# Oracle® FLEXCUBE Investor Servicing Development of Maintenance Form





Oracle FLEXCUBE Investor Servicing Development of Maintenance Form, Release 14.7.6.0.0

G27722-01

Copyright © 2007, 2025, Oracle and/or its affiliates.

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.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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 fail-safe, 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.

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

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

# Contents

Purpose	\
Audience	\
Documentation Accessibility	V
Critical Patches	V
Diversity and Inclusion	V
Conventions	V
Screenshot Disclaimer	Vİ
Prerequisite	Vİ
Related Resources	vi
Overview of Maintenance Screen	
Screen Development	
2.1 Header Information	2-3
2.2 Preferences	2-3
2.3 Data Sources	2-4
2.4 Data Blocks	2-8
2.5 Screens	2-10
	2-12
2.6 Field Sets	0.14
2.7 List Of Values	
<ul><li>2.7 List Of Values</li><li>2.8 Add Call Forms</li></ul>	2-16 2-19
2.7 List Of Values	
<ul><li>2.7 List Of Values</li><li>2.8 Add Call Forms</li></ul>	
<ul><li>2.7 List Of Values</li><li>2.8 Add Call Forms</li><li>2.9 Add Summary</li></ul>	2-2 2-2
<ul> <li>2.7 List Of Values</li> <li>2.8 Add Call Forms</li> <li>2.9 Add Summary</li> <li>2.10 Maintain Amendable fields</li> <li>Generate and Deploy Files</li> </ul>	2-1 2-2



4.3 Other Units 4-3

# 5 Extensible Development

5.1	1 Extensibility in JavaScript Coding		5-2
5.2	Exte	nsibility in Backend Coding	5-2
	5.2.1	Functions in Hook Packages	5-2
	5.2.2	Flow of control through Hook Packages	5-2
	523	Bynass Base Release Functionality	5-4



## **Preface**

**Oracle FLEXCUBE Investor Servicing** is a comprehensive mutual funds automation software from Oracle® Financial Servicing Software Ltd.©.

You can use the system to achieve optimum automation of all your mutual fund investor servicing processes, as it provides guidelines for specific tasks, descriptions of various features and processes, and general information.

This topic contains the following sub-topics:

- Purpose
- Audience
- Documentation Accessibility
- Critical Patches
- Diversity and Inclusion
- Conventions
- Screenshot Disclaimer
- Prerequisite
- Related Resources

## Purpose

This manual is designed to help FLEXCUBE Application developers/users to familiarize with ORACLE FLEXCUBE Development Workbench for Investor Servicing.

## **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:

Table 1 Proficiency and Resources

Proficiency	Resources
FLEXCUBE Functional Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Object Naming Conventions	Development Overview Guide



Table 1 (Cont.) Proficiency and Resources

Proficiency	Resources
Working knowledge of Web based Applications	Self-Acquired
Working knowledge of Oracle Database	Oracle Documentations
Working knowledge of PLSQL developer	Respective vendor documents
Working knowledge of PLSQL and SQL Language	Self-Acquired
Working knowledge of XML files	Self-Acquired

# **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 customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

## **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.

# **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.

## 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.

## 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.

# Prerequisite

Specify User ID and Password, and log in to Home Screen.

## Related Resources

The functions of ORACLE FLEXCUBE Development Workbench for Investor Servicing system is organized into various guides, each discussing a component.

For more information, see these Open Development Tool documents:

- Open Development Tool Installation
- Development Workbench Getting Started
- Development Workbench Administration
- Development Workbench Screen Development I
- Development Workbench Screen Development II
- Development Workbench Screen Customizer
- Development Workbench Notifications
- Development Workbench Bulk Generation
- Development Workbench Source Upgrade
- Development Workbench Tracking Changes
- Child and Screen Childs Concept and Design
- Development of Online Form
- Development of Call Form
- Development of Launch Forms and Other Screens
- Development of Dashboard Form
- Development Workbench Service XML Development
- Development Workbench Performance Tuning Enhancements
- Development Workbench Rest Services Development



1

# Overview of Maintenance Screen

This topic provides an overview of the Maintenance Screen.

Maintenance function IDs are used for storing maintenance data which are required for processing any contracts, batches, or for any other maintenance which are dependent on this.

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

## **Example 1-1 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 an account for the customer) as well as for transaction processing (debiting of customer account).



# Screen Development

This topic provides an overview of maintenance screen development.

The design and development of a Maintenance function id are similar to any other function Ids.

This topic briefs the steps in designing the Maintenance screen. **UTDFNDRL** is the sample function id used for demonstration in this document. For a detailed explanation, refer to the topic **Development WorkBench - Screen Development I**.

This topic contains the following sub-topics:

#### Header Information

This topic describes about defining the header information for Maintenance Forms.

#### Preferences

This topic describes about defining the preferences for Maintenance Forms.

#### Data Sources

This topic provides the systematic instructions to create Data Sources for Maintenance screens.

#### Data Blocks

This topic provides systematic instructions to create Data Block.

#### Screens

This topic provides systematic instructions to create a new screen.

#### Field Sets

This topic provides systematic instructions to create a new Fieldset.

#### List Of Values

This topic provides systematic instructions to define LOVs.

## Add Call Forms

This topic provides systematic instructions to attach Call Forms.

#### Add Summary

This topic provides systematic instructions to Add Summary.

## · Maintain Amendable fields

This topic provides systematic instructions to maintain amendable fields.

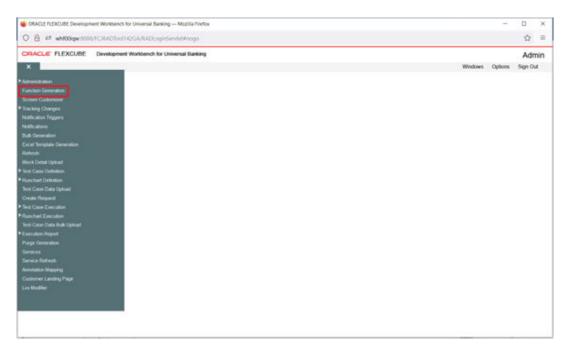
# 2.1 Header Information

This topic describes about defining the header information for Maintenance Forms.

