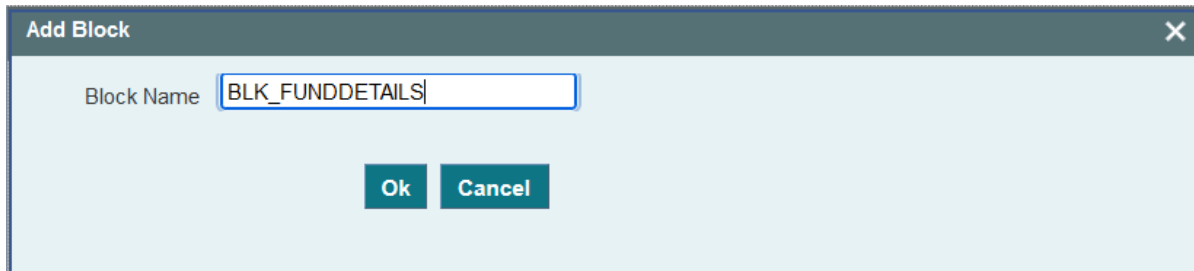
Note the following while adding data sources:

- If the data source is designed with relation type as 1: N with its parent, then it should have at least one more Pk col than its parent (assuming relationship is based on Pk cols).
- Master data source needs to have the audit columns if footer template is Maint audit; but those should not be added to data source fields as system will handle it

Refer [Development WorkBench Screen Development-I.docx](#) for detailed explanation on data sources

4.4 Data Blocks

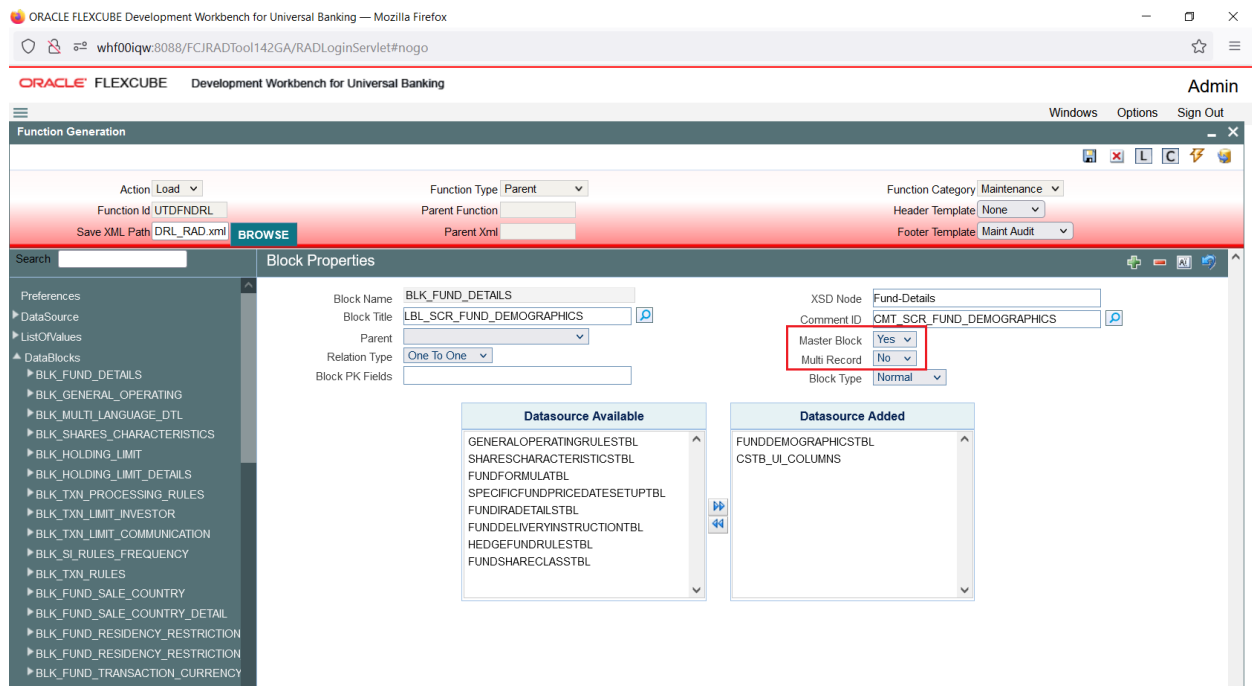
- Block Name should start with BLK_<short Name equivalent to data source but not exactly same as Data Source name>.



The 'Add Block' dialog box has a title bar with a close button. Inside, there is a text field labeled 'Block Name' containing the text 'BLK_FUNDDetails'. Below the text field are two buttons: 'Ok' and 'Cancel'.

Fig: Creating a new Data Block

- Select Parent block if added block is not Master Block.
- Select Multi Record (Yes/No) based on this value, available data sources will displayed in data source available text area.



The screenshot shows the 'Block Properties' window in the Oracle Flexcube Development Workbench. The window has a title bar with 'Admin' and 'Sign Out' buttons. The main area is divided into several sections:

- Function Generation:** Includes fields for 'Action' (Load), 'Function Id' (UTDFNDRL), 'Function Category' (Maintenance), 'Header Template' (None), and 'Footer Template' (Maint Audit). There is a 'Browse' button for 'Save XML Path' (DRL_RAD.xml).
- Block Properties:** Includes fields for 'Block Name' (BLK_FUND_DETAILS), 'Block Title' (LBL_SCR_FUND_DEMOGRAPHICS), 'Parent' (blank), 'Relation Type' (One To One), and 'Block PK Fields' (blank).
- Block Type:** Includes a dropdown for 'Block Type' (Normal).
- Master Block:** Includes a dropdown for 'Master Block' (Yes).
- Multi Record:** Includes a dropdown for 'Multi Record' (No).
- Datasource Available:** A list of available data sources including GENERALOPERATINGRULESTBL, SHARESCHARACTERISTICSTBL, FUNDFORMULATBL, SPECIFICFUNDPRICEDATESETUPTBL, FUNDIRADDETAILSTBL, FUNDELIVERYINSTRUCTIONTBL, HEDGEFUNDRULESTBL, and FUNDSHARECLASSTBL.
- Datasource Added:** A list of added data sources including FUNDDEMOGRAPHICSTBL and CSTB_UL_COLUMNS.

Fig: Providing properties for Data Block

- Select the required data source and click move button to attach Data Source to the block

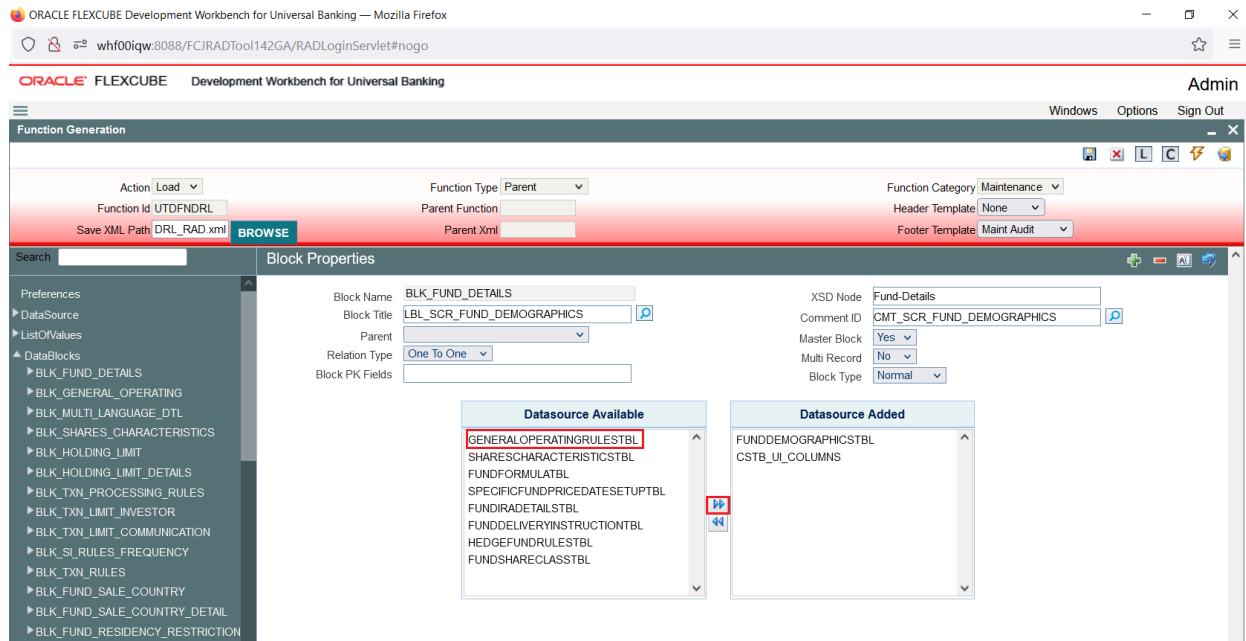


Fig: Attaching Data Sources to Data Block

Adding multi record data source to data block:

User on selecting Multi record 'Yes' in data block properties all the data sources with multi record Yes will be populated. **Multi Data Source once used to one block won't available for reuse whereas single record data source can be used in multiple blocks**

Select Block Fields:

- Right click on added block. Select Fields window will get opened. Developer needs to check the right side check box to add the required fields.
- **Field Name:** It should not be the same as column name .Special characters are also not allowed in the field name (including underscore and space)
- **Label Code:** It will be automatically populated based on field name.

