

Oracle® Banking Treasury Management

Development of Call Forms



Release 14.7.6.0.0
April 2025

ORACLE®

Copyright © 2020, 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

Preface

Purpose	v
Audience	v
Documentation Accessibility	vi
Critical Patches	vi
Diversity and Inclusion	vi
Conventions	vi
Screenshot Disclaimer	vii
Prerequisite	vii
Related Resources	vii

1 Overview of Call Form

2 Screen Development

2.1	Header Information	2-1
2.2	Preferences	2-3
2.3	Data Sources	2-4
2.4	Data Blocks	2-5
2.5	Screens	2-6
2.6	Field Sets	2-8
2.7	Actions	2-9
2.8	Launch Forms	2-9
2.9	Call Forms	2-9
2.10	Summary	2-10
2.11	Preview	2-10

3 Attach Call Form to Main Function Id

4 Generated Units

4.1	Front End Units	4-1
-----	-----------------	-----

4.1.1	Language xml	4-1
4.1.2	SYS JavaScript File	4-1
4.1.3	Release Type Specific JavaScript File	4-1
4.2	Data Base Units	4-2
4.2.1	Static Scripts	4-2
4.2.2	System Packages	4-2
4.2.3	Hook Packages	4-3
4.3	Other Units	4-3
4.3.1	Xsd	4-3

5 Extensible Development

5.1	Extensibility in JavaScript Coding	5-1
5.2	Extensibility in Backend Coding	5-1

Preface

This document describes the steps to develop the notification XML and notification trigger using Oracle FLEXCUBE Development Workbench.

- [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.

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
Working knowledge of Web based Applications	Self-Acquired
Working knowledge of Oracle Database	Oracle Documentations
Working knowledge of PLSQL developer	Respective vendor documents

Table 1 (Cont.) Proficiency and Resources

Proficiency	Resources
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.
<i>italic</i>	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 Maintenance 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 Call Form

This topic provides an overview on Call Form.

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

Example: Tax Processing for a Contract

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

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

There are two types of Call forms they are:

1. Maintenance Call Forms
2. Transaction Call Forms

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

2

Screen Development

This topic describes about the development of screen.

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

Refer to the topic *Development Workbench - Screen Development I* for detailed explanation on this.

This topic contains the following sub-topics:

- [Header Information](#)
This topic describes about defining the header information for Call Forms.
- [Preferences](#)
This topic describes about defining the preferences for Call Forms.
- [Data Sources](#)
This topic describes about defining the data sources for Call Forms.
- [Data Blocks](#)
This topic describes about defining the data blocks for Call Forms.
- [Screens](#)
This topic describes about designing the screens for Call Forms.
- [Field Sets](#)
This topic describes about defining the field sets for Call Forms.
- [Actions](#)
This topic describes about the actions screen for Call Forms.
- [Launch Forms](#)
This topic describes about the launch forms.
- [Call Forms](#)
This topic describes about the launch forms.
- [Summary](#)
This topic describes about the summary.
- [Preview](#)
This topic describes about the preview.