 On Expand Menu of the Development Workbench for Universal Banking, click Function Generation node.

The Function Generation screen displays.

Figure 2-1 Function Generation



On the Function Generation screen, right-click the Header Information and select Add option to create the new data block.

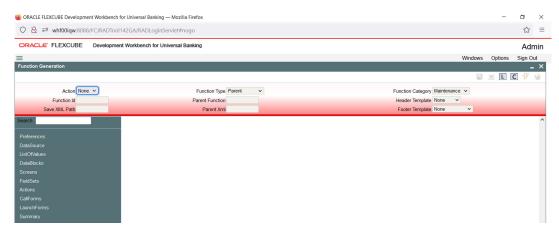
For more information on fields, refer to the field description table.

**Table 2-1** Function Generation - Field Description

Field	Description
Function ID	It is the name of the Maintenance Form.
	The third letter of the function id has to be <b>D</b> . Ideally the function id name should have 8 characters.
	Example: UTDFNDRL
Function Category	It is the Maintenance Form Category.
	It has to be <b>Maintenance</b> .
Function Type	Parent
Parent Function Id	None
Parent XmI	None
Header Template	None (Only for Process flow screens)
Footer Template	Select the footer template from the drop-down list.  None  Maint Audit
	Maint Audit     Maint Process
	• Process
	For Maintenance Screen, footer template should be selected as <b>Maint Audit</b> .
	Make sure that the master data source has the audit columns if footer template is provided as Maint log



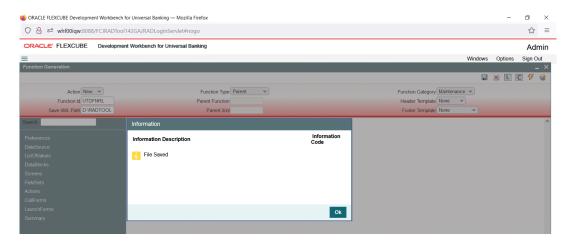
Figure 2-2 Header Information for Maintenance Screen



3. You can save your work at any time by clicking the **Save** icon. Select the **Action** as **Load** to load the radxml file from the required path.

The Information Window is displayed.

Figure 2-3 Information Window



## 2.2 Preferences

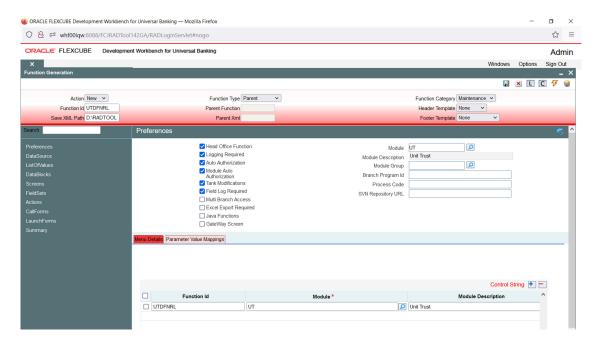
This topic describes about defining the preferences for Maintenance Forms.

Details entered in **Preferences** screen are used in generating INCS for **SMTB\_MENU**, **SMTB\_FUNCTION\_DESCRIPTION** and **SMTB\_ROLE\_DETAILS**.

In the **Control String** field, the developer needs to select the actions which should be available for **Preferences** screen in the FLEXCUBE.



Figure 2-4 Preferences for Maintenance Screen



Note the following points while providing details in the Preferences screen

- Control String: REVERSE, ROLLOVER, CONFIRM, LIQUIDATE, HOLD operations are not applicable for maintenance screens.
- 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

Refer *Development WorkBench - Screen Development I* for the detailed explanation on preferences.

#### Example 2-1 UTDFNDRL

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

## 2.3 Data Sources

This topic provides the systematic instructions to create Data Sources for Maintenance screens.



 On the Function Generation screen, right-click the Data Source node and select the Add option to create a new Data Source.

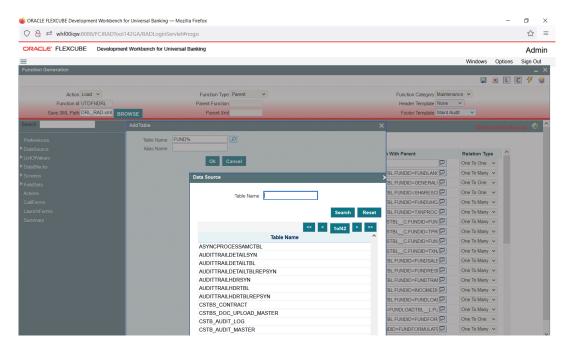
The **AddTable** window displays.

On the AddTable window, select the Table Name from the list of values to get the list of tables available and select the required table from the list.

If the user knows the exact **Table Name**, the user can specify the name directly.

The **Data Source** window is displayed.

Figure 2-5 Add Table\_Data Source



3. Select Master as Yes if the added data source is Master Data Source for the screen.

Every function id should have one master data source.

4. Verify **Primary Key columns** and **Primary Types** fields are populated and if its not entered then specify the **Pk Cols** and **Pk Types** fields.

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 populate automatically otherwise the user needs to enter values without fail. If the user misses **Pk Cols** and **Pk Types** package generation will fail.



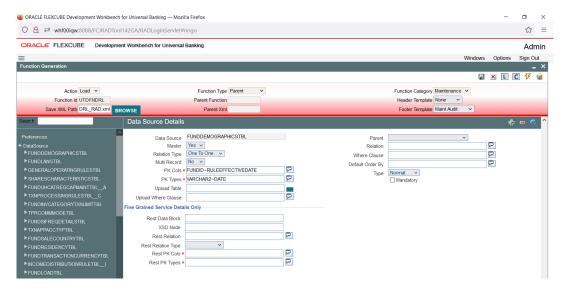
Master Data Source cannot have any parent.

5. Click the **Save** icon at the top right of the screen.

The Data Source Details screen displays.