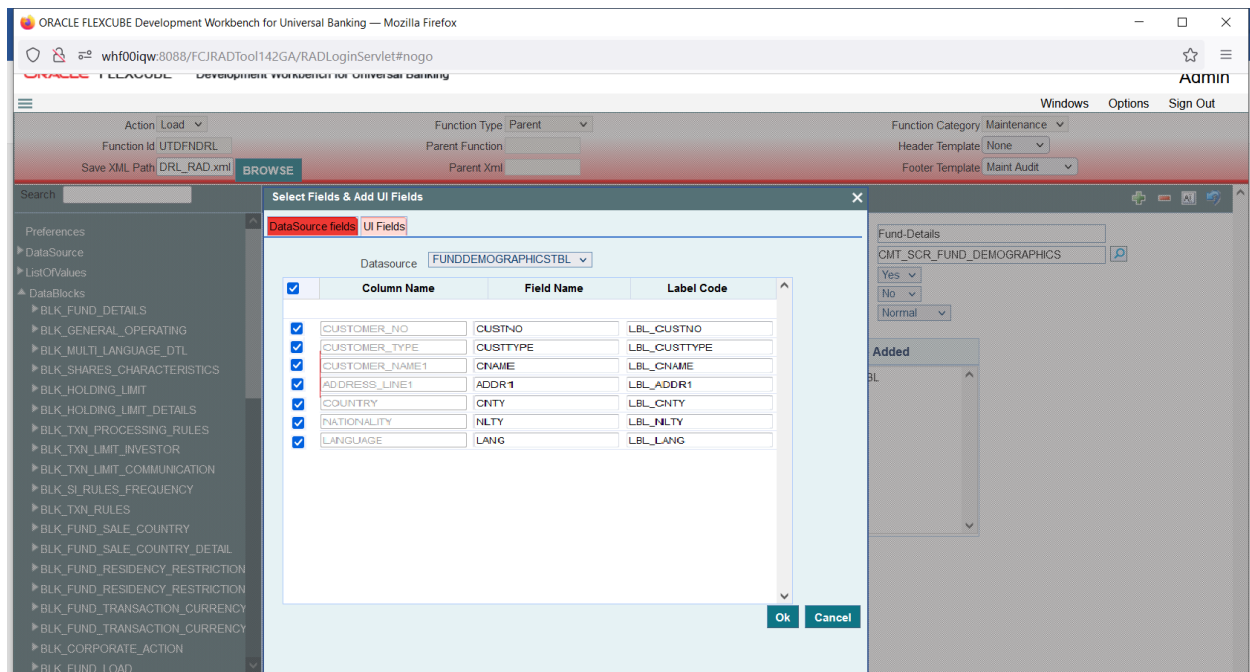


Fig: Adding Block Fields to Data Block

Refer [Development WorkBench_Screen_Development-I.docx](#) for detailed explanation on data blocks and block field properties

4.5 Screens

- Right click on Screens node to add a new screen
- Screen Name should start with CVS_<Name>.
- By default screen are divided into 3 parts.
- One Main Screen is Mandatory.
- Tabs can be defined on any of the screen portions as required
- User can add sections to tabs.
- Each section can be divided into partitions.