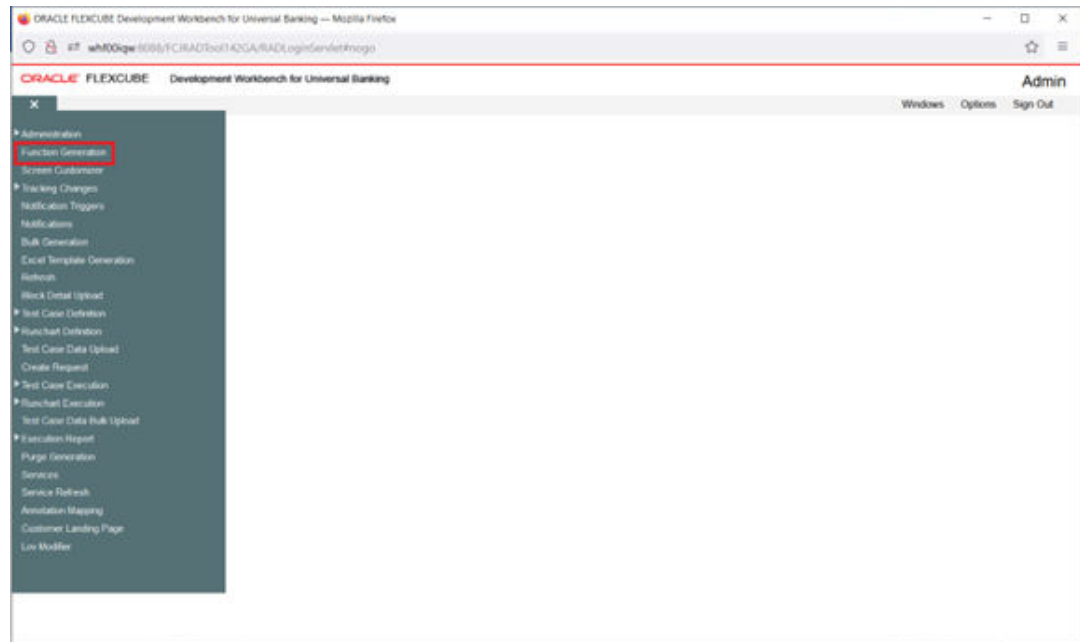
2.1 Header Information

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

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

The **Function Generation** screen displays.

Figure 2-1 Function Generation



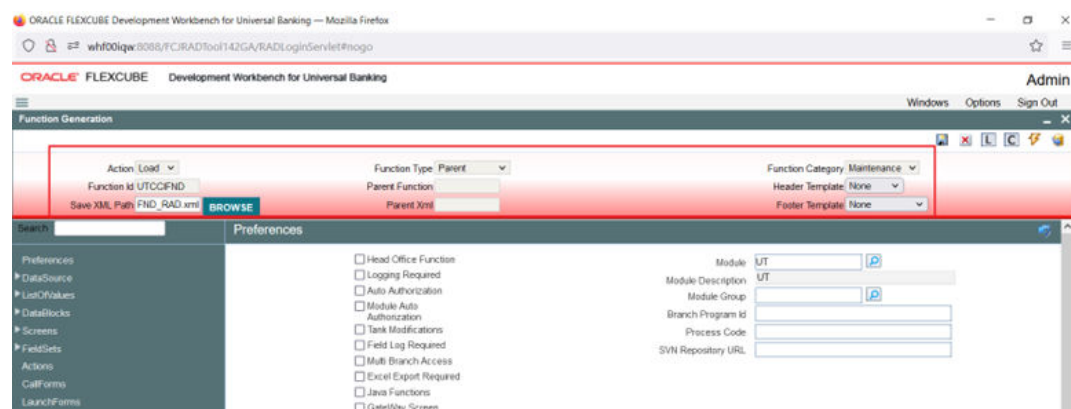
2. On the **Function Generation** screen, specify the following fields in the **Header** section for **CallForms**.

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 Call Form. Call Form name has to have the third character as C . This is how system differentiates a call form from other screens. Ideally, the length of the name should be 8 characters. Example: UTCCIFND, UTCAPPUH etc. are valid call form names.
Function Category	It is the Call Form Category. It has to be either Maintenance or Transaction depending on the functionality and the screens from which it will be invoked.
Footer Template	Select the footer template from the drop-down list. • None • Maint Audit • Maint Process • Process Footer template can be provided as required. Note for Transaction screens, footer template has to be selected as None unless it is a process screen.
Function Type	Parent and child functionality is supported for call forms.