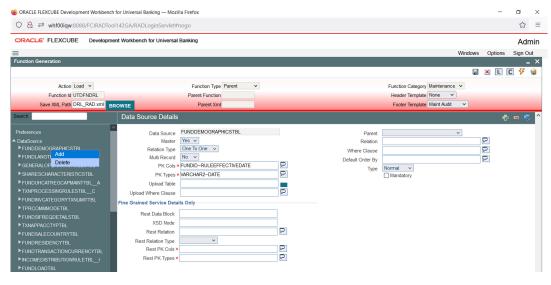
Figure 2-6 Providing master Data Source Properties



Under the DataSource node, right-click the added table (STTM\_CUSTOMER) to add fields to the table.

The **Data Source Details** screen displays with added table.

Figure 2-7 Including Data Source Fields for the Data Source

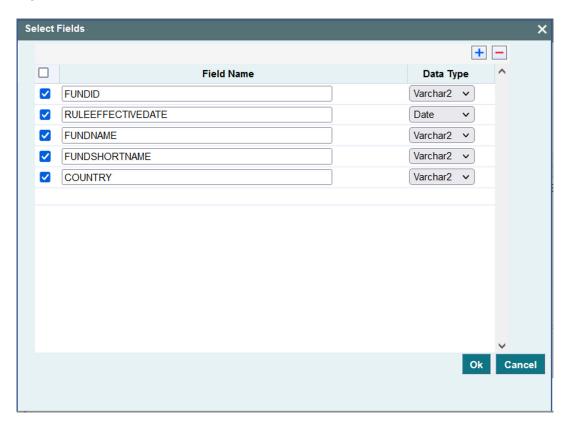


7. Select the **Add** option to add fields to the table.

The **Select Fields** window displays.



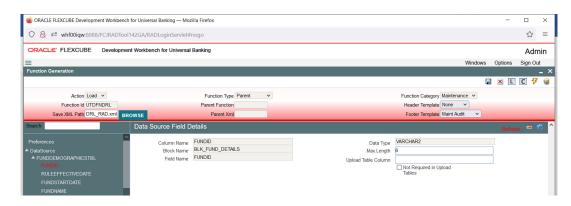
Figure 2-8 Select Fields



8. Select the required fields and click **Ok** button.

The selected fields will get added to the Data Source Tree. Provide **Data Source Field Properties**: Only maximum length can be modified by the developer in **Data Source Field Properties**. Rest will have defaulted from the table definition.

Figure 2-9 Data Source Field Details



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

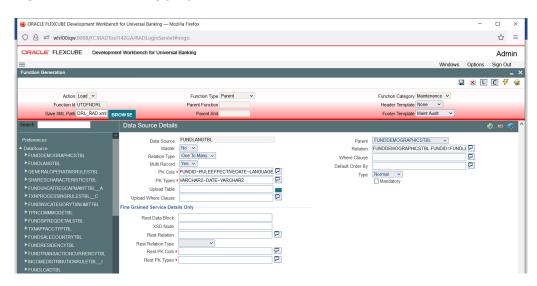
- 9. To add Child Data Source, follow the below steps:
  - a. In Data Source Details screen, select Multi Record value as Yes if the child data source is Multi record table.



- Child Data sources should always be associated with a parent.
- **c.** Relation is mandatory between parent and child. While giving relation, the parent data source should come on the left side of the relation.



Figure 2-10 Providing properties for Child Data Source



- 10. 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 the 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 the system will handle it.

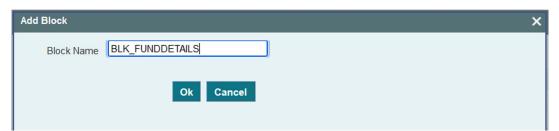
## 2.4 Data Blocks

This topic provides systematic instructions to create Data Block.

 Under Function Generation screen, right-click the Data Block and select Add option to create the new data block.

The Add Block window displays.

Figure 2-11 Add Block



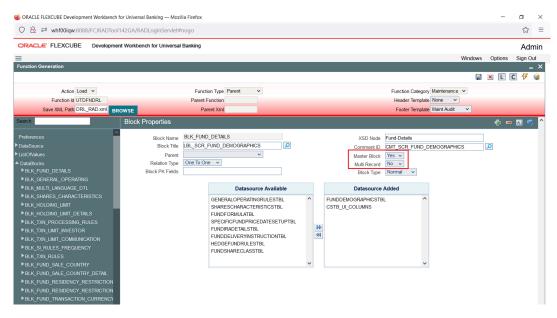


Specify the Block Name field and click Ok.

Block Name should start with **BLK**\_. The **Block Name** should be short and equivalent to a data source but not the same as **DataSource Name**.

The new DataBlock gets created and **Block Properties** screen displays.

Figure 2-12 Providing properties for Data Block



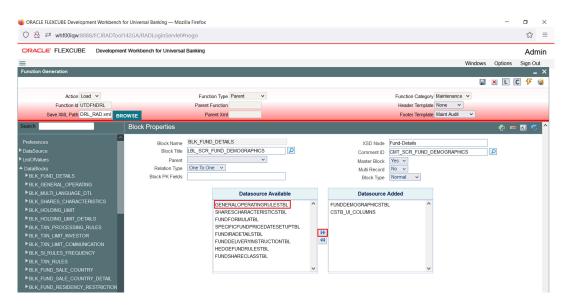
- Select Parent block if added block is not Master Block.
- Select Multi Record field as Yes/No based on Master Block value.

Available data sources displays in the **DataSource Available** text area.

Select the required data source and click the Move button to attach Data Source to the Block.

The selected Data sources gets attached to the Block.

Figure 2-13 Attaching Data Sources to Data Block





Select Multi Record as Yes in the data Block Properties screen to add multi record data source to data block.