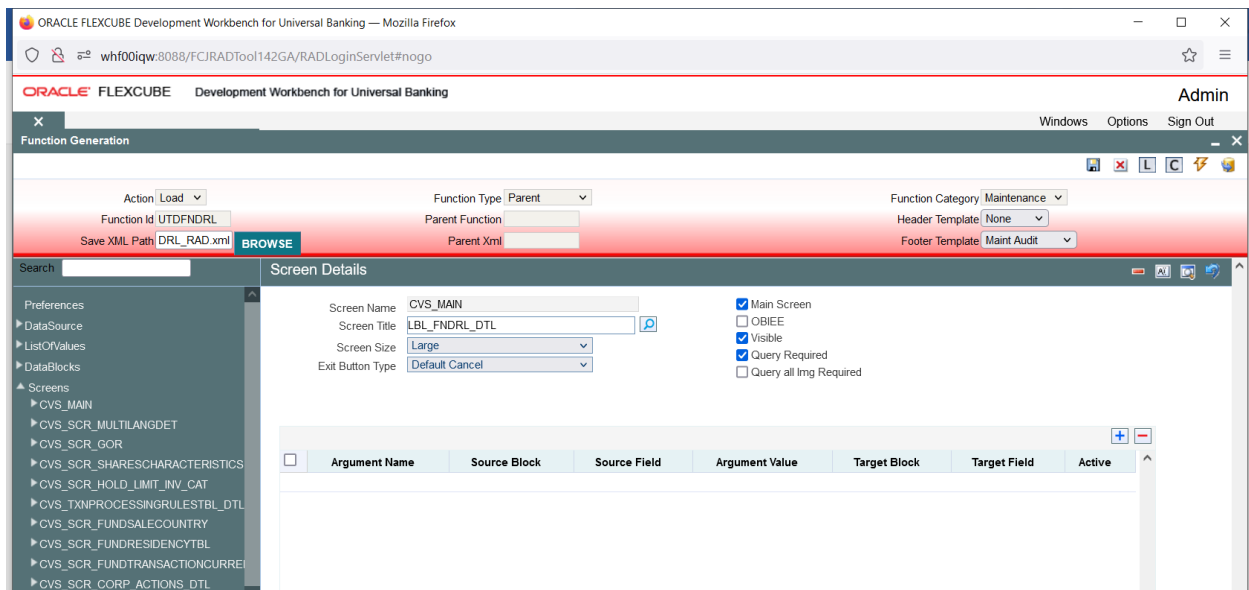


Fig: Providing properties to new Screen

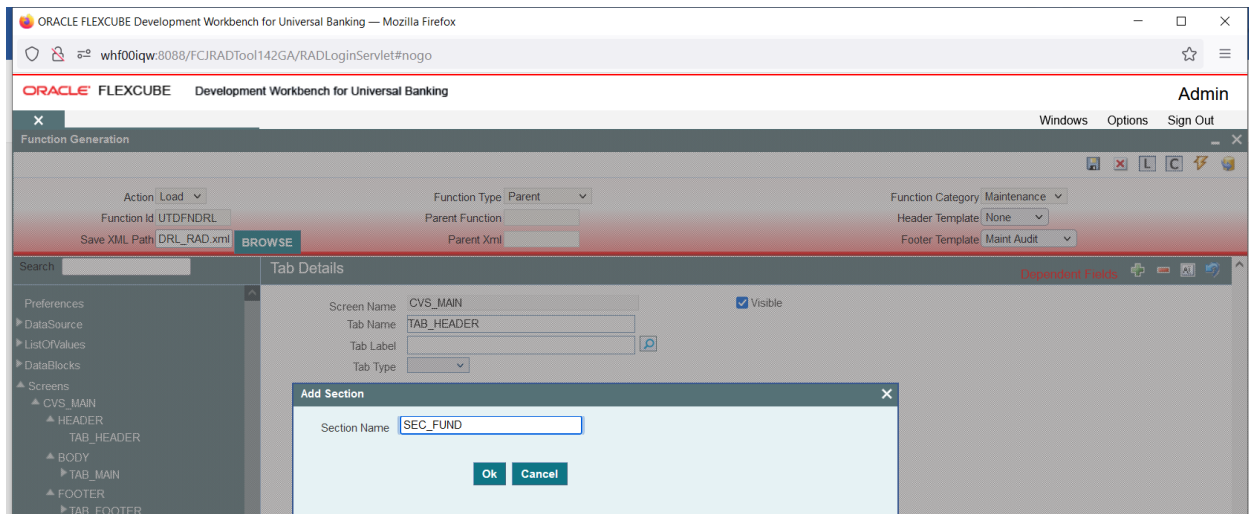


Fig: Creating new section in TAB_MAIN in the body of screen CVS_MAIN

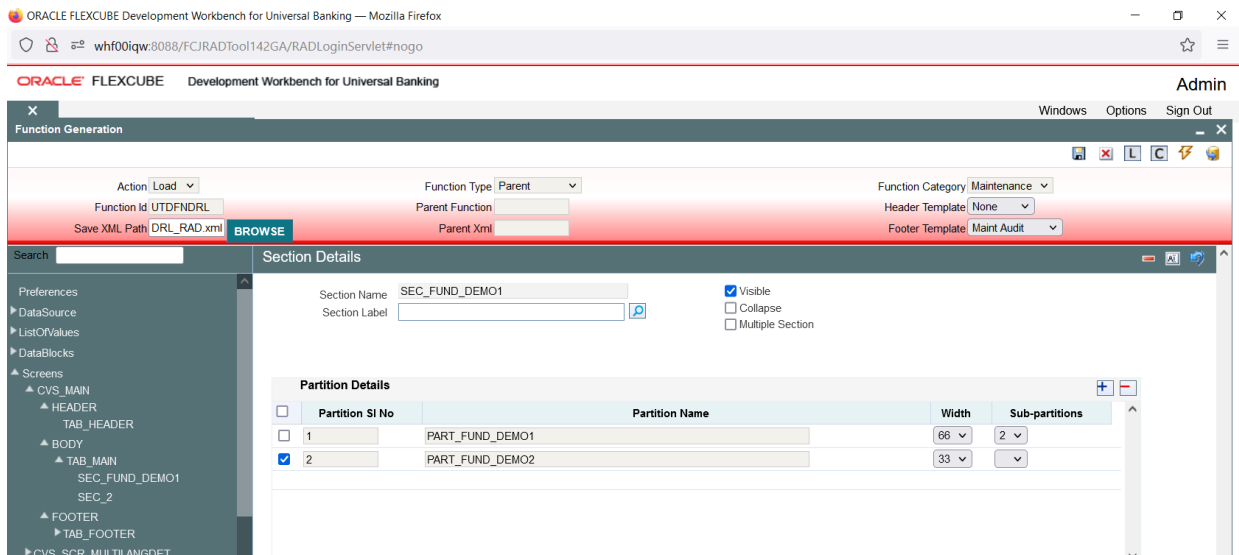


Fig: Defining partitions for the Section

4.6 Field Sets

A group of fields can be grouped together in a Field set which can be placed together in the screen

- Field Set Name should start with FST_<>.
- Select the Block adding to field set.
- All fields available to the block will be displayed in to the data block fields text area. Move fields from data block fields to Field set fields.
- The order of fields in *field set fields* will reflect in the screen as well

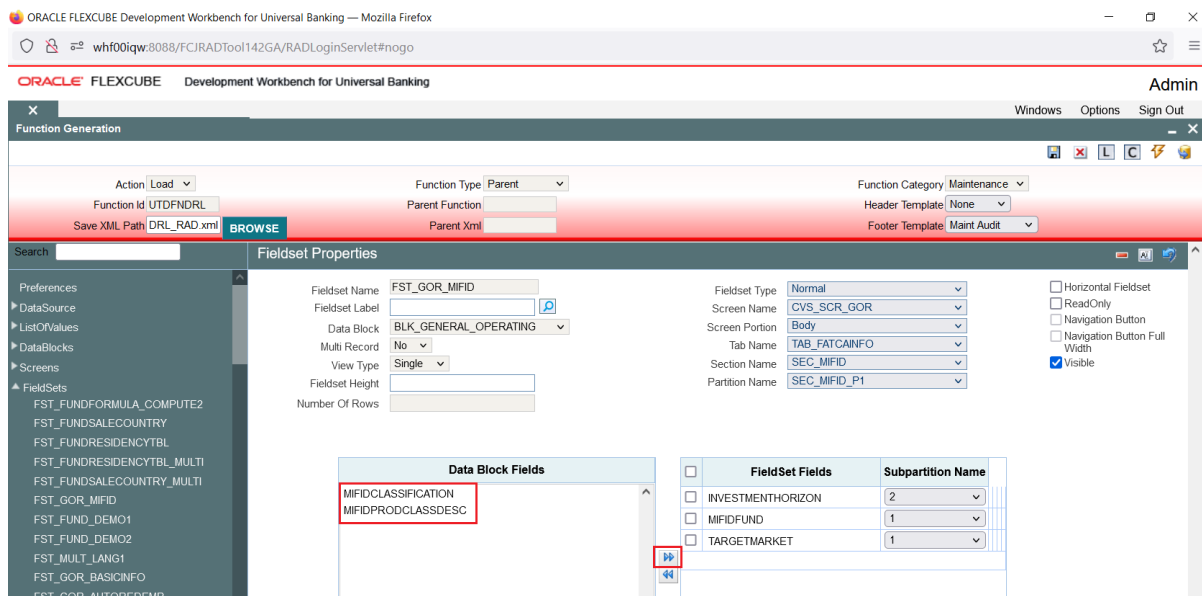


Fig: Attaching Fields to a Field set

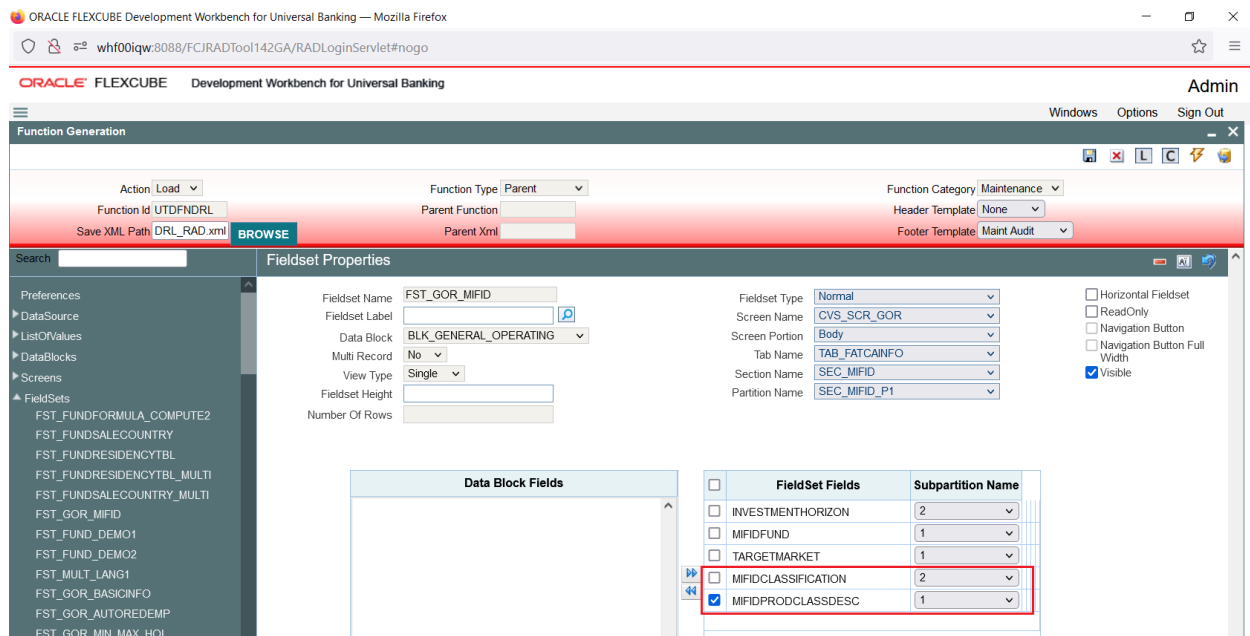


Fig: Order of fields in the field set highlighted

- Select the screen portion (Header/Body/Footer) where this field set has to be placed. Select remaining details like tab, section and partition.