Figure 2-2 Call Form header Information

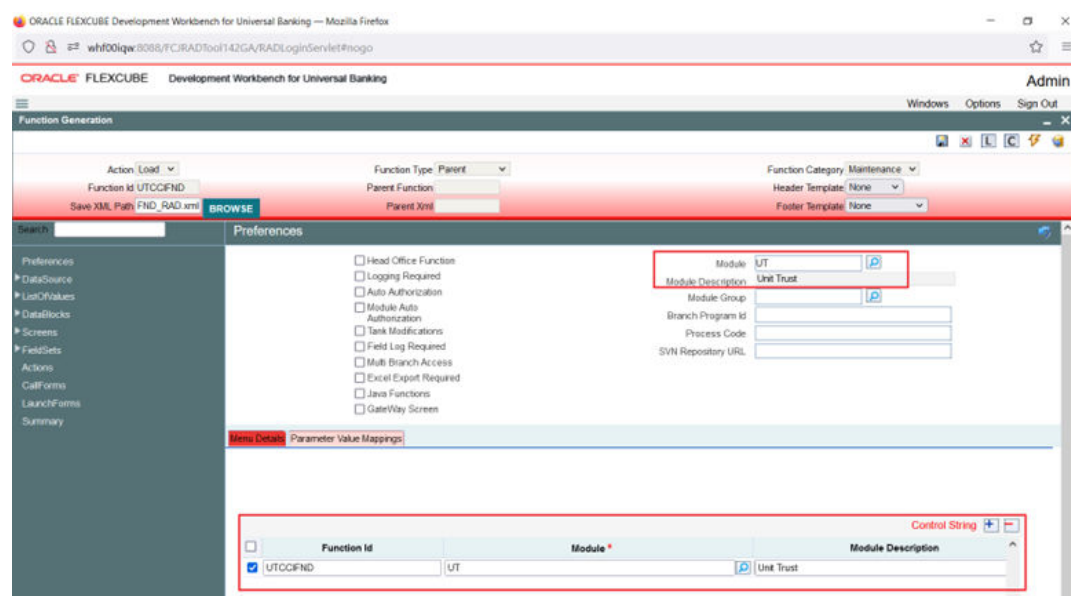


2.2 Preferences

This topic describes about defining the preferences for Call Forms.

1. Specify the menu details in the **Preferences** screen.

Figure 2-3 Call Form Preferences



2. On the **Preferences** screen, specify the following fields in the **Header** section for **CallForms**.

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

Table 2-2 Preferences - Field Description

Field	Description
Module	Module name is a mandatory field and has to be provided. It is recommended that the first two letters of the function id is kept as same as the module name. Naming of the generated package will be derived from the module code maintained.

Of the menu details inc generated, only script for SMTB_MENU and SMTB_FCC_FCJ_MAPPING is required for Call Forms.

3. **Browser menu options:** Call Forms cannot be launched independently .Hence browser menu labels need not be maintained. Script for smtb_function_description is not required for call forms.

2.3 Data Sources

This topic describes about defining the data sources for Call Forms.

1. Identify the tables/views for the call form. Define Data Sources and add data source fields as required in the **Data Source Details** screen.

Master Data Source has to be a single entry data source.

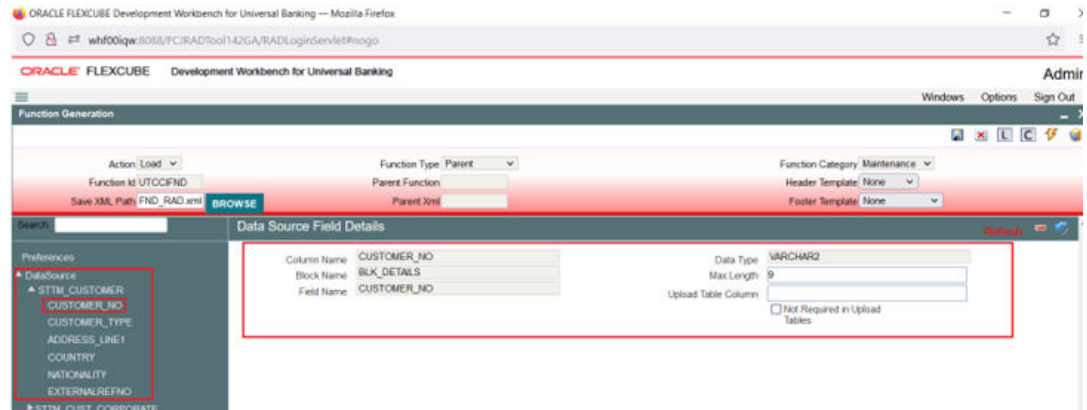
Figure 2-4 Adding data sources and maintaining properties

2. Maintain Logical Relationships for all data sources except the **Parent**.
3. Provide **PK Cols** and **PK Types** for all data sources.

