

Development of Maintenance Form
Oracle Banking Treasury Management
Release 14.7.2.0.0

[November] [2023]



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.....	4
4.	Screen Development	4
4.1	Header Information	4
4.2	Preferences	6
4.3	Data Sources	8
4.4	Data Blocks	12
4.5	Screens	14
4.6	Field Sets	15
4.7	LOV	18
4.8	Attaching Call forms	20
4.9	Adding Summary	22
4.10	Amendable fields Maintenance	22
5.	Generation and Deployment of files	23
6.	Generated Units	23
6.1	Front End Units	23
6.1.1	Language xml	23
6.1.2	SYS JavaScript File	23
6.1.3	Release Type Specific JavaScript File	23
6.2	Data Base Units	23
6.2.1	Static Scripts	23
6.2.2	System Packages	24
6.2.3	Hook Packages	24
6.3	Other Units	25
6.3.1	Xsd	25
7.	Extensible Development	25
7.1	Extensibility in JavaScript Coding	25
7.2	Extensibility in Backend Coding	25
7.2.1	Functions in Hook Packages	25
7.2.2	Flow of control through Hook packages	26
7.2.3	By passing Base Release Functionality	27

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>Development Workbench - Screen Development I</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

[*Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development*](#)
[*Development Workbench - Screen Development II*](#)

2. Introduction

2.1 How to use this Guide

The information in this document includes:

- [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. STDCINF is sample function id used for demonstration in this document

For detailed explanation, refer the document: [Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development](#)

4.1 Header Information

Provide the header information as shown in the figure.

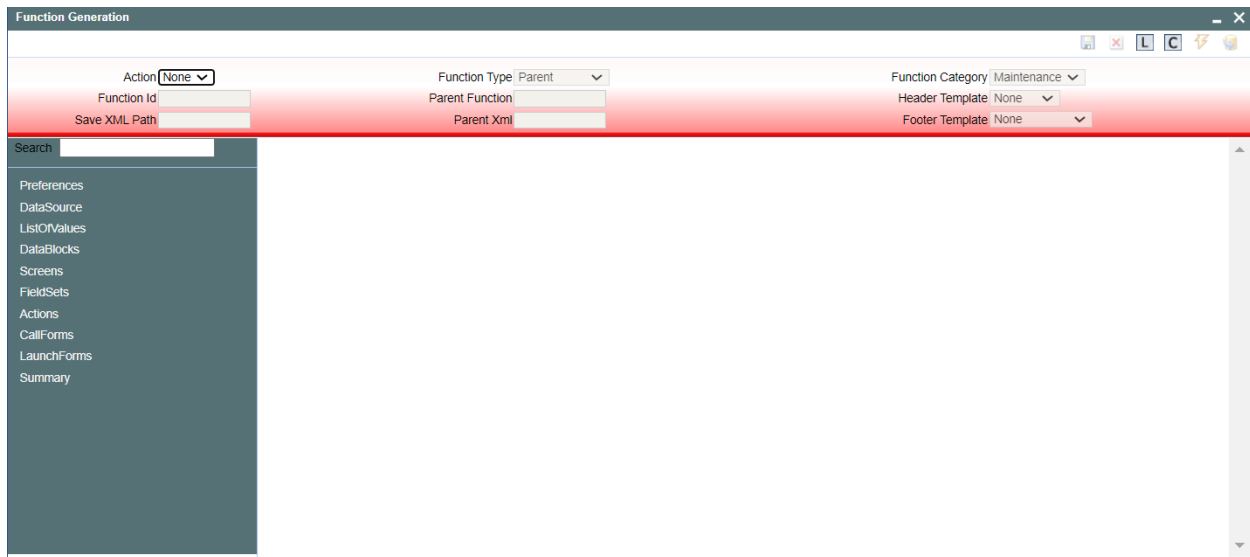


Fig 12.1: Providing Header Information for Maintenance Screen

- For new screen select action As New.
- Enter Function ID → STDCIF
- 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