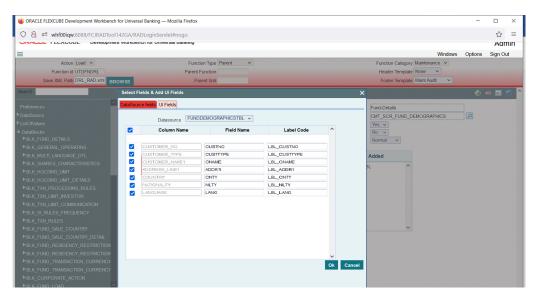
Multi Data Source once used to one block won't be available for reuse whereas a single record data source can be used in multiple blocks.

All the data sources with **Multi Record** as **Yes** will be populated.

- Select Block Fields as follows:
  - a. Right-click the newly added block.

The Select Fields & Add UI Fields window displays.

Figure 2-14 Select Fields & Add UI Fields



- b. Check the right side check boxes to add the required fields.
- c. Verify the Field Name and Label Code details and click the OK button.

The **Field Name** should not be the same as the column name. Special characters are also not allowed in the **Field Name** (including underscore and space). **Label Code** will be automatically populated based on the **Field Name**.

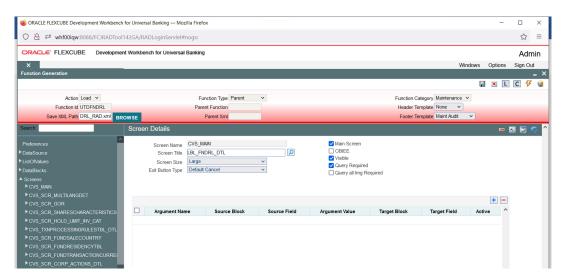
## 2.5 Screens

This topic provides systematic instructions to create a new screen.

 On the Function Generation screen, right-click on the Screens node and select Add option.

The Add Screen window displays.

Figure 2-15 Providing properties to new Screen



2. Specify the Screen Name field.

The Screen Name should start with CVS\_<Name>.

The **Screen Details** screen displays.

Specify the Screen Details and click the Save icon to create a new screen.

A new screen gets created and displays under the **Screen** node.

#### Figure 2-16 Screen Details- New Screen

By default screens are divided into 3 parts:

- Header
- Body
- Footer

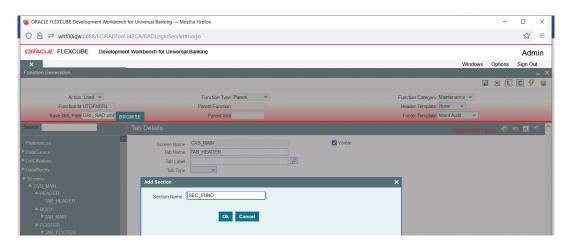
One Main Screen is Mandatory and **Tabs** can be defined on any of the screen portions as required. User can add **Sections** to **Tabs**. Each section can be divided into **Partitions**.

4. To add a new **Section** to **Tab**, right-click the **Tab** and then select **Add** option.

The Add Section screen displays.



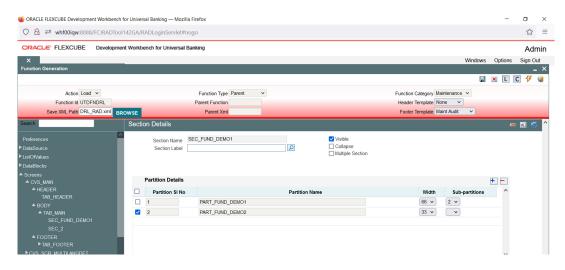
Figure 2-17 Creating new section



- 5. On the Add Section screen, specify the Section Name field and click the OK button.
  - A new **Section** gets created and displays under the respective **Tab**.
- 6. User can divide each section into required partitions.

The **Partition Details** window displays and the user can specify details.

Figure 2-18 Defining partitions for the Section



## 2.6 Field Sets

This topic provides systematic instructions to create a new Fieldset.

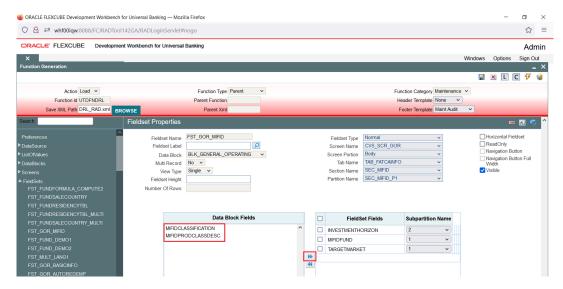
A group of fields can be grouped in a Fieldset which can be placed together on the screen. FieldSet Name should start with **FST\_<>**.

1. On the Fieldset Properties screen, select the block adding to field set.

All fields available to the block displays in the **Data Block** fields text area.



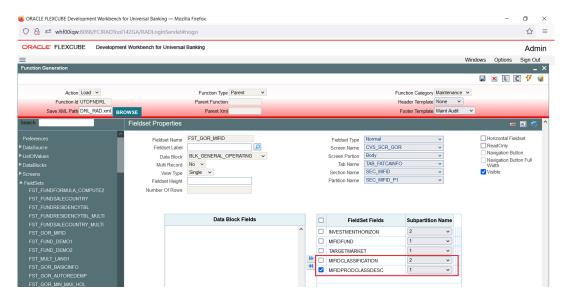
Figure 2-19 Attaching Fields to a Field set



Move fields from Data Block Fields to FieldSet Fields.

All fields available to the block displays in the **FieldSet Fields** text area. The order of fields in field set fields will reflect in the screen as well.

Figure 2-20 Fieldset Properties- FieldSet Fields

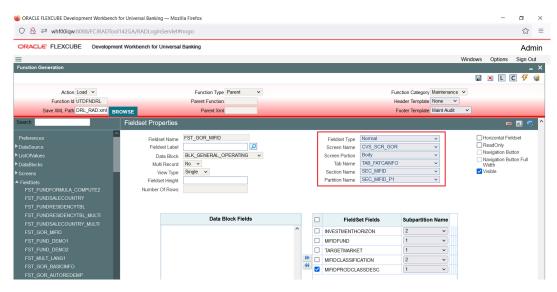


- 3. Select the screen portion (Header/Body/Footer) where this fieldset has to be placed.
- 4. Select remaining details like **Tab**, **Section**, and **Partition**.