Note:

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

Figure 2-5 Adding data sources fields and its properties

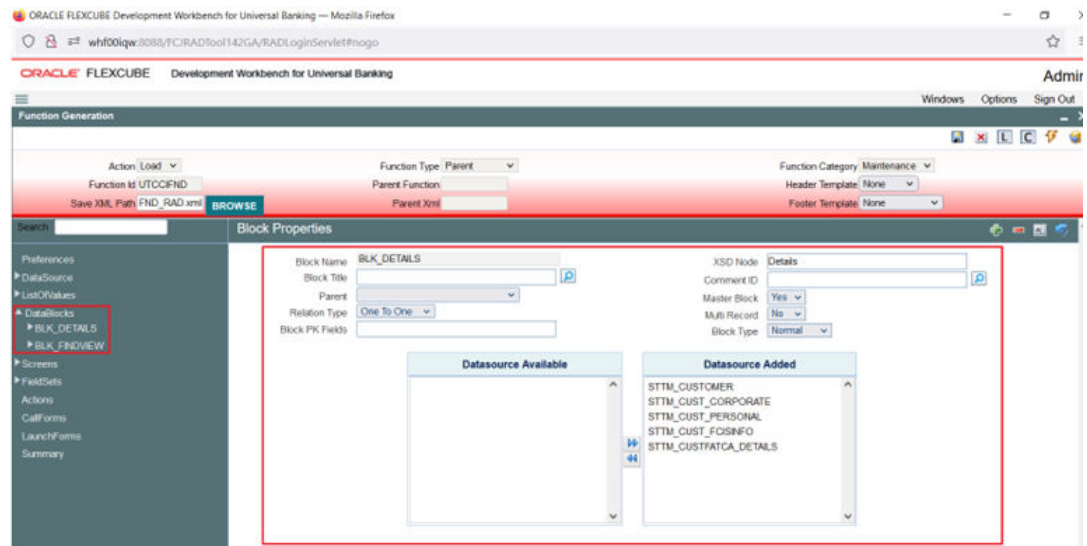


2.4 Data Blocks

This topic describes about defining the data blocks for Call Forms.

1. Determine the block structure for the function id .Define Data Blocks as per the design in the **Block Properties** screen.

Figure 2-6 Defining Data Blocks and maintaining its properties



2. Master Data Block has to be a single entry data block.
3. Logical Relationships with the parent has to be maintained for all data blocks.
4. Provide **XSD Node** name if the block is normal and is required in gateway request.

Figure 2-7 Attaching Block Fields and maintaining its properties

5. In case the block is not required in XSD, select the **Not Required in XSD** checkbox.
6. Ensure that **Related Block** and **Related Field** are given for Amount Fields.
7. Minimize the use of query data sources by using DESC fields wherever possible.

Note:

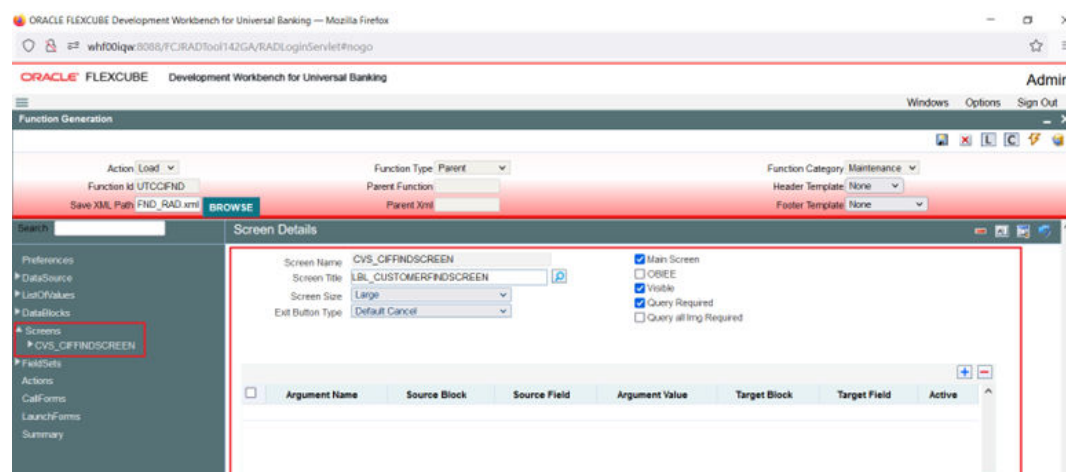
Query data sources is rarely required for a Call Form screen; as launch form can be used for query only screens.

2.5 Screens

This topic describes about designing the screens for Call Forms.