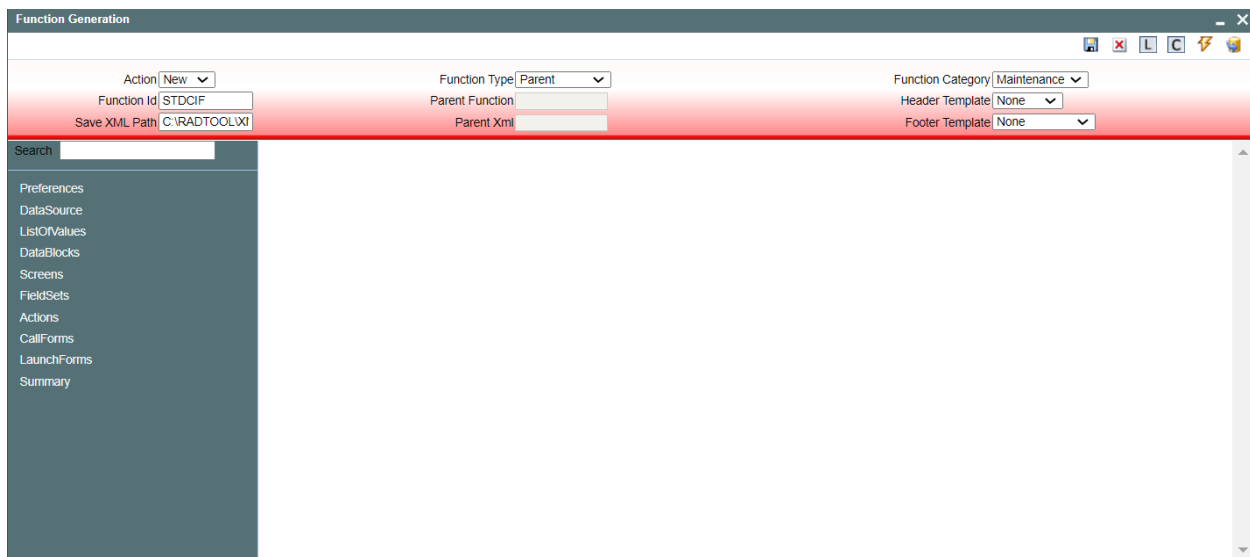


Fig 12.2: Save icon used for saving the radxml

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

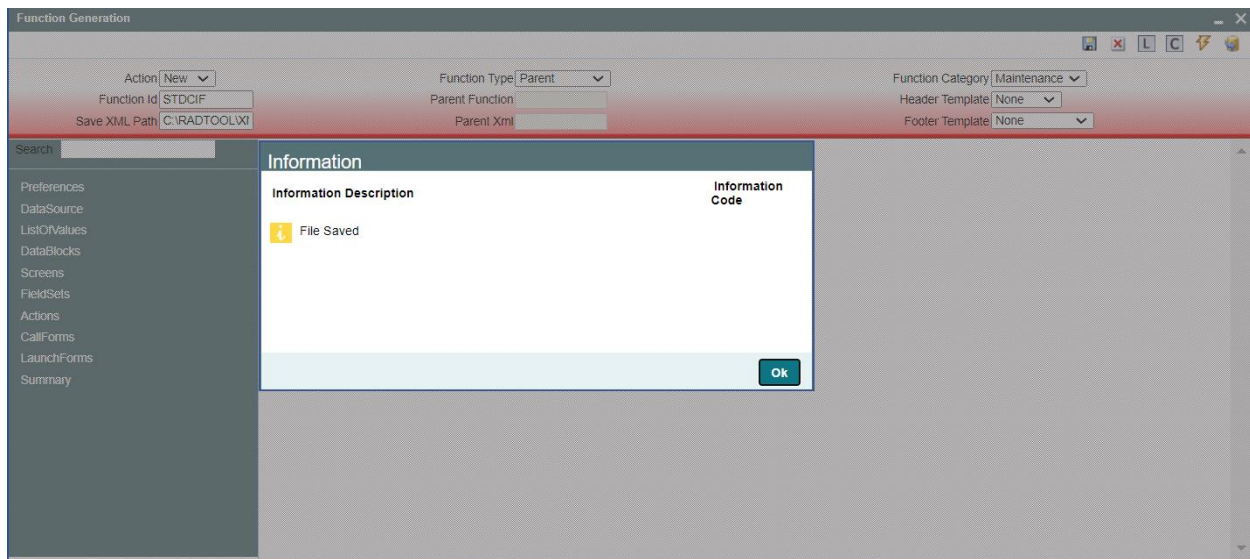


Fig 12.3: Saved File Information page

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 [Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development](#) 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.

Function Generation

Action: New | Function Id: STDCIF | Function Type: Parent | Function Category: Maintenance | Header Template: None | Footer Template: None

Save XML Path: C:\RADTOOL\W\

Search: []

Preferences

- ☒ Head Office Function
- ☒ Logging Required
- ☒ Auto Authorization
- ☐ Module Auto Authorization
- ☒ Tank Modifications
- ☒ Field Log Required
- ☐ Multi Branch Access
- ☐ Excel Export Required
- ☐ Java Functions
- ☐ GateWay Screen

Module: ST | Module Description: Static Maintenance

Module Group: [] | Branch Program Id: [] | Process Code: [] | SVN Repository URL: []

Menu Details

Parameter Value Mappings

Function Id	Module	Control String
☒ STDCIF	ST	Static Maintenance

Fig 12.4: 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.
- ii) **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 STDCIF, following labels has to be maintained
LBL_STDCIF_MAIN_MENU, LBL_STDCIF_SUB_MENU_1,
LBL_STDCIF_SUB_MENU_2, LBL_STDCIF_DESC

Refer [Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development](#) 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.

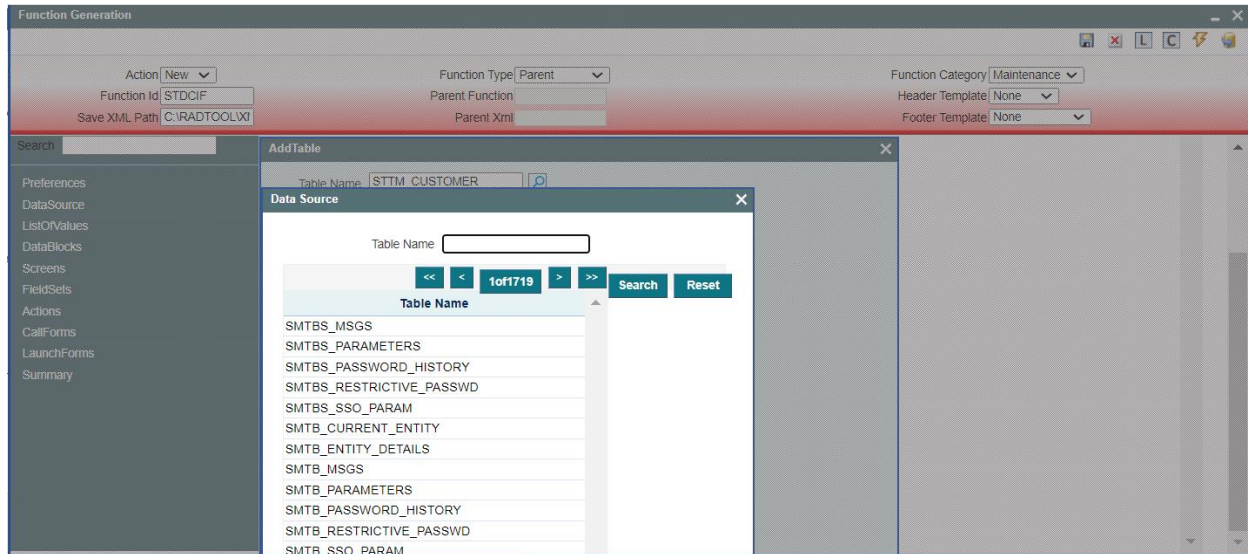


Fig 12.5: Adding Data Sources for the Function id

- 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.

The screenshot shows the 'Function Generation' window with the 'Data Source Details' tab selected. The 'Data Source' is 'STTM_CUSTOMER'. The 'Master' checkbox is checked. The 'Relation Type' is 'One To One'. The 'Multi Record' checkbox is unchecked. The 'PK Cols' is 'CUSTOMER_NO'. The 'PK Types' is 'VARCHAR2'. The 'Upload Table' is empty. The 'Upload Where Clause' is empty. The 'Fine Grained Service Details Only' section is expanded, showing 'Rest Data Block', 'XSD Node', 'Rest Relation', 'Rest Relation Type', 'Rest PK Cols', 'Rest PK Types', 'Parent', and 'Relation' fields.

Fig 12.6: 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.

The screenshot shows the 'Function Generation' window with the 'Data Source Details' tab selected. The 'Data Source' is 'STTM_CUSTOMER'. The 'Master' checkbox is checked. The 'Relation Type' is 'One To One'. The 'Multi Record' checkbox is unchecked. The 'PK Cols' is 'CUSTOMER_NO'. The 'PK Types' is 'VARCHAR2'. The 'Upload Table' is empty. The 'Upload Where Clause' is empty. The 'Fine Grained Service Details Only' section is expanded, showing 'Rest Data Block', 'XSD Node', 'Rest Relation', 'Rest Relation Type', 'Rest PK Cols', 'Rest PK Types', 'Parent', and 'Relation' fields. The 'Add' button is highlighted in the 'Data Source' section of the left sidebar.