Once fields are added to the fieldset, the developer can check the preview of the designed screen.



Figure 2-21 Providing details where Field Set has to be placed



5. Right-click the screen name and click on the **Preview** option.

The preview of the designed screen displays.

Figure 2-22 Preview of the designed Screen



- **6.** For adding Multi entry block to the fieldset, follow the below-given steps:
  - a. On the Fieldset Properties screen, select Multi Record field as Yes.
     In the case of Multi records, the View Type can be either Single or Multiple (By Default).

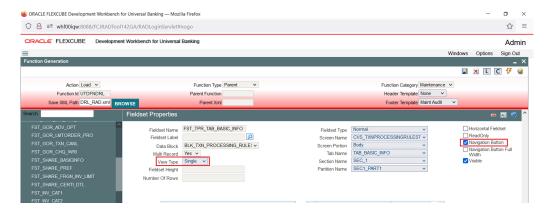


Figure 2-23 Preview - Multiple view multi-record Fieldset



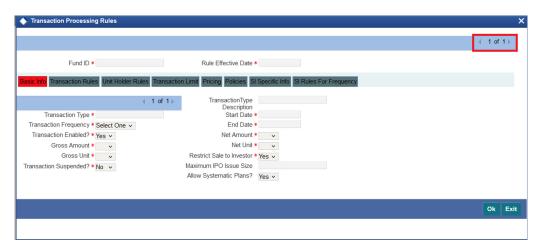
**b.** Select the **Navigation Button** field for multi-record single view.

Figure 2-24 Fieldset Properties - Single view multi-record Fieldset



Below figure shows the preview of a single view multi-record fieldset:

Figure 2-25 Preview - Single view multi-record Fieldset



## 2.7 List Of Values

This topic provides systematic instructions to define LOVs.

 To add LOV, right-click the List Of Values node and select Add option from the right-click menu.

List Of values can be defined for the function id using  ${f LOV}$  node

LOV window displays.

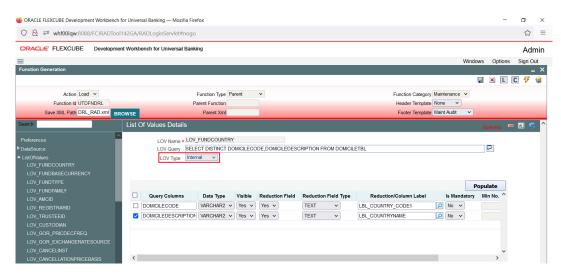
2. Specify the **LOV Name** field and then click **Ok** button.

LOV name should start with LOV\_<name>.

For Example: LOV\_COUNTRY

The List Of Values Details screen displays.

Figure 2-26 List Of Values Details

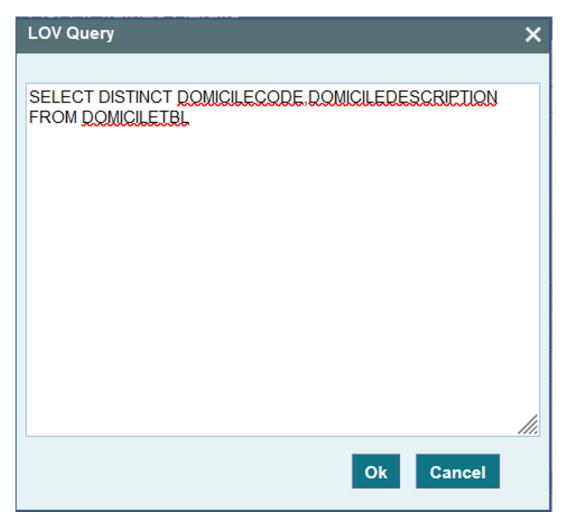


- 3. Specify the fields in the List Of Values Details screen.
- 4. Enter a valid query and click the **Populate** button.

The **LOV Query** window displays.



Figure 2-27 LOV Query



5. Click the **OK** button.



**Reduction/Column Label** are mandatory. If the user won't provide **Reduction/Column Label** field, an error displays on click of the **LOV** button after deployment in FLEXCUBE.

- **6.** After defining LOV, go to block and the corresponding field where the LOV has to be attached.
- 7. For Block Field Properties to attach LOV to the field, follow the steps as follows:
  - a. Select Display Type as LOV.
  - b. Select the required **LOV Name** from the list of all defined LOV's.
  - c. Click the Return Fields tab.

Return Fields Mapping tab opens.

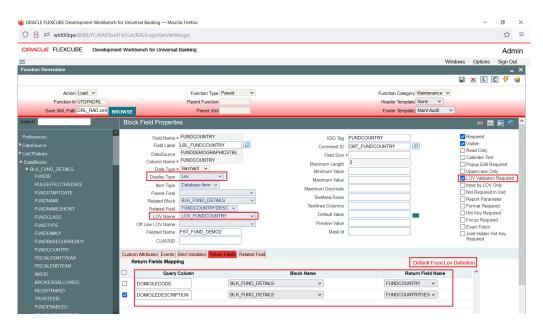
d. Click on **Default from Lov Definition** button.

The result fields maintained in the LOV query gets populated.

 Select the desired field (and its block) to which the result of the LOV query should have defaulted.

If the return field is not required to have defaulted to any field in the screen, the return field value can be left blank.

Figure 2-28 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.