1. Design the screen layout based on the requirement in the **Screen Details**.

Figure 2-8 Designing Screens and providing Screen Properties



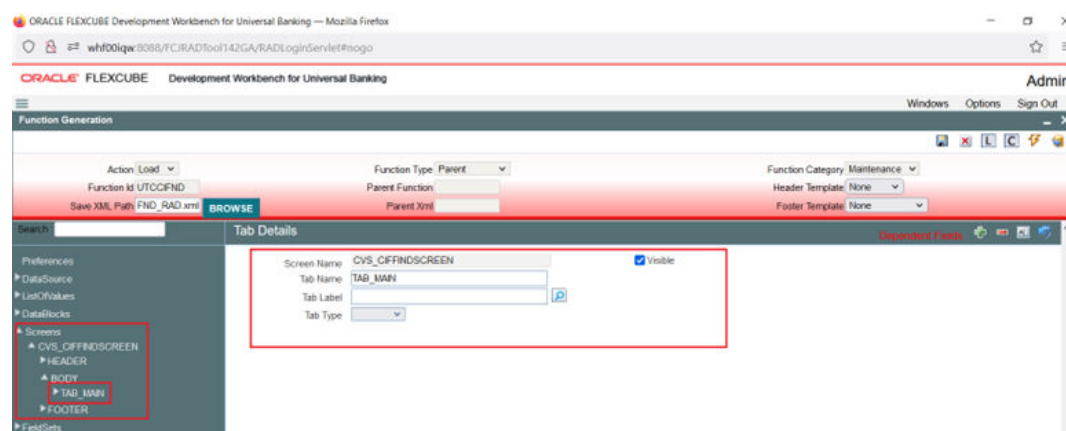
2. Identify one screen as the main screen; if multiple screens are present.
3. In the function id ,where the call form is called is for the button (which launches call form) events, mention the main screen of the call form.
4. Provide **Screen Arguments** for the main screen.

Any field which has to be populated based on the data from the calling Function id can be provided as the target block and target field.

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

5. Add Tabs, sections and partitions as per the screen design.

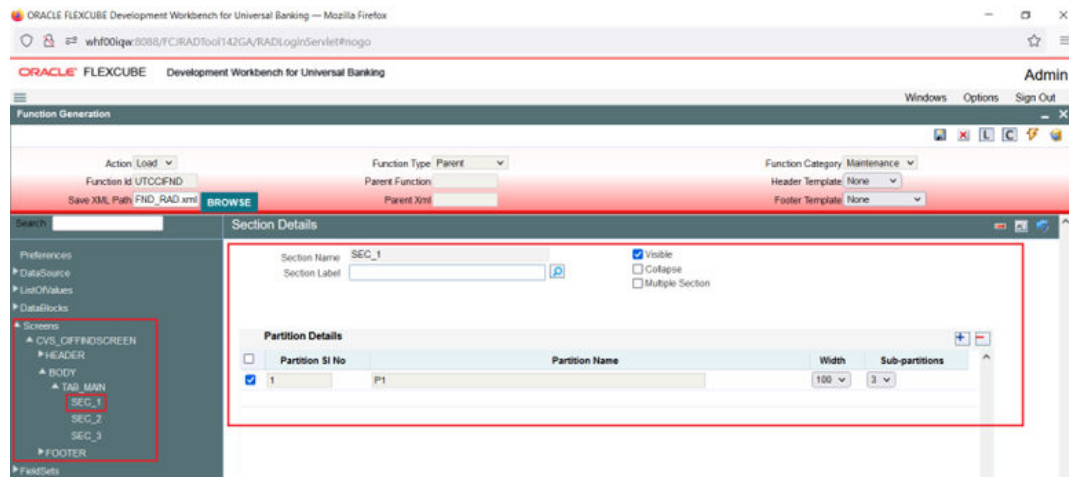
Figure 2-9 Creating Tabs and maintaining Properties



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

Multiple Screens can be designed if required.