Fig 12.7: Including Data Source Fields for the Data Source

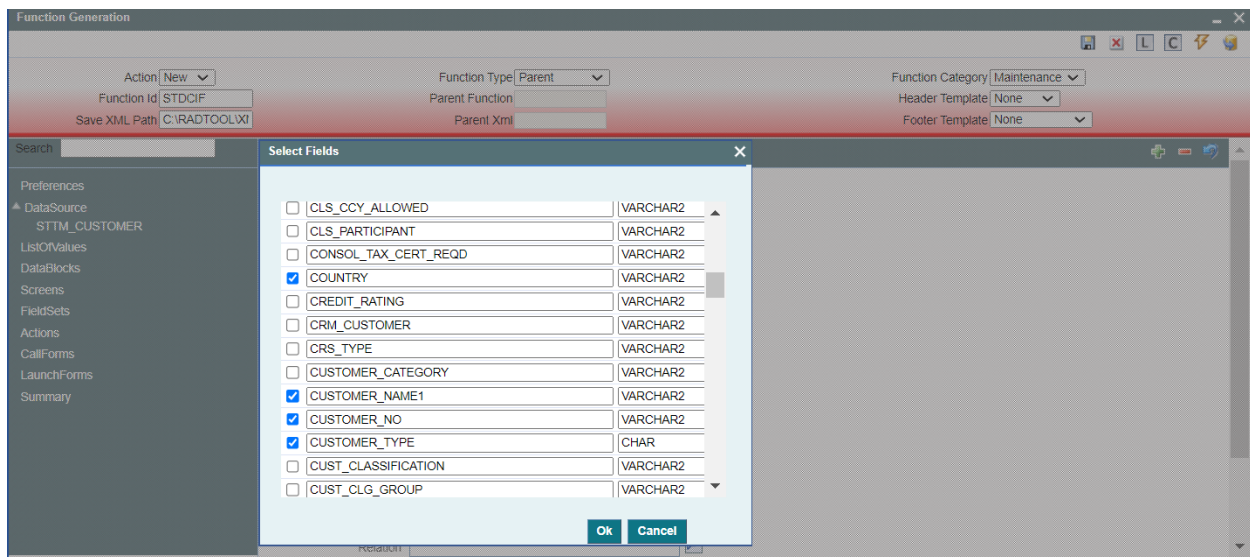


Fig 12.7: Selecting Data Source Fields for the Data Source

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

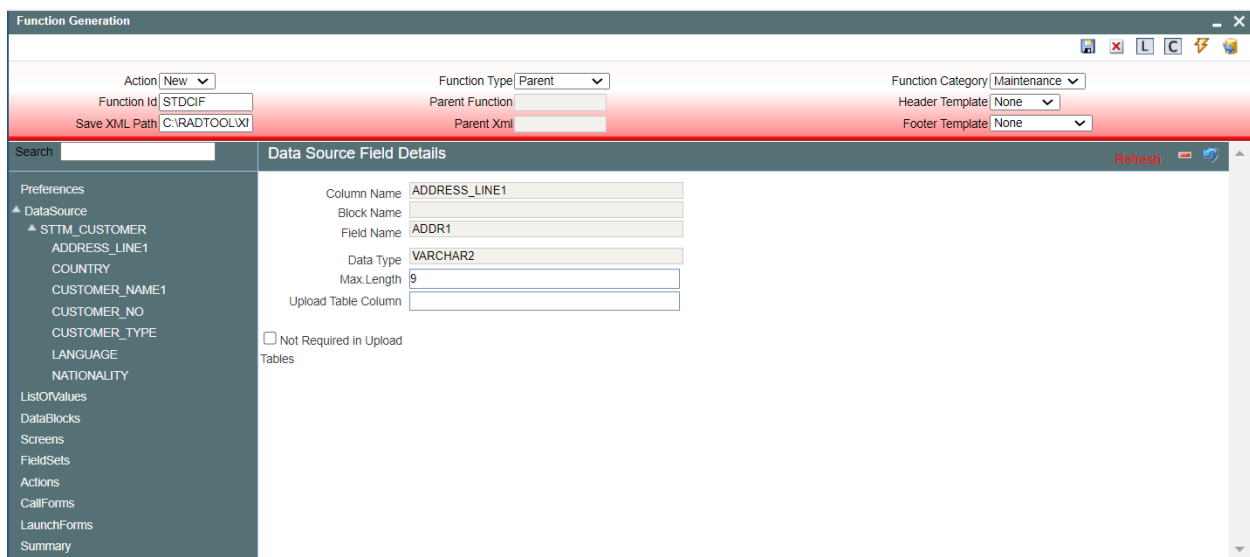


Fig 12.7: Providing properties for Data Source Fields

Data model of a single function id would include multiple tables .All the tables needs to added 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.

The screenshot shows the 'Function Generation' window with the 'Data Source Details' tab selected. The left sidebar lists various preferences, with 'Data Source' expanded and 'STTM_CUST_GROUP' selected. The main area displays the following details:

- Data Source:** STTM_CUST_GROUP
- Master:** No
- Relation Type:** One To Many
- Multi Record:** Yes
- PK Cols:** GROUP_ID~CUSTOMER_NO
- PK Types:** VARCHAR2~VARCHAR2
- Upload Table:** (empty field)
- Upload Where Clause:** (empty field)
- Fine Grained Service Details Only:**
 - Rest Data Block:** (empty field)
 - XSD Node:** (empty field)
 - Rest Relation:** (empty field)
 - Rest Relation Type:** (dropdown menu)
 - Rest PK Cols:** (empty field)
 - Rest PK Types:** (empty field)
 - Parent:** STTM_CUSTOMER
 - Relation:** (empty field)

Fig 12.7: 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 [Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development](#) 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>.