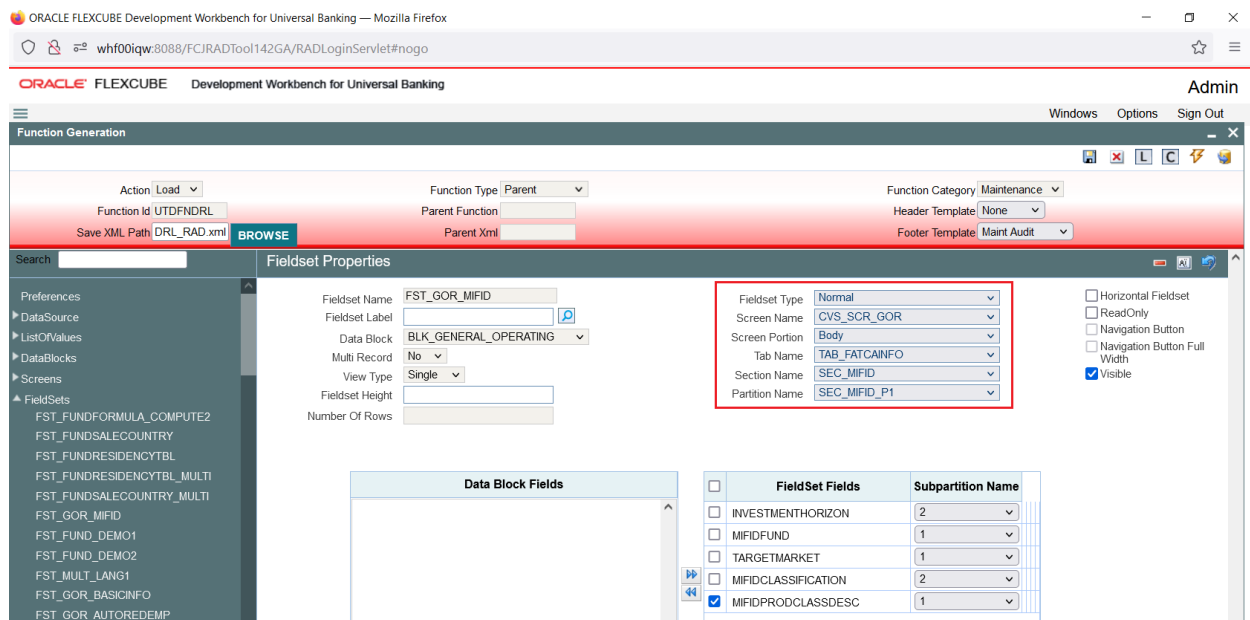


Fig: Providing details where Field Set has to be placed

Once fields are added to field set, developer can check the preview of the designed screen. Right click on Screen Name and click on Preview.

Fig: Preview of the designed Screen

Adding Multi entry block to field set.

- On selecting a multiple block, Multi Record Property will be defaulted to Yes.
- In case of Multi record, View type can be either Single or Multiple (By Default).

Below image shows a multiple view multi record field set

Fig: Multiple View Multi Record Field set

- For multi record single view navigation button should be checked.

The screenshot shows the Oracle Flexcube Development Workbench for Universal Banking. The 'Function Generation' window is open, displaying the 'Fieldset Properties' dialog. The 'Fieldset Name' is 'FST_TPR_TAB_BASIC_INFO'. The 'View Type' is set to 'Single'. The 'Navigation Button' checkbox is checked. Other properties include 'Fieldset Type: Normal', 'Screen Name: CVS_TXNPROCESSINGRULEST', 'Screen Portion: Body', 'Tab Name: TAB_BASIC_INFO', 'Section Name: SEC_1', and 'Partition Name: SEC1_PART1'.

Fig: Properties for Single View Multi Record Field set

Below figure shows the preview of a single view multi record field set

The screenshot shows the 'Transaction Processing Rules' window. The 'Basic info' tab is selected. The '1 of 1' indicator is visible in the top right corner. The window displays various fields for defining a transaction rule, including 'Fund ID', 'Rule Effective Date', 'Transaction Type', 'Transaction Frequency', 'Transaction Enabled?', 'Gross Amount', 'Gross Unit', 'Transaction Suspended?', 'Transaction Type', 'Description', 'Start Date', 'End Date', 'Net Amount', 'Net Unit', 'Restrict Sale to Investor', 'Maximum IPO Issue Size', and 'Allow Systematic Plans?'. The 'Ok' and 'Exit' buttons are at the bottom right.

Fig: Preview for Single View Multi Record Field set

4.7 LOV

List Of values can be defined for the function id using LOV node

- To add LOV right click on List of Values Node. LOV Name should start with LOV_<name>. *Example: LOV_FUNDCOUNTRY.*
- LOV Type - This field can be Null or Internal.
- Enter valid query and click on populate button.