For 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 = '0'
and once_auth = 'Y'
and ac stat de post = 'Y'
```

 $\textbf{f.} \quad \text{In the block field, after selecting } \textbf{Return Fields}, \ \text{click the } \textbf{Bind Variables} \ \text{tab}.$ 

Bind Variables Mapping tab opens.

g. Click the **Default From Lov Definition** button.

New rows will be created depending on the number of bind variables provided in the LOV query.

- Select the bind filed in the screen (and its block) for the LOV.
- i. Select the **Data Type** of the field.

The Block Field Properties\_Bind Variables Mapping screen displays.



5.7 ○ 🎖 🗝 whf00iqw:8088/FCJRADTool142GA/RADLoginServlet#r ORACLE: FLEXCUBE Development Workbench for Universal Banking Admin L C 🗸 Function Id UTDFNDRI Field Name \* BTN UMBFNDLOV Comment ID Read Only DataSource Field Size □ calender Text
□ Popup Edit Required
□ Uppercase Only
□ LOV Validation Require
Valiquation Require
Not Required In Xsd
□ Report Parameter
□ Formal Required
□ Hot Key Required
□ Exact Fetch
□ Joint Molder Hot Key Column Name \* Maximum Length Minimum Value Data Type Varchar2 V
Display Type Lov
Item Type Control V Maximum Value Parent Field TextArea Rows LOV\_UMBFUND Default Value Off Line LOV Name Fieldset Name FST FUND DEMO Mask Id BLK\_FUND\_DETAILS UMBRELLAFUND V

Figure 2-29 Defining bind variable for the LOV

## 2.8 Add Call Forms

This topic provides systematic instructions to attach Call Forms.

Maintenance Call forms can be attached to a maintenance screen.

Refer to the topic *Development of Call Form* for developing call forms.

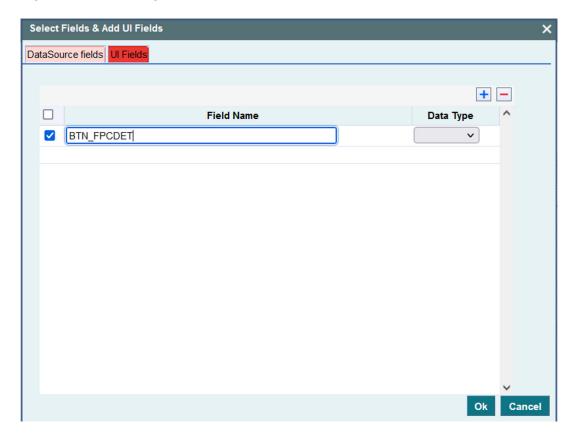
For attaching Call forms follow the below given instructions:

- 1. Add button to block to launch Call form on button click.
- 2. Right-click the **Block** and select **Add Fields** option.

The Select Fields and Add UI fields window displays.



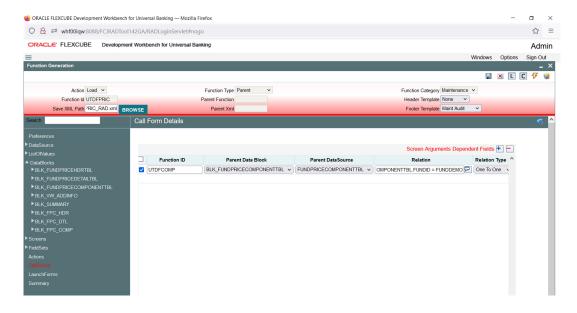
Figure 2-30 Defining Button field



- 3. Select **UI Fields** tab and click on **Add (+)** to add a row.
- 4. Enter button name and click the **Ok** button.
- 5. Select **Data Type** as a button and enter **Field Label**.
- 6. Add Call form details to the Call Form node.

The Call Form Details screen displays.

Figure 2-31 Call Form Details





**Add event to the button**: On If the user needs to place the button position in the desired place on the screen, the **Event Type** should be **Normal**. The user has to write code in release specific JavaScript file to launch the screen.

#### Figure 2-32 Block Field Properties - Events

7. Define the Call form details to be attached in call form node.

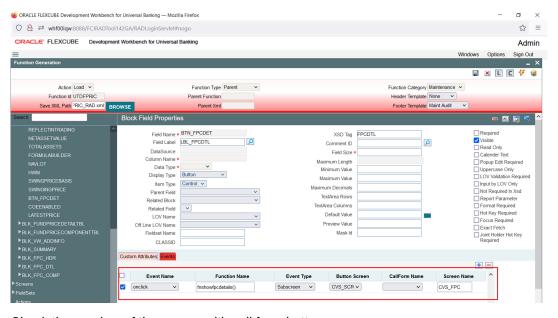
Figure 2-33 Call Form Details\_Call Form Node



8. Add event to the button. Select event type as **Call Form** or **Launch Form** or **Sub Screen**, button will be displayed on the 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.

Figure 2-34 Defining event to the button



9. Check the preview of the screen with call form buttons.

Figure 2-35 Preview of the screen with the Call Form button

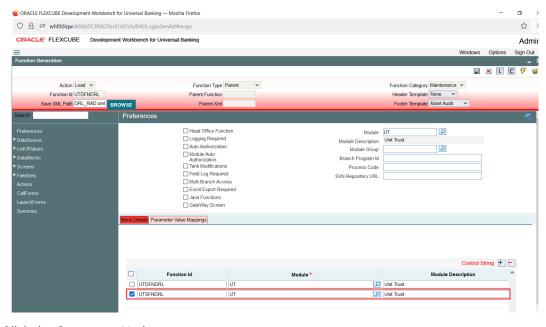


# 2.9 Add Summary

This topic provides systematic instructions to Add Summary.

- 1. On the **Function Generation** screen, click the **Preferences** node.
  - The **Preferences** screen displays.
- 2. In **Preferences** screen, add entry for the summary screen.
  - Preferences screen displays with Menu Details.

Figure 2-36 Adding Summary screen details in Preferences node

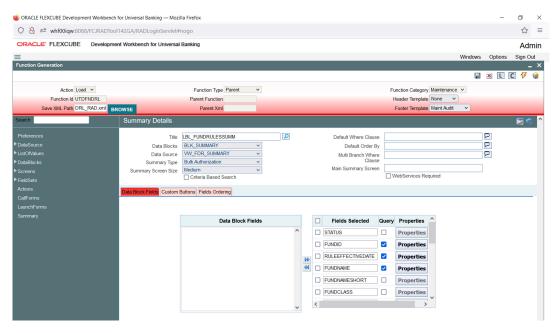


3. Click the Summary Node.



The **Summary Details** screen displays.

Figure 2-37 Providing Properties for Summary Screen



4. On the **Summary Details** screen, specify the fields.

**Table 2-2 Summary Details** 

Field	Description
Title	Enter the Summary title. Select label code from LOV.
Data Blocks	Select Data Block master block and summary blocks will be displayed. Select a required block from the drop-down list.
Data Source	Select Data Source for summary.
Summary Type	Select Summary Type.
Summary Screen Size	Select Summary Screen size.
Default Where Clause	Enter if any where clause is required.
Default Order By	Enter Default order by if required.
Multi Branch Where Clause	Enter Multi Branch where clause if required.