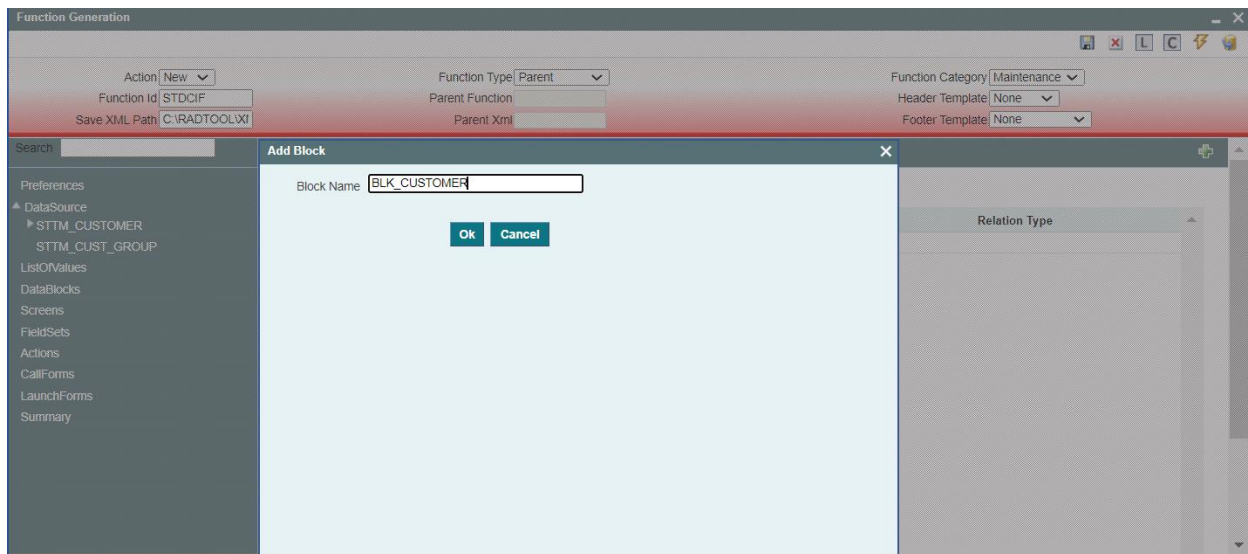


Fig 12.8: Creating a new Data Block

- 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.

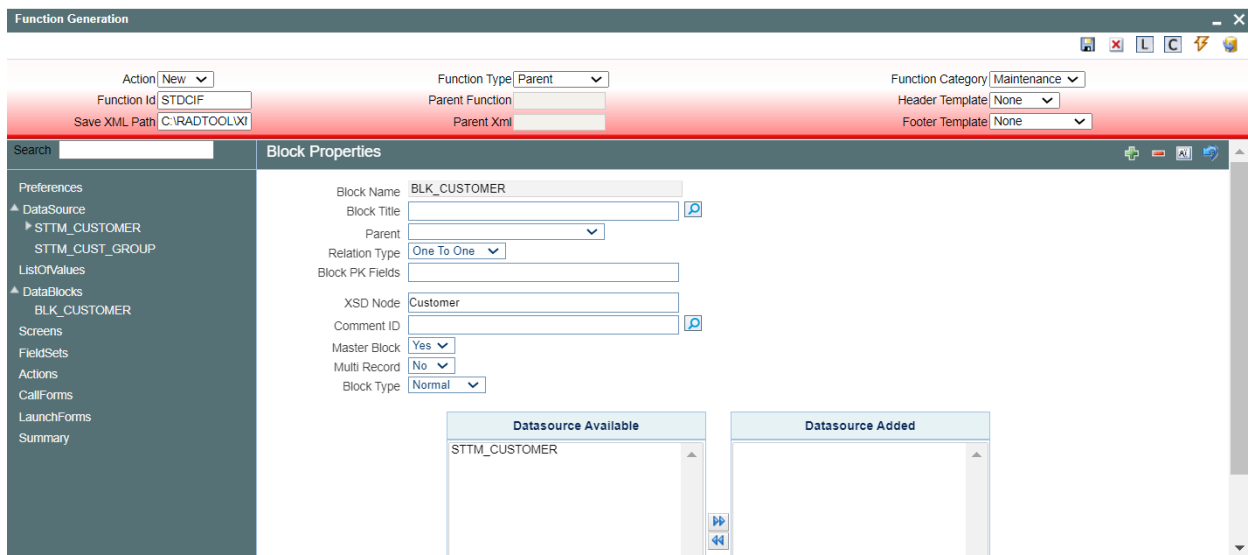


Fig 12.9: Providing properties for Data Block

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

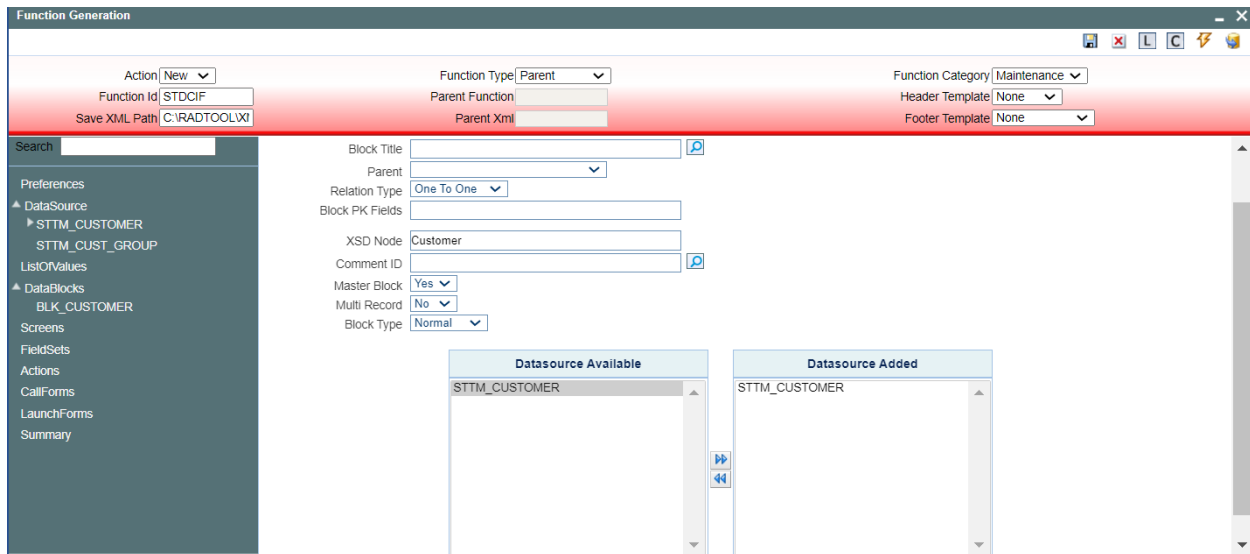


Fig 12.10: Attaching Data Sources to Data Block

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 where as 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.