Figure 2-10 Section Properties

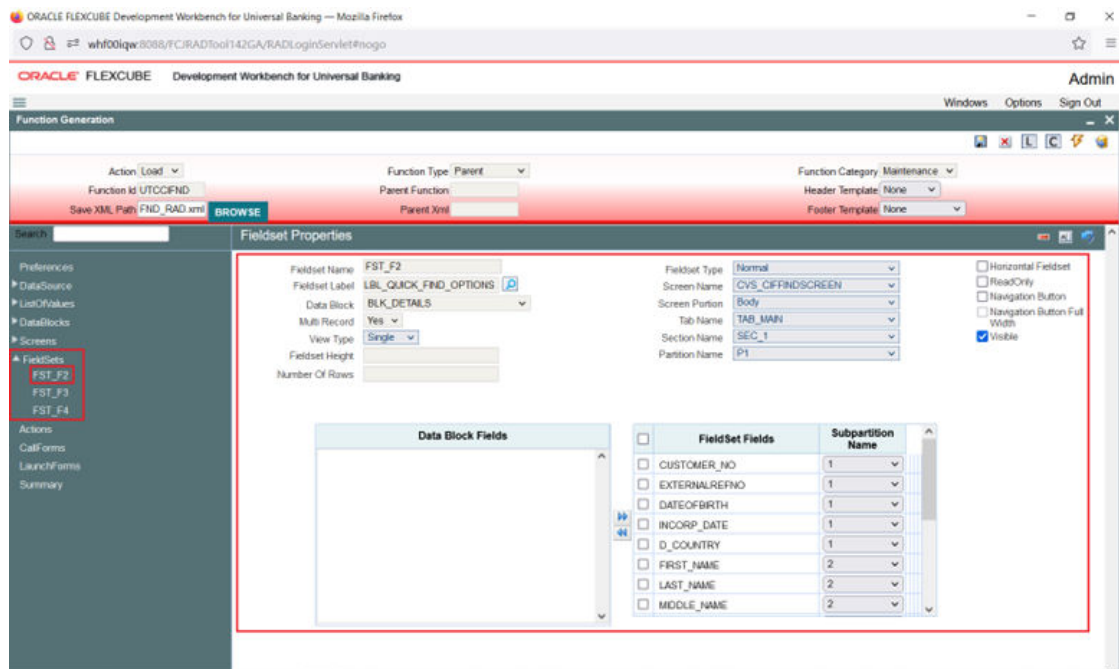


2.6 Field Sets

This topic describes about defining the field sets for Call Forms.

In the **Fieldset Properties** screen, create field sets and attach the fields to the field sets as required.

Figure 2-11 Field Set Properties



Note the following when attaching field to a field set:

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

2.7 Actions

This topic describes about the actions screen for Call Forms.

1. Mention the web service and amendable information in the **Form Actions** Screen.

Figure 2-12 Actions Screen

The screenshot shows the Oracle Flexcube Development Workbench for Universal Banking. The main window is titled 'Form Actions'. It features a top navigation bar with 'Admin' and 'Sign Out' options. Below the navigation bar, there are several dropdown menus and input fields for configuring the function generation process. The main area displays a table of actions with columns for Web Service, Action Code, Operation Code, Action Stage Type, Rest Enabled, Amendables, and Comment Code. The table lists various actions like QUERY, NEW, MODIFY, AUTHORIZE, DELETE, CLOSE, REOPEN, REVERSE, ROLLOVER, CONFIRM, LIQUIDATE, and SUMMARYQUERY, each with corresponding checkboxes and text fields.

Web Service	Action Code	Operation Code	Action Stage Type	Rest Enabled	Amendables	Comment Code
☐	QUERY		☒	☐	Amendables	Comments
☐	NEW		☐	☐	Amendables	Comments
☐	MODIFY		☒	☐	Amendables	Comments
☐	AUTHORIZE		☒	☐	Amendables	Comments
☐	DELETE		☒	☐	Amendables	Comments
☐	CLOSE		☒	☐	Amendables	Comments
☐	REOPEN		☒	☐	Amendables	Comments
☐	REVERSE		☐	☐	Amendables	Comments
☐	ROLLOVER		☐	☐	Amendables	Comments
☐	CONFIRM		☐	☐	Amendables	Comments
☐	LIQUIDATE		☐	☐	Amendables	Comments
☐	SUMMARYQUERY		☐	☐	Amendables	Comments

2. Call forms will generate only Type XSD. Operation specific message xsd's will not be generated. Call form Type will be part of the main function Id xsd; hence separate message xsd is not required for call form **Subsys** will be added to the name of call form type xsd.

Example: The example given in the figure, name of the xsd generated will be SubSysTxnChgDtIs-Types.xsd.

3. You need not to maintain **Operation Id** and **Operation Code** for the above mentioned reason.
4. Maintain amendable information similar to any other function id's.

2.8 Launch Forms

This topic describes about the launch forms.