- **5.** Attach the fields required in the summary result grid.
  - If the field is required as part of filtering, the query has to be checked for the particular field.
- 6. Provide the position of fields in the **Result** grid and **Summary Query** set.
- 7. For summary preview, right-click the **Summary** node and click the **Preview**.
  - The Preview of the designed summary screen displays.



◆ Fund Rules Summary Execute Query C Advanced Search P Reset Clear All Authorization Status Record Status Fund ID Rule Effective Date Fund Name Q Fund Type Q AMC ID Q Fund Enabled? Latest Rule Fund Identification Number Q Fund Ticker Symbol Q Umbrella Fund Hedge Fund Records per page 15 v 1 1 of 1 Fund Name Short Authorization Status Record Status Rule Effective Date Fund Name Fund Clas Status Fund ID Authorize View Changes Exit

Figure 2-38 Preview of the designed summary screen

## 2.10 Maintain Amendable fields

This topic provides systematic instructions to maintain amendable fields.

**Amendable Fields**: If the user needs to modify data of a particular field on unlock in workbench, the developer has to maintain fields as amendable.

1. Click the Action node.

The Amendable Details - QUERY screen displays.

Amendable Details --QUERY DataBlock Fields **Data Blocks** □ New Allowed □ Delete Allowed □ All Records □ Mandatory BLK\_FUND\_DETAILS BLK GENERAL OPERATING BLK\_MULTI\_LANGUAGE\_DTL Field Name Amendable ^ BLK\_SHARES\_CHARACTERISTICS **FUNDID** BLK\_HOLDING\_LIMIT RULEEFFECTIVEDATE BLK\_HOLDING\_LIMIT\_DETAILS FUNDSTARTDATE BLK\_TXN\_PROCESSING\_RULES FUNDNAME BLK\_TXN\_LIMIT\_INVESTOR FUNDNAMESHORT  $\checkmark$ BLK\_TXN\_LIMIT\_COMMUNICATION **FUNDCLASS** BLK\_SI\_RULES\_FREQUENCY BLK\_TXN\_RULES **FUNDTYPE** BLK\_FUND\_SALE\_COUNTRY FUNDFAMILY BLK\_FUND\_SALE\_COUNTRY\_DETAIL FUNDBASECURRENCY V BLK\_FUND\_RESIDENCY\_RESTRICTION FUNDCOUNTRY BLK\_FUND\_RESIDENCY\_RESTRICTION\_DETAILS FISCALSTARTYEAR BLK\_FUND\_TRANSACTION\_CURRENCY  $\checkmark$ FISCALENDYEAR BLK\_FUND\_TRANSACTION\_CURRENCY\_DETAILS AMCID BLK\_CORPORATE\_ACTION BROKERSALLOWED Ok Cancel

Figure 2-39 Amendable Details - QUERY

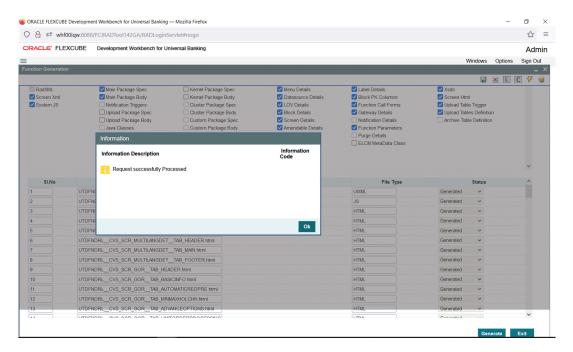
- 2. Click the **Amendable** button next to the action for which the field has to be made amendable.
- 3. Select the fields in each block which user can modify for the selected action.
- 4. Click the **Ok** button.

# Generate and Deploy Files

This topic provides systematic instructions to generate and deploy Files.

Select the required files and click the Generate button to generate files.
 The Information window displays with the status.

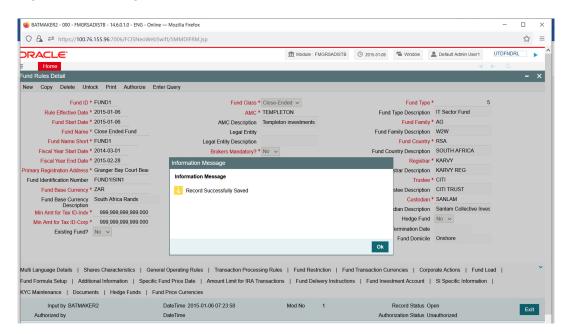
Figure 3-1 Generate Files



- 2. Select the required files to be deployed to the server and then click the **Deploy** button.
- 3. Click **Ok** on successful deployment.
- For testing, start the screen from FLEXCUBE and try sample operations (New, Modify, Query, etc.) on the screen.

On Successful deployment, save the record.

Figure 3-2 Saving Record for the function id in FLEXCUBE





4

# **Generated Units**

This topic provides an overview of generated units for a Maintenance screen.

The following units will be generated for a Maintenance screen.

Refer to the topic *Development WorkBench - ScreenDevelopment -II* for a detailed explanation on the same.

Front End Units

This topic provides an overview of the Front End Units.

Data Base Units

This topic provides an overview of the Data Base Units.

Other Units

This topic provides an overview of the Other Units.

## 4.1 Front End Units

This topic provides an overview of the Front End Units.

**Table 4-1** Front End Units

Front End Units	Description
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 specifically in that release.

## 4.2 Data Base Units

This topic provides an overview of the Data Base Units.

**Table 4-2 Data Base Units** 

Data Base Units	Description
Static Scripts	The following static scripts generated are required for the proper functioning of a <b>Call Form</b> screen. Refer document on generated units for a detailed explanation.
	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.
	2. Lov Details
	3. Amendable Details
	4. Label Details
	5. Screen Details
	6. Block Details
	7. Data Source Details