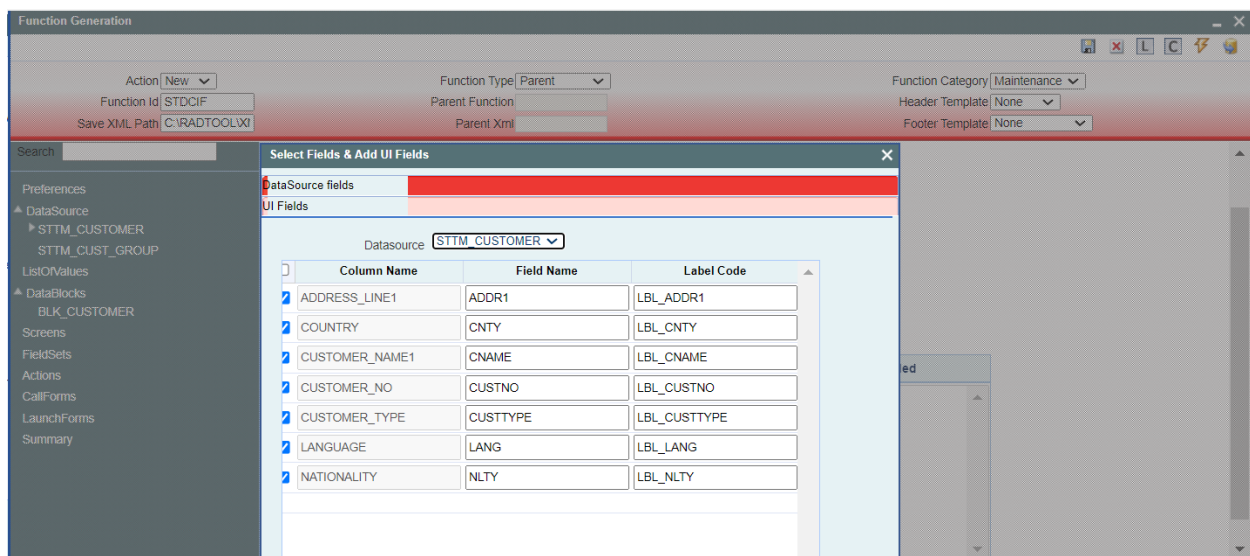


Fig 12.11: Adding Block Fields to Data Block

Refer [Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development](#) 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.

The screenshot displays the 'Function Generation' application window. The top section contains configuration fields: 'Action' set to 'New', 'Function Id' as 'STDCIF', 'Save XML Path' as 'C:\RADTOOL\XI', 'Function Type' as 'Parent', 'Parent Function' and 'Parent Xml' as empty fields, 'Function Category' as 'Maintenance', 'Header Template' as 'None', and 'Footer Template' as 'None'. The left sidebar shows a tree view with 'DataBlocks' expanded, listing fields like BLK_CUSTOMER, ADDR1, CNTY, CNAME, CUSTNO, CUSTTYPE, LANG, and NLTY. Below this, the 'Screens' node is expanded, showing 'CVS_MAIN' with sub-items 'HEADER', 'BODY', and 'FOOTER'. The main 'Screen Details' panel for 'CVS_MAIN' shows 'Screen Title' as an empty field, 'Screen Size' as 'Small', and 'Exit Button Type' as 'Default Cancel'. Checkboxes for 'Main Screen' (checked), 'OBIEE', 'Visible' (checked), 'Query Required', and 'Query all Img Required' are present. At the bottom, a table with columns 'Argument Name', 'Source Block', 'Source Field', 'Argument Value', 'Target Block', 'Target Field', and 'Active' is shown, currently empty.

Fig 12.12: Providing properties to new Screen

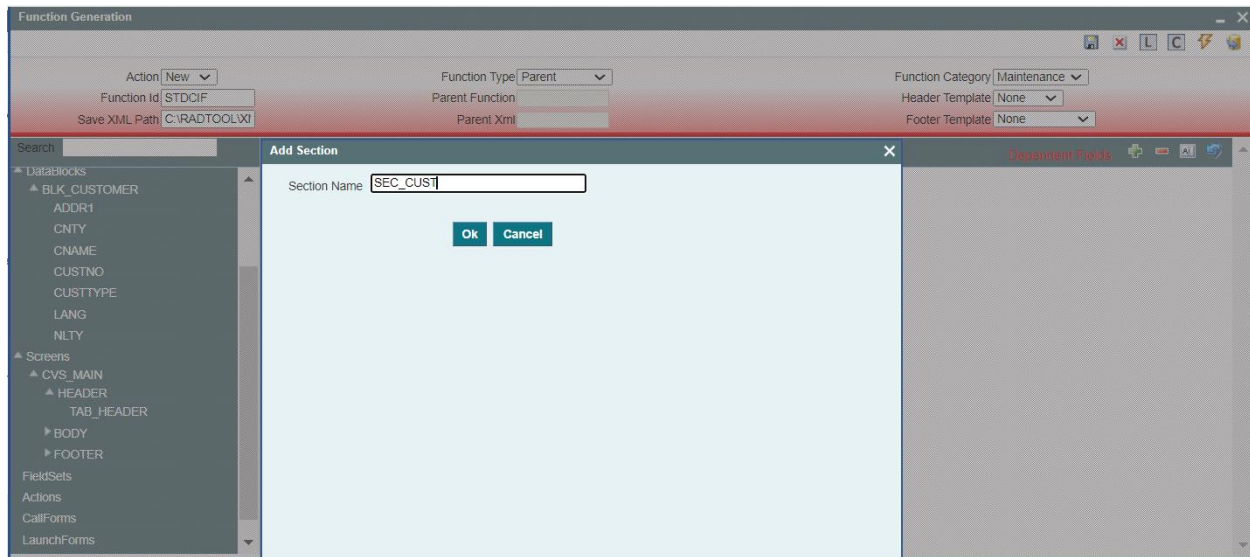


Fig 12.13: Creating new section in TAB_MAIN in the body of screen CVS_MAIN

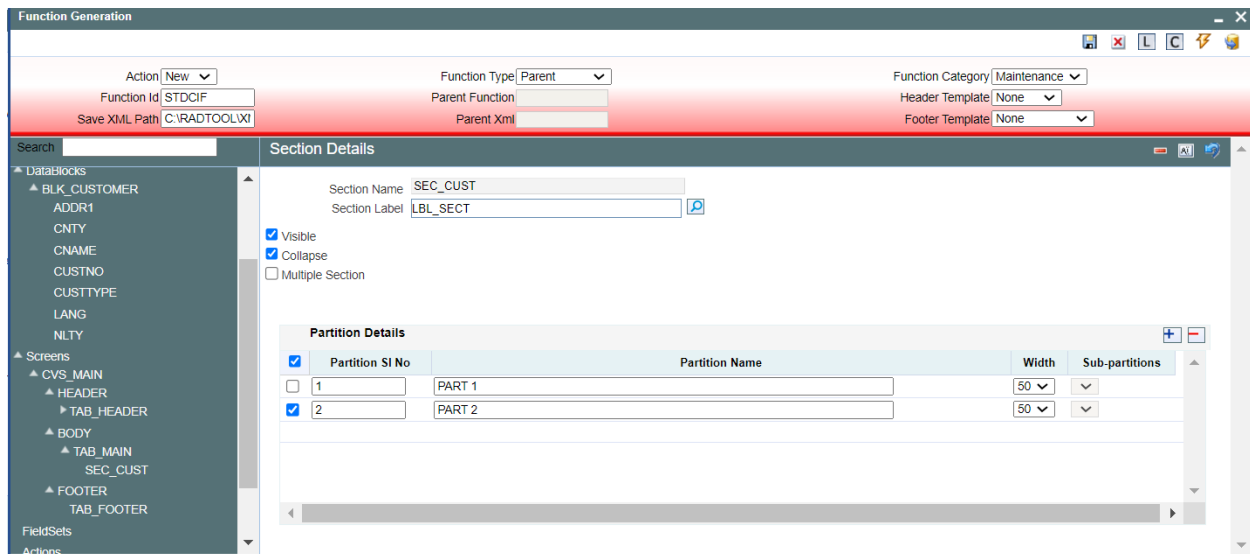


Fig 12.14: Defining partitions for the Section

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

Function Generation

Action: New
Function Id: STDCIF
Save XML Path: C:\RADTOOL\XI

Function Type: Parent
Parent Function:
Parent Xml:

Function Category: Maintenance
Header Template: None
Footer Template: None

Search:

Fieldset Properties

Fieldset Name: FST_CUST1
Fieldset Label: 
Data Block: BLK_CUSTOMER
Multi Record: No
View Type: Single
Fieldset Height:
Number Of Rows:
Fieldset Type: Normal
Screen Name: CVS_MAIN
Screen Portion:
Tab Name:
Section Name:
Partition Name:

☐ Horizontal Fieldset
☐ ReadOnly
☐ Navigation Button
☐ Navigation Button Full
Width
☒ Visible

Data Block Fields

ADDR1
CNTY
CNAME
CUSTNO
CUSTTYPE
LANG
NLTY

FieldSet Fields Subpartition Name

Fig 12.14: Attaching Fields to a Field set

Function Generation

Action: New
Function Id: STDCIF
Save XML Path: C:\RADTOOL\XI

Function Type: Parent
Parent Function:
Parent Xml:

Function Category: Maintenance
Header Template: None
Footer Template: None

Search:

Fieldset Properties

Fieldset Name: FST_CUST1
Fieldset Label: 
Data Block: BLK_CUSTOMER
Multi Record: No
View Type: Single
Fieldset Height:
Number Of Rows:
Fieldset Type: Normal
Screen Name: CVS_MAIN
Screen Portion:
Tab Name:
Section Name:
Partition Name:

☐ Horizontal Fieldset
☐ ReadOnly
☐ Navigation Button
☐ Navigation Button Full
Width
☒ Visible

Data Block Fields

CNTY
LANG
NLTY

FieldSet Fields Subpartition Name

FieldSet Fields	Subpartition Name
☐ CUSTTYPE	
☐ CUSTNO	
☐ CNAME	
☒ ADDR1	

Fig 12.14: 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.

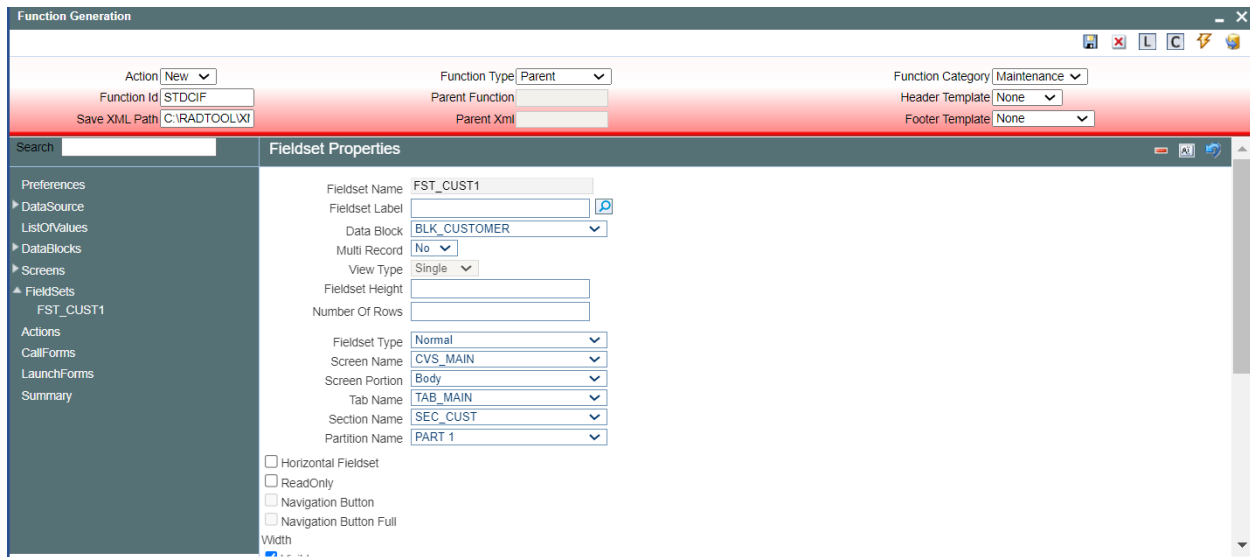


Fig 12.15: Providing details where Field Set has to be placed

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.