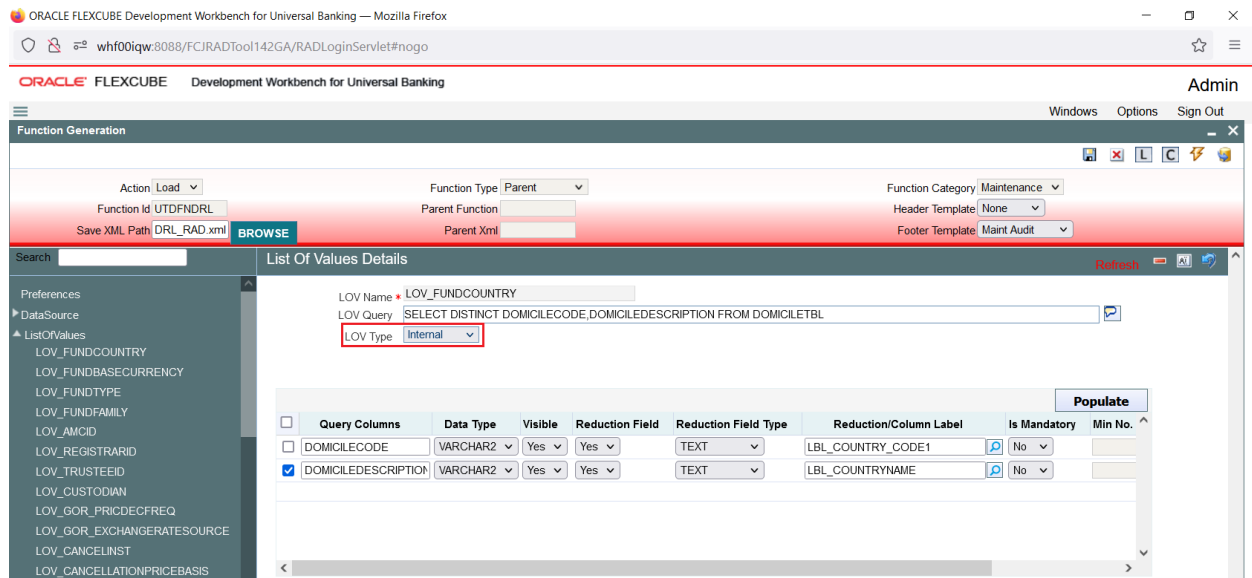


Fig: Defining new LOV

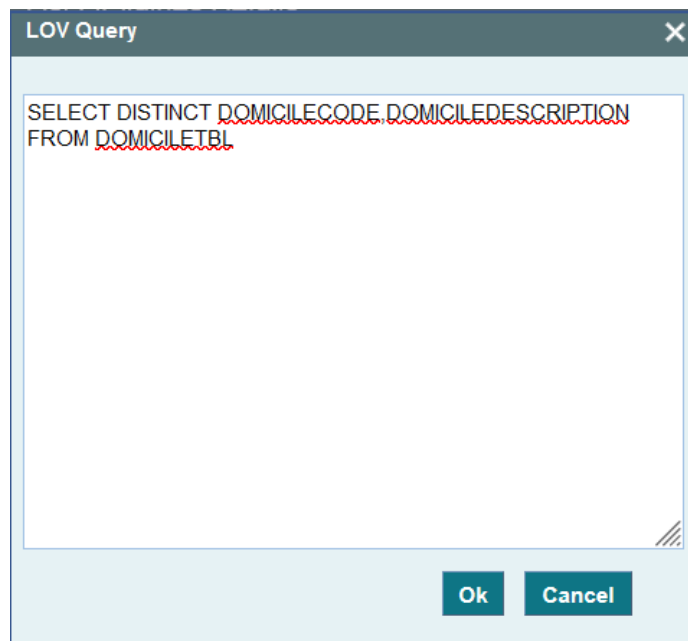


Fig: Providing LOV query

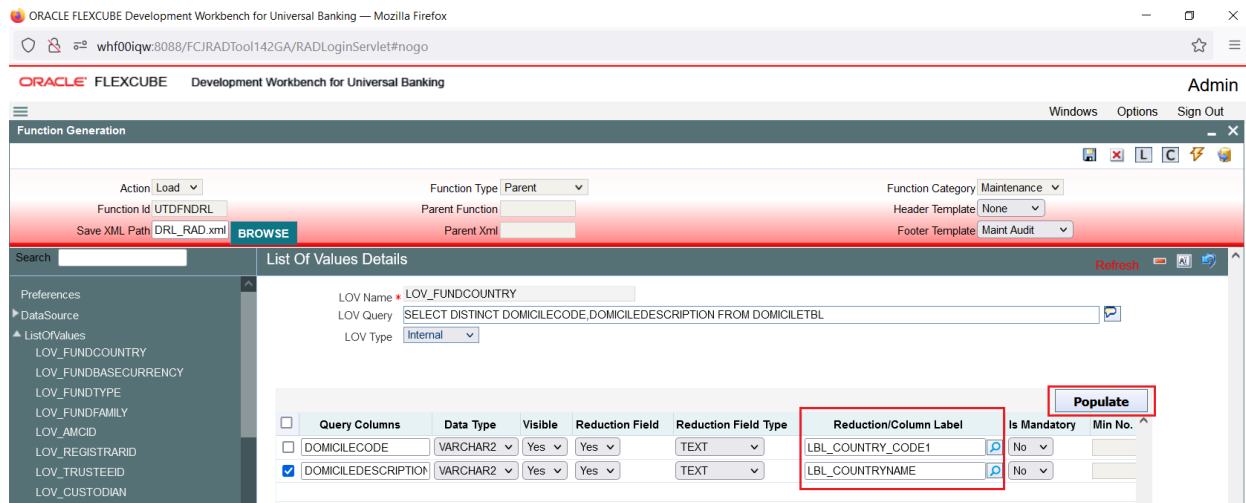


Fig: Providing LOV details

- Reduction/Column Labels are mandatory. If user won't provide will get error on click of LOV button after deployment in FLEXCUBE
- After defining LOV go to block and corresponding field where the LOV has to be attached.

Block Field Properties to attach LOV to the field

- **Display Type:** Select display type as Lov.
- **Lov Name:** Select the required Lov name from the list of all defined LOV's.
- Click on return fields tab. The result fields maintained in the LOV query will be populated on click of **Default from Lov Definition** button
- Select the desired field (and its block)to which the result of the LOV query should be defaulted
- If return field is not required to be defaulted to any field in the screen, return field value can be left blank.

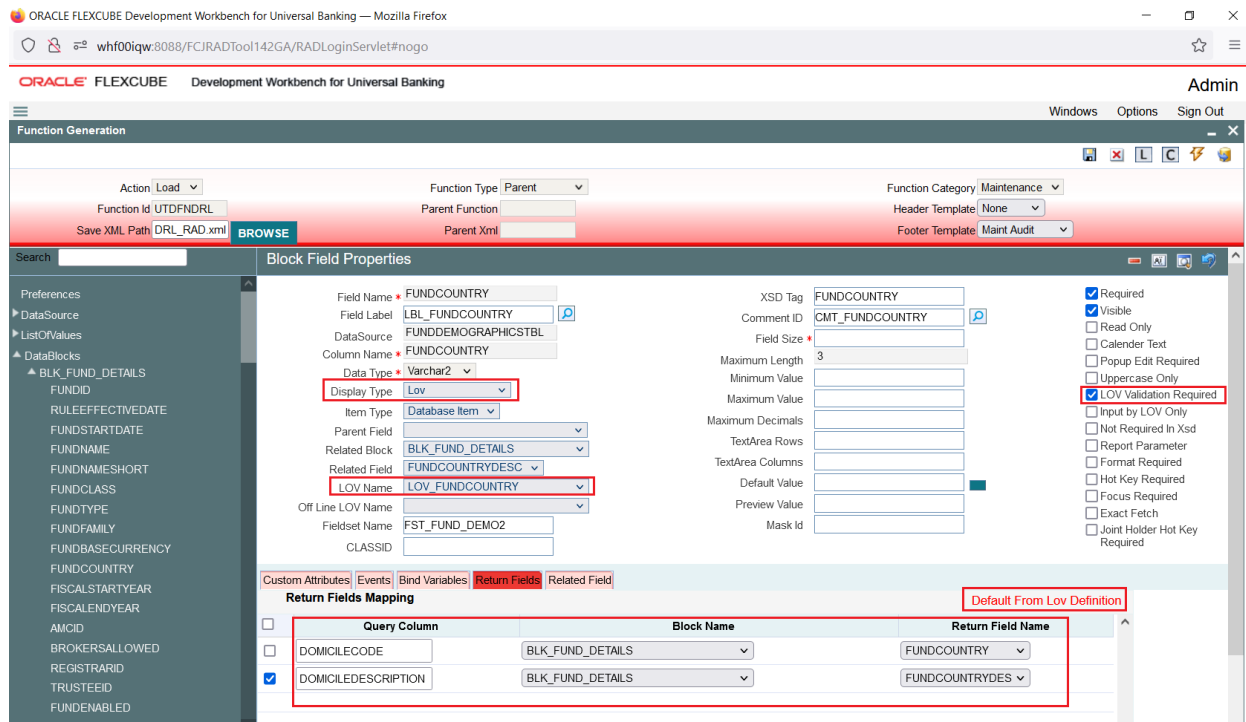


Fig: Attaching LOV to a block Field

Use of Bind Variable

If the list of values should be based on any other field value from the screen, bind variables can be used.

Example:

Define lov as shown in below query; where clause should contain condition with '?'.

```
SELECT cust_ac_no, branch_code, ccy
from sttms_cust_account
where cust_no = ?
and record_stat = 'O'
and once_auth = 'Y'
and ac_stat_de_post = 'Y'
```

In the block field, after selecting return fields, click on bind variables tab. Click on **Default from Lov Definition** button. New rows will be created depending on the number of bind variable provided in the LOV query. Select the bind filed in the screen (and its block) for the LOV. Data type of the field has also to be selected.

ORACLE FLEXCUBE Development Workbench for Universal Banking — Mozilla Firefox

whf001qwr:8088/FCJ/RADTool142GA/RADLoginServlet#nogo

ORACLE FLEXCUBE Development Workbench for Universal Banking

Admin

Function Generation

Action: Load

Function Type: Parent

Function Category: Maintenance

Function Id: UTDNDRL

Parent Function:

Header Template: None

Save XML Path: DRL_RAD.xml

Parent Xml:

Footer Template: Maint Audit

Search:

Block Field Properties

Field Name: BTN_UMBFNDLOV

Field Label: LBL_UMB_FUND

Data Source:

Column Name:

Data Type: Varchar2

Display Type: Lov

Item Type: Control

Parent Field:

Related Block:

Related Field:

LOV Name: LOV_UMBFUND

Off Line LOV Name:

Fieldset Name: FST_FUND_DEMO1

CLASSID:

XSD Tag: UMB_FUND

Comment ID:

Field Size:

Maximum Length:

Minimum Value:

Maximum Value:

Maximum Decimals:

TextArea Rows:

TextArea Columns:

Default Value:

Preview Value:

Mask Id:

Required: ☐

Visible: ☒

Read Only: ☐

Calendar Text: ☐

Popup Edit Required: ☐

Uppercase Only: ☐

LOV Validation Required: ☐

Input by LOV Only: ☒

Not Required In Xsd: ☐

Report Parameter: ☐

Format Required: ☐

Hot Key Required: ☐

Focus Required: ☐

Exact Fetch: ☐

Joint Holder Hot Key Required: ☐

Custom Attributes: Events: Bind Variables: Return Fields: Related Field

Bind Variables Mapping

Block Name	Bind Variable	Data Type
BLK_FUND_DETAILS	UMBRELLAFUND	NUMBER

Fig: Defining bind variable for the LOV

4.8 Attaching Call forms

Maintenance Call forms can be attached to a maintenance screen. Refer the document [15-Development of Call Form.docx](#) for developing call forms

Attaching Call forms

- Add button to block to launch call form on button click.
- Right click on Block
- Select Add fields. Select fields and Add UI field's window will be launched
- Select UI Fields tab. Click add row button. Enter button name and click ok.
- Select display type as button and enter field label.