	8. Call form Details
	9. Summary Details
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 the database  Converting the Modified Composite Type again to TS  Each of these stages has <b>Pre</b> and <b>Post</b> 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



Table 4-2 (Cont.) Data Base Units

Data Base Units	Description
Hook Packages	Release-specific packages will be generated based on the release type (KERNEL, CLUSTER or CUSTOM). The 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_Pre_Check_Mandatory  • Fn_Pre_Default_and_Validate  • Fn_Pre_Upload_Db  • Fn_Pre_Upload_Db  • Fn_Post_Upload_Db  • 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 a similar structure and are for the respective teams to work on.

# 4.3 Other Units

This topic provides an overview of the Other Units.

**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 the **Actions** node.



# **Extensible Development**

This topic provides an overview of Extensible Development.

This topics contains following sub-topics:

- Extensibility in JavaScript Coding
   This topic provides an overview of extensibility in JavaScript Coding.
- Extensibility in Backend Coding
   This topic provides an overview of extensibility in backend coding for the Hook Packages.

# 5.1 Extensibility in JavaScript Coding

This topic provides an overview of extensibility in JavaScript Coding.

The developer can add code in hook packages and release specific JavaScript files.

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**.

For Example: Code in STDCIFD CLUSTER.js is exclusive to cluster release.

This JavaScript file allows the 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 ()

# 5.2 Extensibility in Backend Coding

This topic provides an overview of extensibility in backend coding for the Hook Packages.

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

This topic contains following sub-topics:

- Functions in Hook Packages
   This topic provides an overview of Functions in Hook Packages.
- Flow of control through Hook Packages
   This topic provides an overview of Flow of control through Hook Packages.
- Bypass Base Release Functionality
   This topic offers an overview of how to bypass Base Release Functionality.

## 5.2.1 Functions in Hook Packages

This topic provides an overview of 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. For 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 the 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.

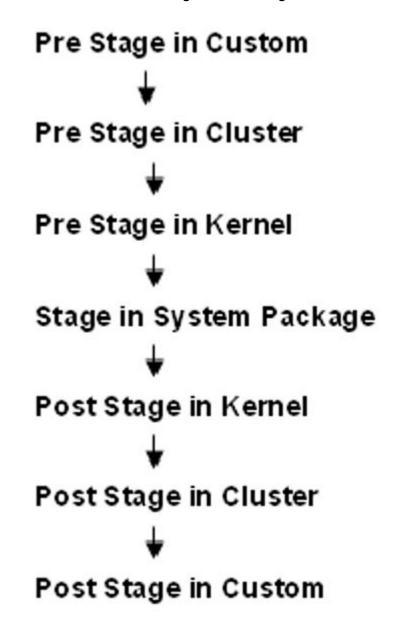
## 5.2.2 Flow of control through Hook Packages

This topic provides an overview of Flow of control through Hook Packages.

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



Figure 5-1 Flow of control through Hook Packages



#### **Example**: For **Fn\_check\_mandatory**, flow will be as:

UTPKS\_UTDFNDRL\_MAIN. Fn\_Check\_Mandatory

UTPKS\_UTDFNDRL\_CUSTOM.Fn\_Pre\_Check\_Mandatory

UTPKS\_UTDFNDRL\_CLUSTER.Fn\_Pre\_Check\_Mandatory

UTPKS\_UTDFNDRL\_KERNEL.Fn\_Pre\_Check\_Mandatory

UTPKS\_UTDFNDRL\_MAIN .Fn\_Sys\_Check\_Mandatory

UTPKS\_UTDFNDRL\_KERNEL.Fn\_Check\_Mandatory

UTPKS\_UTDFNDRL\_CLUSTER.Fn\_Check\_Mandatory

UTPKS\_UTDFNDRL\_CLUSTER.Fn\_Check\_Mandatory



## 5.2.3 Bypass Base Release Functionality

This topic offers an overview of how to bypass Base Release Functionality.

There are auto-generated functions like **FN\_SKIP\_<RELEAE\_TYPE>** which would determine whether or not a particular hook needs to be called.

The developer also has an option to bypass the base release hook if need be.

For example, if the validations are written in

**UTPKS\_UTDFNDRL\_KERNEL.FN\_PRE\_CHECK\_MANDATORY** are not required or not suitable for the Cluster release, the system provides an option to bypass the code written by the 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 reactivate the hooks of parent release.

The developer should refrain from adding validations or checks in the Pre Stage of any function, such as **Fn\_Pre\_Check\_Mandatory**, and instead focus on implementing all validations in the **Fn\_Post\_Default\_and\_Validate** stage.

For Example: The flow for the Mandatory Stage for STDCIFD:



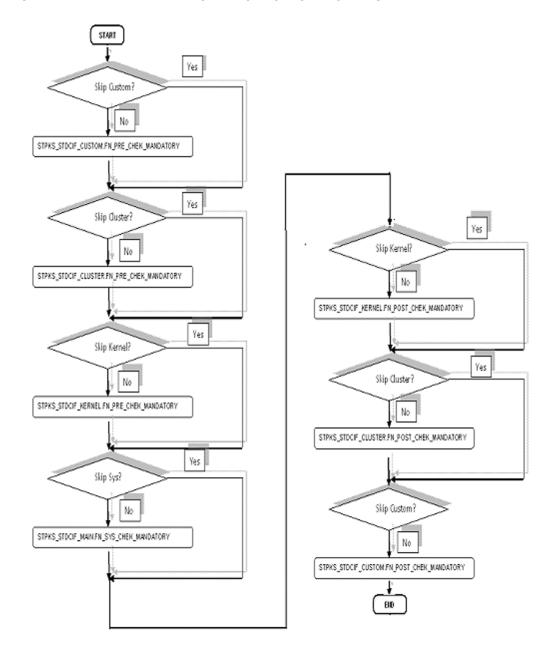


Figure 5-2 Flow of control explaining skip logic in packages