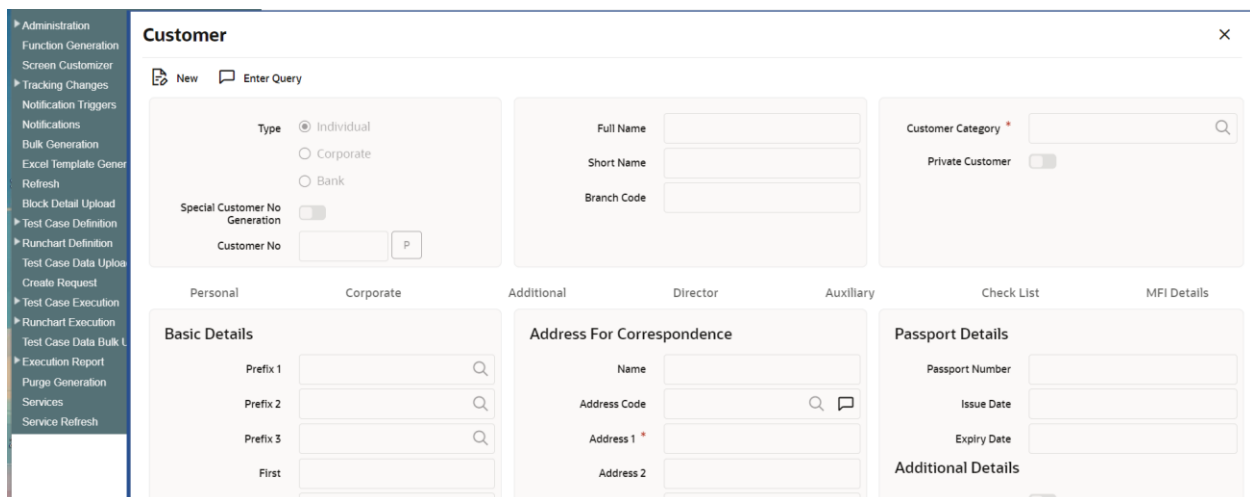


Fig 12.16: Preview of the designed Screen

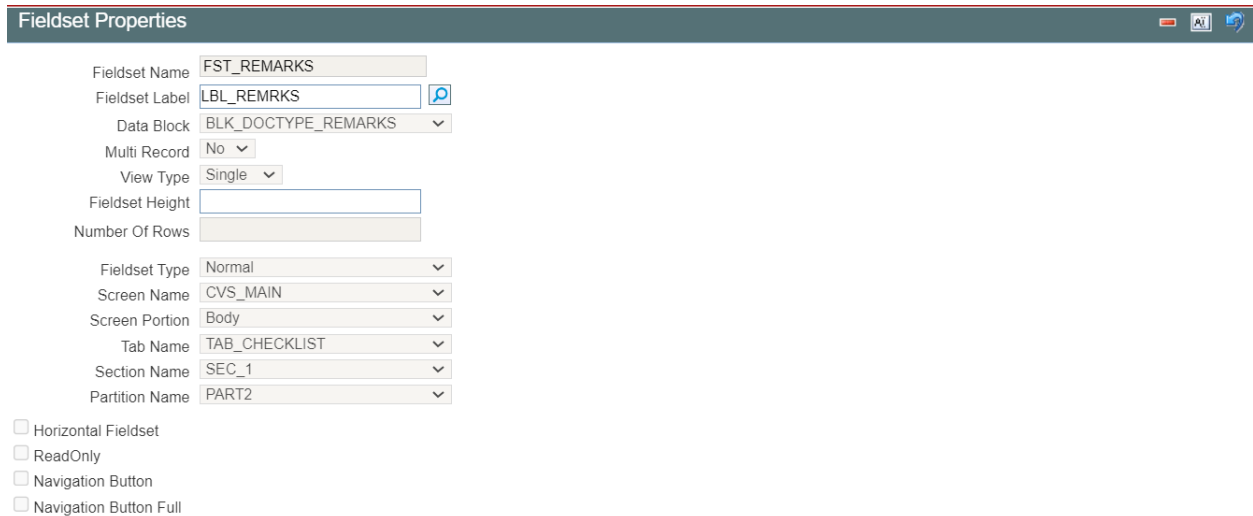
Adding Multi entry block to field set.

- 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 12.17: Multiple View Multi Record Field set

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



Fieldset Properties

Fieldset Name: FST_REMARKS

Fieldset Label: LBL_REMRKS

Data Block: BLK_DOCTYPE_REMARKS

Multi Record: No

View Type: Single

Fieldset Height:

Number Of Rows:

Fieldset Type: Normal

Screen Name: CVS_MAIN

Screen Portion: Body

Tab Name: TAB_CHECKLIST

Section Name: SEC_1

Partition Name: PART2

☐ Horizontal Fieldset

☐ ReadOnly

☐ Navigation Button

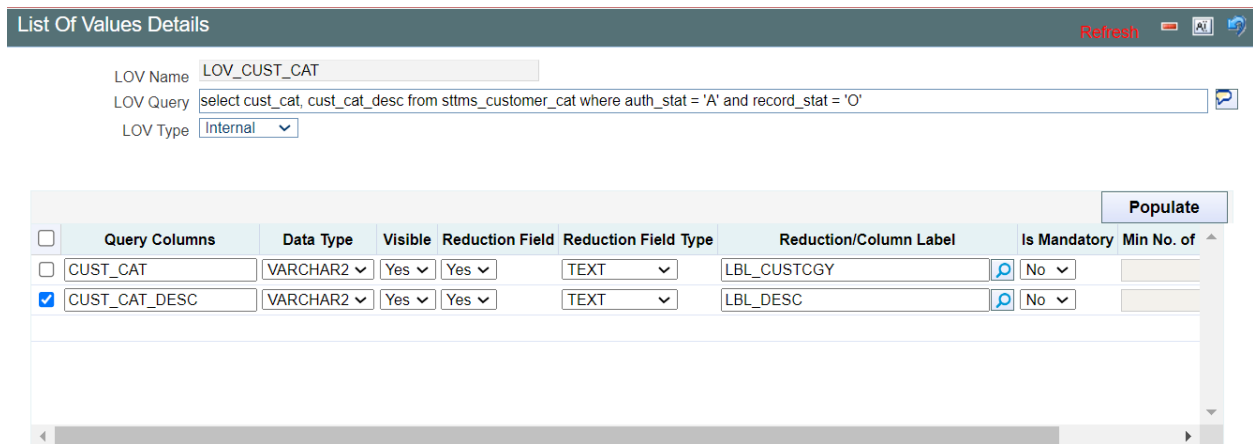
☐ Navigation Button Full

Fig 12.18: Properties 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_COUNTRY.
- Enter valid query and click on populate button



List Of Values Details

LOV Name: LOV_CUST_CAT

LOV Query: select cust_cat, cust_cat_desc from sttms_customer_cat where auth_stat = 'A' and record_stat = 'O'

LOV Type: Internal

Populate

	Query Columns	Data Type	Visible	Reduction Field	Reduction Field Type	Reduction/Column Label	Is Mandatory	Min No. of
☐	CUST_CAT	VARCHAR2	Yes	Yes	TEXT	LBL_CUSTCGY	No	
☒	CUST_CAT_DESC	VARCHAR2	Yes	Yes	TEXT	LBL_DESC	No	

Fig 12.19: Defining new LOV

LOV Query

```
select cust_cat, cust_cat_desc from sttms_customer_cat where
auth_stat = 'A' and record_stat = 'O'
```

Ok Cancel

Fig 12.20: Providing LOV query

List Of Values Details

Refresh

LOV Name: LOV_CUST_CAT

LOV Query: select cust_cat, cust_cat_desc from sttms_customer_cat where auth_stat = 'A' and record_stat = 'O'

LOV Type: Internal

Populate

☐	Query Columns	Data Type	Visible	Reduction Field	Reduction Field Type	Reduction/Column Label	Is Mandatory	Min No. of
☐	CUST_CAT	VARCHAR2	Yes	Yes	TEXT	LBL_CUSTCGY	No	
☒	CUST_CAT_DESC	VARCHAR2	Yes	Yes	TEXT	LBL_DESC	No	

Fig 12.21: Providing LOV details

- Redn/Col 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

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.*

4.8 Attaching Call forms

Maintenance Call forms can be attached to a maintenance screen. Refer the document [Development of Online Forms](#) 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

Datasource: **STTMS_CUSTOMER**

☐	Column Name	Field Name	Label Code
☐	JOINT_VENTURE		

Ok Cancel

Fig 12.24: Defining Button field

- Add Call form details to Call form node

Function Generation

Action: Load
Function Id: STDCIF
Save XML Path: STDCIF_RAD.X

Function Type: Parent
Parent Function:
Parent Xml:

Function Category: Maintenance
Header Template: None
Footer Template: Maint Audit

Search:

Call Form Details

☐	Function ID	Parent Data Block	Parent DataSource	Relation	Relation Type	Callfr
☐	MICCUSTM	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	STCCIFJ	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	SICDIARC	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	CSCCURLN	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	STCCIFTX	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	CSCFNUDF	BLK_CUSTOMER	STTMS_CUSTOMER		One To Many	
☐	STCCRDSA	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	CSCOFACM	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO=CS	One To One	
☐	STCCUACC	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO = S	One To One	
☐	STCNSCK	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO=ST	One To One	
☒	CSCUPDOC	BLK_CUSTOMER	STTMS_CUSTOMER	STTMS_CUSTOMER.CUSTOMER_NO=CS	One To One	

Screen Arguments Dependent Fields

Fig 12.25: 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
- Check the preview.