Select Fields & Add UI Fields

DataSource fields UI Fields

	Field Name	Data Type
<input checked="" type="checkbox"/>	BTN_FPCDET	

Ok Cancel

Fig: Defining Button field

- Add Call form details to Call form node

ORACLE FLEXCUBE Development Workbench for Universal Banking — Mozilla Firefox

whf00iqw:8088/FCJRADTool142GA/RADLoginServlet#nogo

ORACLE FLEXCUBE Development Workbench for Universal Banking

Admin

Function Generation

Action Load

Function Id UTDFPRIC

Save XML Path RIC_RAD.xml

BROWSE

Function Type Parent

Parent Function

Parent Xml

Function Category Maintenance

Header Template None

Footer Template Maint Audit

Search

Call Form Details

Screen Arguments Dependent Fields

Function ID	Parent Data Block	Parent DataSource	Relation	Relation Type
<input checked="" type="checkbox"/> UTDFCOMP	BLK_FUNDPRICECOMPONENTTBL	FUNDPRICECOMPONENTTBL	COMPONENTTBL.FUNDID = FUNDDEMO	One To One

Preferences

DataSource

ListOfValues

DataBlocks

BLK_FUNDPRICEHRTBL

BLK_FUNDPRICEDTAILTBL

BLK_FUNDPRICECOMPONENTTBL

BLK_VW_ADDINFO

BLK_SUMMARY

BLK_FPC_HDR

BLK_FPC_DTL

BLK_FPC_COMP

Screens

FieldSets

Actions

CallForms

LaunchForms

Summary

Relation	Relation Type	Callform Screen	Display Type	Active	Rest Relation With Parent
ENTTB.L.FUNDID = FUNDD.DEMO	One To One		Button	Yes	

Fig: Defining details of the Call form to be attached in call form node

- Add event to button.
- On selecting event type as call form or launch form or sub screen button will be displayed on bottom of the screen.
- If user needs to place button position in desired place on the screen, event type should be Normal .User has to write code in release specific JavaScript file to launch the screen.

ORACLE FLEXCUBE Development Workbench for Universal Banking — Mozilla Firefox

whf001qw:8088/FCJ/RADTool142GA/RADLoginServlet#nogo

ORACLE FLEXCUBE Development Workbench for Universal Banking

Admin

Function Generation

Action: Load

Function Id: UTDFFPRIC

Save XML Path: 'RIC_RAD.xml'

Function Type: Parent

Parent Function:

Parent Xml:

Function Category: Maintenance

Header Template: None

Footer Template: Maint Audit

Block Field Properties

Field Name: BTN_FPCDET

Field Label: LBL_FPCDTL

Data Source:

Column Name:

Data Type:

Display Type: Button

Item Type:

Parent Field:

Related Block:

Related Field:

LOV Name:

Off Line LOV Name:

Fieldset Name:

CLASSID:

XSD Tag: FPCDTL

Comment ID:

Field Size:

Maximum Length:

Minimum Value:

Maximum Value:

Maximum Decimals:

TextArea Rows:

TextArea Columns:

Default Value:

Preview Value:

Mask Id:

Required:

Visible:

Read Only:

Calendar Text:

Popup Edit Required:

Uppercase Only:

LOV Validation Required:

Input by LOV Only:

Not Required In Xsd:

Report Parameter:

Format Required:

Hot Key Required:

Focus Required:

Exact Fetch:

Joint Holder Hot Key Required:

Custom Attributes: Events

Event Name	Function Name	Event Type	Button Screen	CallForm Name	Screen Name
onlick	fnshowfpcdetails()	Subscreen	CVS_SCR		CVS_FPC

Fig: Defining event to the button such that call form is linked to the button

- Check the preview.

Fig: Preview of the screen with the Call Form button

4.9 Adding Summary

- 1) Add entry in Preferences node for Summary screen

Fig: Adding Summary screen details in Preferences node

2) Click on Summary Node.

- Enter Summary title .Select label code from lov.
- Select Data Block master block and summary blocks will be displayed. Select required block from drop down list.
- Select Data Source for summary.
- Select Summary Type.
- Select Summary Screen size.
- Enter if anywhere clause is required.
- Enter Default order by if required.
- Enter Multi Branch where clause if required.
- Attach the fields required in the summary result grid
- If the field is required as part of filtering, query has to be checked for the particular field
- Provide the position of fields in Result grid and Summary Query set.

ORACLE FLEXCUBE Development Workbench for Universal Banking — Mozilla Firefox

whf001qw:8088/FCJRADTool142GA/RADLoginServlet#nogo

ORACLE FLEXCUBE Development Workbench for Universal Banking

Admin

Windows Options Sign Out

Function Generation

Action: Load

Function Type: Parent

Function Category: Maintenance

Function Id: UTDFNDRL

Parent Function:

Header Template: None

Save XML Path: DRL_RAD.xml

Parent Xml:

Footer Template: Maint Audit

Search:

Summary Details

Title: LBL_FUNDRULESSUMM

Data Blocks: BLK_SUMMARY

Data Source: VW_FDR_SUMMARY

Summary Type: Bulk Authorization

Summary Screen Size: Medium

☐ Criteria Based Search

Default Where Clause:

Default Order By:

Multi Branch Where Clause:

Main Summary Screen:

☐ WebServices Required

Data Block Fields Custom Buttons Fields Ordering

Data Block Fields

Fields Selected	Query	Properties
<input type="checkbox"/> STATUS	<input type="checkbox"/>	Properties
<input type="checkbox"/> FUNID	<input checked="" type="checkbox"/>	Properties
<input type="checkbox"/> RULEEFFECTIVEDATE	<input checked="" type="checkbox"/>	Properties
<input type="checkbox"/> FUNNAME	<input checked="" type="checkbox"/>	Properties
<input type="checkbox"/> FUNDNAMESHORT	<input type="checkbox"/>	Properties
<input type="checkbox"/> FUNDCASS	<input type="checkbox"/>	Properties

Fig: Providing Properties for Summary Screen

Summary Preview

Right click on summary node and click on preview.

Fund Rules Summary

Execute Query Advanced Search Reset Clear All

Authorization Status Record Status

Fund ID Rule Effective Date

Fund Name Fund Type

AMC ID Fund Enabled?

Latest Rule Fund Identification Number

Fund Ticker Symbol Umbrella Fund

Hedge Fund

Records per page 15 1 of 1 Go to Page

<input type="checkbox"/>	Authorization Status	Record Status	Status	Fund ID	Rule Effective Date	Fund Name	Fund Name Short	Fund Clas
<input type="checkbox"/>								
<input type="checkbox"/>								
<input type="checkbox"/>								
<input type="checkbox"/>								
<input type="checkbox"/>								
<input type="checkbox"/>								
<input type="checkbox"/>								

Authorize View Changes

Exit

Fig: Summary Screen Preview

4.10 Amendable fields Maintenance

Amendable Fields

If user needs to modify data of a particular field on unlock, in Workbench developer has to maintain fields as amendable.

- Click ACTIONS node.
- Click on *Amendables* button next to the action for which the field has to be made amendable
- Select the fields in each block which user can modify for the selected action.

Amendable Details --QUERY

Data Blocks

BLK_FUND_DETAILS

BLK_GENERAL_OPERATING

BLK_MULTI_LANGUAGE_DTL

BLK_SHARES_CHARACTERISTICS

BLK_HOLDING_LIMIT

BLK_HOLDING_LIMIT_DETAILS

BLK_TXN_PROCESSING_RULES

BLK_TXN_LIMIT_INVESTOR

BLK_TXN_LIMIT_COMMUNICATION

BLK_SI_RULES_FREQUENCY

BLK_TXN_RULES

BLK_FUND_SALE_COUNTRY

BLK_FUND_SALE_COUNTRY_DETAIL

BLK_FUND_RESIDENCY_RESTRICTION

BLK_FUND_RESIDENCY_RESTRICTION_DETAILS

BLK_FUND_TRANSACTION_CURRENCY

BLK_FUND_TRANSACTION_CURRENCY_DETAILS

BLK_CORPORATE_ACTION

BLK_FUND_LOAD

DataBlock Fields

☐ New Allowed
☐ Delete Allowed
☐ All Records
☐ Mandatory

Field Name	Amendable
FUNDID	<input type="checkbox"/>
RULEEFFECTIVEDATE	<input type="checkbox"/>
FUNDSTARTDATE	<input type="checkbox"/>
FUNDNAME	<input type="checkbox"/>
FUNDNAMESHORT	<input checked="" type="checkbox"/>
FUNDCLASS	<input type="checkbox"/>
FUNDTYPE	<input type="checkbox"/>
FUNDFAMILY	<input type="checkbox"/>
FUNDBASECURRENCY	<input checked="" type="checkbox"/>
FUNDCOUNTRY	<input type="checkbox"/>
FISCALSTARTYEAR	<input type="checkbox"/>
FISCALENDYEAR	<input checked="" type="checkbox"/>
AMCID	<input type="checkbox"/>
BROKERSALLOWED	<input type="checkbox"/>

Ok

Cancel

Fig: Maintaining amendable fields

5. Generation of files

Generate Files

- Click on generate button select the required files to generate and click on Generate button.