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

2.9 Call Forms

This topic describes about the launch forms.

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

2.10 Summary

This topic describes about the summary.

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

2.11 Preview

This topic describes about the preview.

The figure shows the preview of the call form screen developed.

Figure 2-13 Call Form Screen Preview

The screenshot displays the 'Customer Find Screen' interface. It features a 'Quick find Options' section with input fields for CIF Number, External Reference Number, Date Of Birth, Incorporation Date, Country Of Domicile, First name, Last Name, Middle Name, Company Name, Nationality, Investor Type (a dropdown menu set to 'Both'), Tax ID, National ID, and IdentificationNumber. A 'Find' button is located at the bottom right of this section. Below the search options is an 'Advanced Find Options' section, which is currently collapsed. The 'Find Results' section shows a table with columns: Investor Type, First name, Last Name, Middle Name, Company Name, Customer No., and Tax I. The table contains one row of data. At the bottom right of the screen is an 'Exit' button.

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

3

Attach Call Form to Main Function Id

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

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



Note:

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

4

Generated Units

This topic describes about the generated units.

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

- [Front End Units](#)
This topic describes about the front end units.
- [Data Base Units](#)
This topic describes about the data base units.
- [Other Units](#)
This topic describes about the other units used in the module.

4.1 Front End Units

This topic describes about the front end units.

This topic consists of sub-topics;

- [Language xml](#)
This topic describes about the language xml.
- [SYS JavaScript File](#)
This topic describes about the SYS javaScript file.
- [Release Type Specific JavaScript File](#)
This topic describes about the release type specific javascript file.

4.1.1 Language xml

This topic describes about the language xml.

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

4.1.2 SYS JavaScript File

This topic describes about the SYS javaScript file.

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

4.1.3 Release Type Specific JavaScript File

This topic describes about the 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.

4.2 Data Base Units

This topic describes about the data base units.

This topic consists of sub-topics:

- [Static Scripts](#)
This topic describes about the static scripts.
- [System Packages](#)
This topic describes about the system packages.
- [Hook Packages](#)
This topic describes about the hook packages.

4.2.1 Static Scripts

This topic describes about the 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.

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

4.2.2 System Packages

This topic describes about the system packages.

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

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

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

Main package contains functions for :

- Converting Ts to PL/SQL Composite Type
- Calling fn_main.

- Mandatory checks (fn_check_mandatory).
- Default and validation(fn_default_and_validate)
- Querying(fn_query)
- Converting the Modified Composite Type again to TS

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

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

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

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

4.2.3 Hook Packages

This topic describes about the hook packages.

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

4.3 Other Units

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

This topic consists of sub-topics:

- [Xsd](#)
This topic describes about the Xsd.

4.3.1 Xsd

This topic describes about the Xsd.

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

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

5

Extensible Development

This topic describes about the extensible development.

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

This topic contains the following sub-topics:

- [Extensibility in JavaScript Coding](#)
This topic describes about the extensibility in javascript coding.
- [Extensibility in Backend Coding](#)
This topic describes about the extensibility in backend coding.

5.1 Extensibility in JavaScript Coding

This topic describes about the extensibility in javascript coding.

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

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

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

- Functions triggered by screen loading events
Example: fnPreLoad_CLUSTER(), fnPostLoad_CLUSTER()
- Functions triggered by screen action events
Example: fnPreNew_CLUSTER (), fnPostNew_CLUSTER ()

5.2 Extensibility in Backend Coding

This topic describes about the extensibility in backend coding.

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



Note:

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

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

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

2. Fn Main
This is called from the fn_main in main package.
3. Fn_pre_query
4. Fn_post_query
Any specific logic while querying can be written in these functions. It is called from fn_query of the main package
5. Fn_pre_upload_db
6. Fn_post_upload_db
Any logic while uploading data to tables can be written here.
7. Fn_pre_default_and_validate
8. Fn_post_default_and_validate
Any release specific logic for defaulting and validation can be written here . It is called from the fn_default_and_validate in the main package
9. Fn_pre_check_mandatory
10. Fn_post_check_mandatory
Any mandatory checks can be validated here.
11. Fn_pre_process
12. Fn_post_process
These hook functions are specific to transaction call from screens. These are called from fn_process of the main package which in turn is called from fn_process of the calling function id

Refer maintenance and Transaction Screen development document for further explanation.