4.9 Adding Summary

- 1) Add entry in Preferences node for Summary screen
- 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 any where 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 .

Summary Preview

Right click on summary node and click on 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.

5. Generation and Deployment of files

Generate Files

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

Deploy files

- Click on deploy button select the required files to deployed to server and click on deploy. On successful deployment status will be displayed as Deployed.

Testing

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

6. Generated Units

The following units will be generated for a Maintenance screen.

Refer document [Development Workbench - Screen Development II](#) 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

ix) Summary 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 STDCIF_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

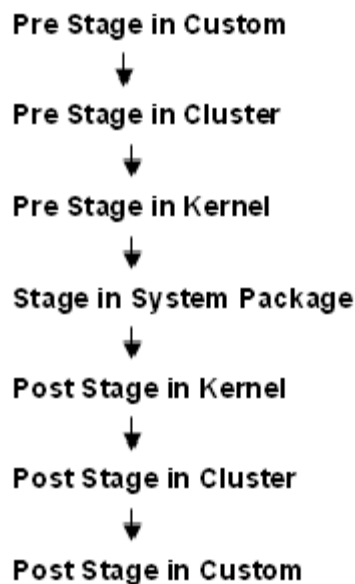


Fig 12.31: Flow of control through Hook Packages

Example: For Fn_check_mandatory, flow will be as

STPKS_ STDCIF_MAIN.Fn_Check_Mandatory



STPKS_ STDCIF_CUSTOM.Fn_Pre_Check_Mandatory



STPKS_ STDCIF_CLUSTER.Fn_Pre_Check_Mandatory



STPKS_ STDCIF_KERNEL.Fn_Pre_Check_Mandatory



STPKS_STDCIF_MAIN.Fn_Sys_Check_Mandatory

STPKS_STDCIF_KERNEL.Fn_Check_Mandatory

STPKS_STDCIF_CLUSTER.Fn_Check_Mandatory

STPKS_STDCIF_CUSTOM.Fn_Check_Mandatory

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 *STPKS_STDCINF_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 STDCIF:

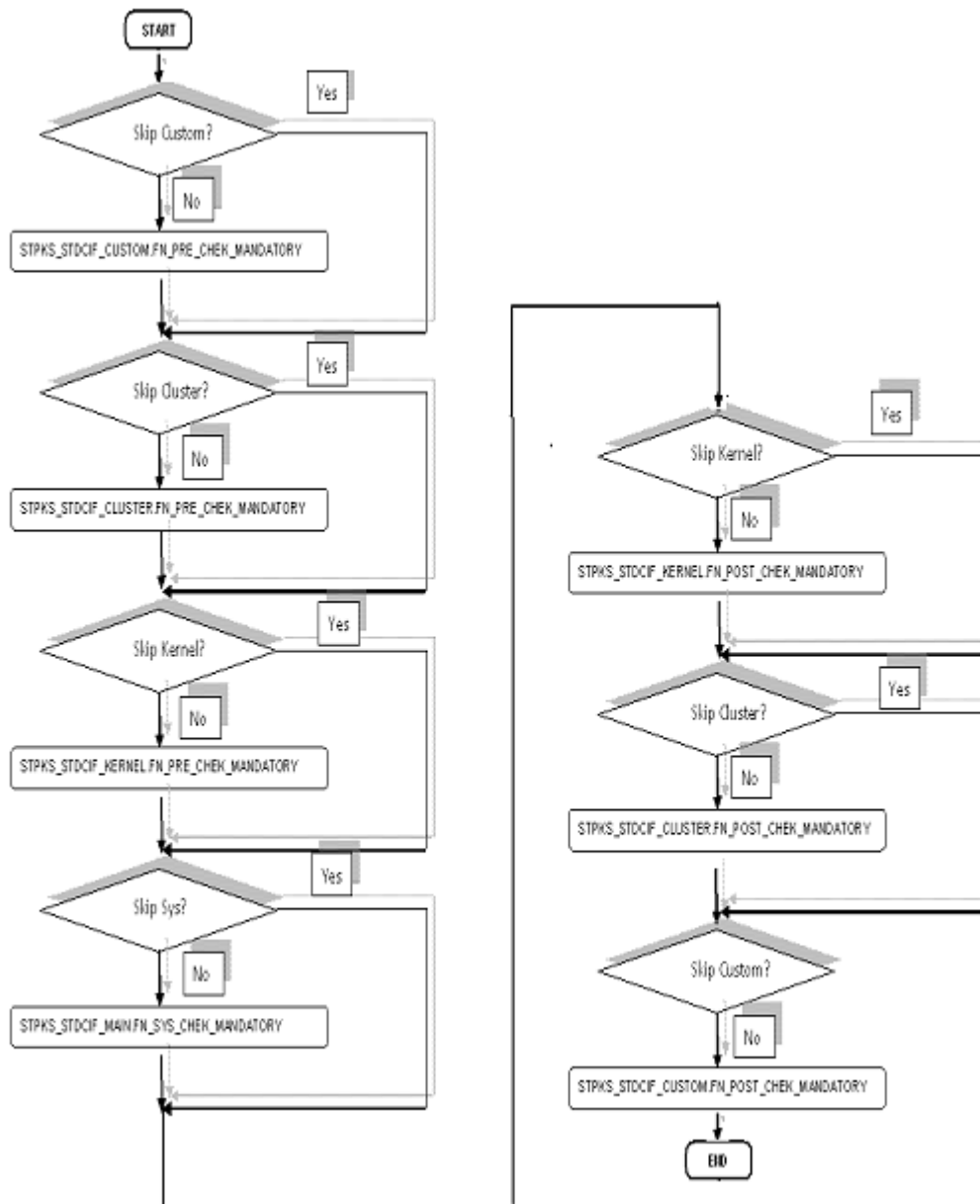


Fig 12.31: Flow of control explaining skip logic in packages



Development of Maintenance Form
[November] [2023]
Version 14.7.2.0.0

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India

Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax: +91 22 6718 3001
www.oracle.com/financialservices/

Copyright © 2007, 2023, Oracle and/or its affiliates. All rights reserved.

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.