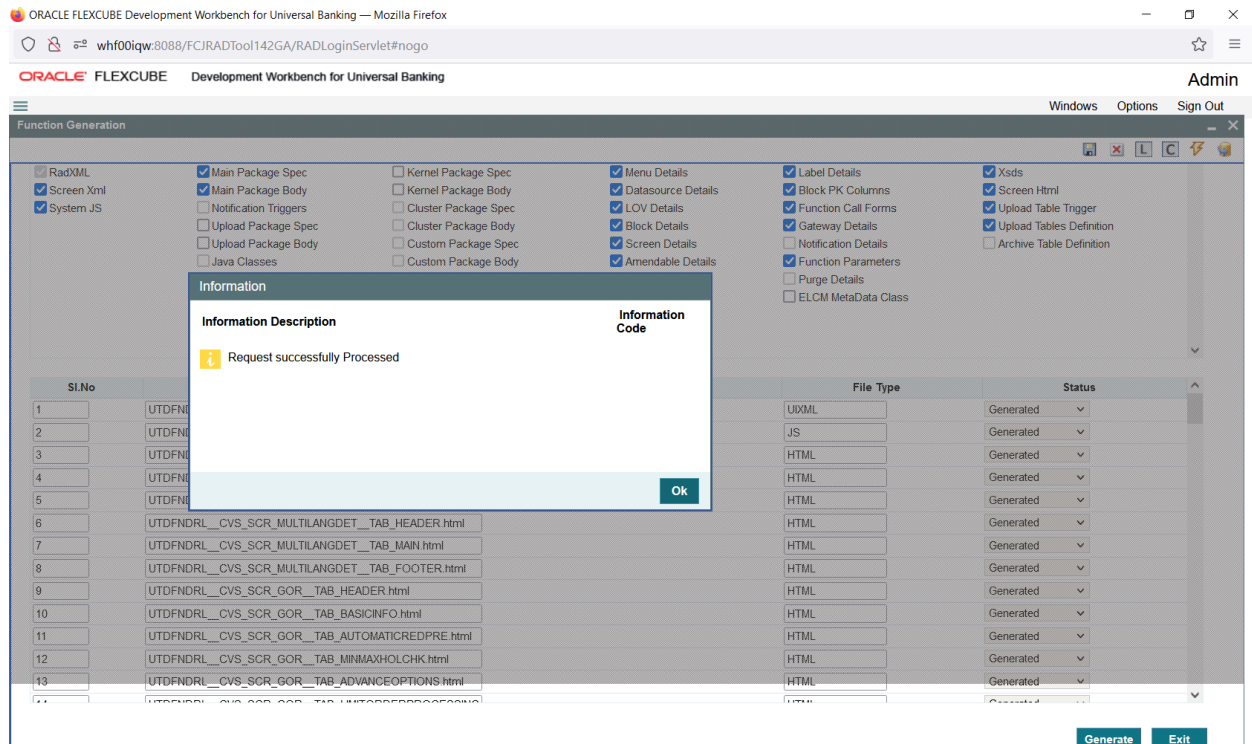


Fig: Generation of Files

Testing

- Launch the screen from FLEXCUBE
- Try sample operations on the screen (NEW,MODIFY,QUERY etc.)

BATMAKER2 - 000 - FMGRSADISTB - 14.6.0.1.0 - ENG - Online — Mozilla Firefox

https://100.76.155.96:7006/FCISNeoWebSwift/SMMDDIFRM.jsp

ORACLE®

Module: FMGRSADISTB 2015-01-06 Window Default Admin User1 UTDNDRL

Home

Fund Rules Detail

New Copy Delete Unlock Print Authorize Enter Query

Fund ID * FUND1

Rule Effective Date * 2015-01-06

Fund Start Date * 2015-01-06

Fund Name * Close Ended Fund

Fund Name Short * FUND1

Fiscal Year Start Date * 2014-03-01

Fiscal Year End Date * 2015-02-28

Primary Registration Address * Granger Bay Court Bea

Fund Identification Number FUND11SIN1

Fund Base Currency * ZAR

Fund Base Currency Description South Africa Rands

Min Amt for Tax ID-Indv * 999,999,999,999.000

Min Amt for Tax ID-Corp * 999,999,999,999.000

Existing Fund? No

Fund Class * Close-Ended

AMC * TEMPLETON

AMC Description Templeton investments

Legal Entity

Legal Entity Description

Brokers Mandatory? * No

Fund Type * 5

Fund Type Description IT Sector Fund

Fund Family * AG

Fund Family Description W2W

Fund Country * RSA

Fund Country Description SOUTH AFRICA

Registrar * KARVY

Registrar Description KARVY REG

Trustee * CITI

Trustee Description CITI TRUST

Custodian * SANLAM

Custodian Description Sanlam Collective Inves

Hedge Fund No

Termination Date

Fund Domicile Onshore

Information Message

Information Message

Record Successfully Saved

Ok

Multi Language Details | Shares Characteristics | General Operating Rules | Transaction Processing Rules | Fund Restriction | Fund Transaction Currencies | Corporate Actions | Fund Load |

Fund Formula Setup | Additional Information | Specific Fund Price Date | Amount Limit for IRA Transactions | Fund Delivery Instructions | Fund Investment Account | SI Specific Information |

KYC Maintenance | Documents | Hedge Funds | Fund Price Currencies

Input by BATMAKER2

Authorized by

DateTime 2015-01-06 07:23:58

DateTime

Mod No 1

Record Status Open

Authorization Status Unauthorized

Exit

Fig: Saving Record for the function id in FLEXCUBE

6. Generated Units

The following units will be generated for a Maintenance screen.

Refer document [Development WorkBench_Screen_Development-II.docx](#) for detailed explanation on the same.

6.1 Front End Units

6.1.1 Language xml

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

6.1.2 SYS JavaScript File

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

6.1.3 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.

6.2 Data Base Units

6.2.1 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

- i) Menu Details
Scripts for SMTB_MENU and SMTB_FCC_FCJ_MAPPING, SMTB_ROLE_DETAIL, SMTB_FCC_GCJ_MAPPING are required for the functioning of Maintenance screen
- ii) Lov Details
- iii) Amendable Details
- iv) Label details
- v) Screen Details
- vi) Block details
- vii) Data Source Details
- viii) Call form details

6.2.2 **System Packages**

The Main Package contains the basic validations and backend logic for the Maintenance function id. The Main package contains the mandatory checks required. It will also contain function calls to the other packages generated by Workbench.

The main package has the below stages for a maintenance form:

- Converting Ts to PL/SQL Composite Type
- Checking for mandatory fields
- Defaulting and validating the data
- Writing into Database
- Querying the Data from database
- Converting the Modified Composite Type again to TS

Each of these stages has a 'Pre' and 'Post' hooks in the Kernel, Cluster and Custom Packages.

And these Hooks are called from the Main Package itself

Main Package has the system-generated code and should not be modified by the developer
Kernel, Cluster and Custom Packages are the packages where the respective team can add business logic in appropriate functions using the Pre and Post hooks available.

6.2.3 **Hook Packages**

Release specific packages will be generated based on the release type (KERNEL.CLUSTER or CUSTOM). Developer can add his code in the release specific hook package.

The Main Package has designated calls to these Hook Packages for executing any functional checks and Business validations added by the user. The structure for all the Hook Packages are the same, like:

Fn_Post_Build_Type_Structure

Fn_Pre_Check_Mandatory

Fn_Post_Check_Mandatory

Fn_Pre_Default_and_Validate

Fn_Post_Default_and_Validate

Fn_Pre_Upload_Db

Fn_Post_Upload_Db

Fn_Pre_Query

Fn_Post_Query

These Functions are called from the Main package using the Pre and Post Hooks available in the Main Package. The 3 Hook Packages namely Kernel, Cluster and Custom Packages have similar structure and are for the respective teams to work on.

6.3 **Other Units**

6.3.1 **Xsd**

Xsd's will be generated if gateway operations are required for the particular function id. Maintenance for the same has to be done in ***Actions*** node.

7. Extensible Development

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

7.1 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 `UTDFNDRL_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()`

7.2 Extensibility in Backend Coding

Release specific code has to be written in the Hook Packages generated.

7.2.1 Functions in Hook Packages

Different functions available in the Hook Package of a Maintenance 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_post_bulid_type_structure

If any change has to be made in the field values obtained from the form before start of processing, code can be written here

3) Fn_pre_check_mandatory

4) Fn_post_check_mandatory

Any extra mandatory checks on the field values from the screen can be written here.

5) Fn_pre_query

6) Fn_post_query

Any specific logic while querying can be written in these functions. It is called from fn_query of the main package.

7) Fn_pre_upload_db

8) Fn_post_upload_db

Any logic while uploading data to tables can be written here.

9) Fn_pre_default_and_validate

10) 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.

7.2.2 Flow of control through Hook packages

The flow of control through the Hook Packages for a particular stage is as explained in the figure below

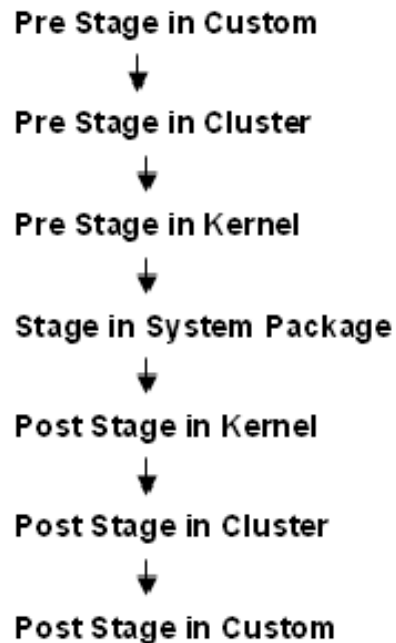
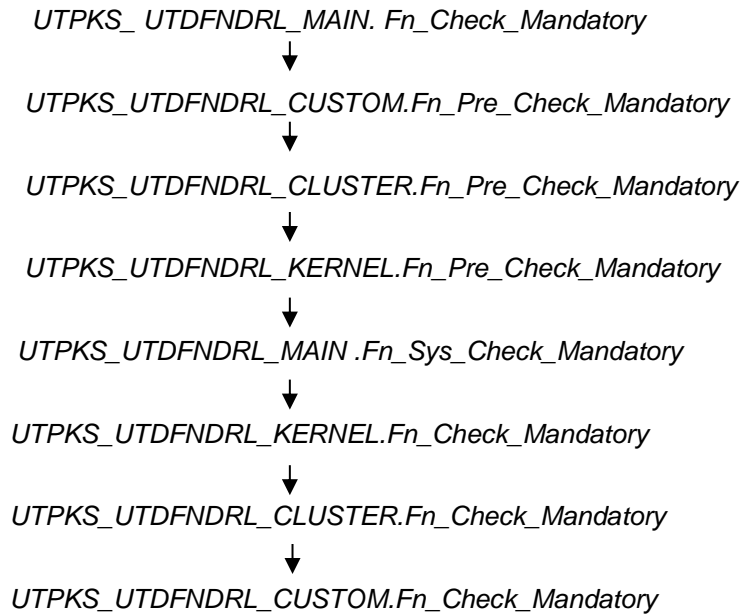


Fig: Flow of control through Hook Packages

Example: For *Fn_check_mandatory*, flow will be as



7.2.3 By passing Base Release Functionality

There are auto generated functions like *FN_SKIP_<RELEASE_TYPE>* which would determine whether or not a particular hooks needs to be called.

Developer also has an option to bypass the base release hook if need be. For example if the validations written in *UTPKS_UTDFNDRL_KERNEL.FN_PRE_CHECK_MANDATORY* are not required or not suitable for the Cluster release, system provides an option to bypass the code written by Kernel team. Similarly a Custom release can also bypass the code written by Kernel and Custom Releases. This can be achieved by calling procedures

PR_SET_SKIP_<RELEASE_TYPE> and ***PR_SET_ACTIVATE_<RELEASETYPE>***. These procedures will be made available in the main package and the development teams of Customization teams can use these procedures to skip and re-activate the hooks of parent release.

The Developer should avoid adding validations or Checks in the Pre Stage of any function, like *Fn_Pre_Check_Mandatory*, etc and should aim to add all the validations in the *Fn_Post_Default_and_Validate*.

For Example let us see the flow for the Mandatory Stage for STDCIFD:

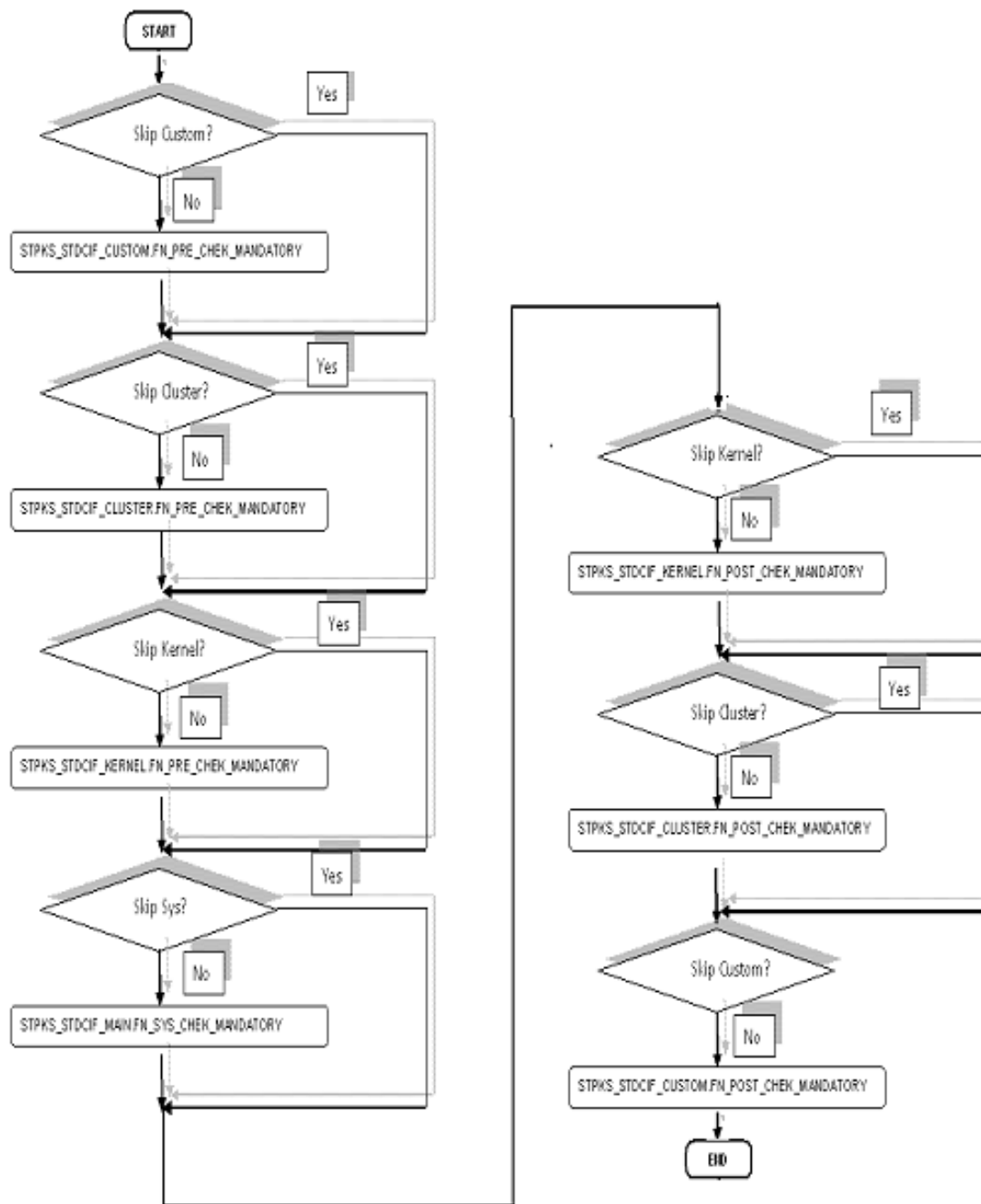


Fig: Flow of control explaining skip logic in packages



Development of Maintenance Form
[May] [2023]
Version 14.7.0.1.0

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/

Copyright © [2007], [2023], Oracle and/or its affiliates.

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

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

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.