

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



Table of Contents

1. Preface	3
1.1 Audience	3
1.2 Related Documents	3
2. Introduction	4
2.1 How to use this Guide	4
3. Overview of Online Form	5
4. Screen Development	6
4.1 Header Information	6
4.2 Preferences	7
4.3 Data Sources	8
4.4 Data Blocks	9
4.5 Screens	11
4.6 Field Sets	13
4.7 Actions	14
4.8 Launch Forms	15
4.9 Call Forms	15
4.9.1 Sub System Pickup/Processing	16
4.10 Summary	17
4.11 Preview	18
5. Generated Units	19
5.1 Front End Units	19
5.1.1 Language xml	19
5.1.2 SYS JavaScript File	19
5.1.3 Release Type Specific JavaScript File	19
5.2 Database Units	19
5.2.1 Static Scripts	19
5.2.2 System Packages	19
5.2.3 Hook Packages	20
5.3 Other Units	20
5.3.1 Xsd	20
6. Extensible Development	21
6.1 Extensibility in JavaScript Coding	21
6.2 Extensibility in Backend Coding	21

1. Preface

This document describes the features of Online Forms in FLEXCUBE and the process of designing an Online form screen using Oracle FLEXCUBE Development Workbench for Universal Banking

1.1 Audience

This document is intended for FLEXCUBE Application developers/users that use Development Workbench to develop various FLEXCUBE components.

To Use this manual, you need conceptual and working knowledge of the below:

<i>Proficiency</i>	<i>Resources</i>
FLEXCUBE Functional Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Screen Development	<i>04-Development_WorkBench_Screen_Development-I.docx</i>
Working knowledge of Web based applications	Self-Acquired
Working knowledge of Oracle Database	Oracle Documentations
Working knowledge of PLSQL & SQL Language	Self-Acquired
Working knowledge of XML files	Self-Acquired

1.2 Related Documents

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

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

2. Introduction

2.1 How to use this Guide

The information in this document includes:

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

3. Overview of Online Form

Online Forms are function Id's (screens) which is used for creating Contracts for respective modules. Same contracts can be processed further for Payments, Availments, Amendments, Reassignments and Authorizations also using Online forms.

All the transaction processing in FLEXCUBE is carried out through Online screens
Online form screens should be launched independently.

*Example: **Letter Of Credit (LC) contract***

An LC contract is an instruction wherein a customer requests the bank to issue, advice or confirm a letter of credit, for a trade transaction. An LC substitutes a bank's name and credit for that of the parties involved. The bank thus undertakes to pay the seller/beneficiary even if the remitter fails to pay.

Thus for each module we should develop different function Id's for creating contracts and others online forms for other operations like Payments, Availments, Amendments, Reassignments and Authorizations.

LCDTRONL - Contract Input
LCDAMEND - Amend Confirmation Input
LCDVMNT - Availment Input
LCDTRPAY - Payment Input
LCDTRANF - Transfer Input
LCDEPMNT - Manual Liquidation Input
LCDTREAS - Contract Reassign
LCDTRAUT - Amend Confirmation Input

On launching the Online form screen, user has to input the respective values to create the contract. Form may have the different user-defined actions like Product-Default, Enrich, and Subsystem-Pickup while creating contract. Once all the user-defined actions performed finally user has to save the contract.

4. Screen Development

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

For detailed explanation, refer the document: **04-Development_WorkBench_Screen_Development-I.docx**

4.1 Header Information

Provide the header information as shown in the figure.

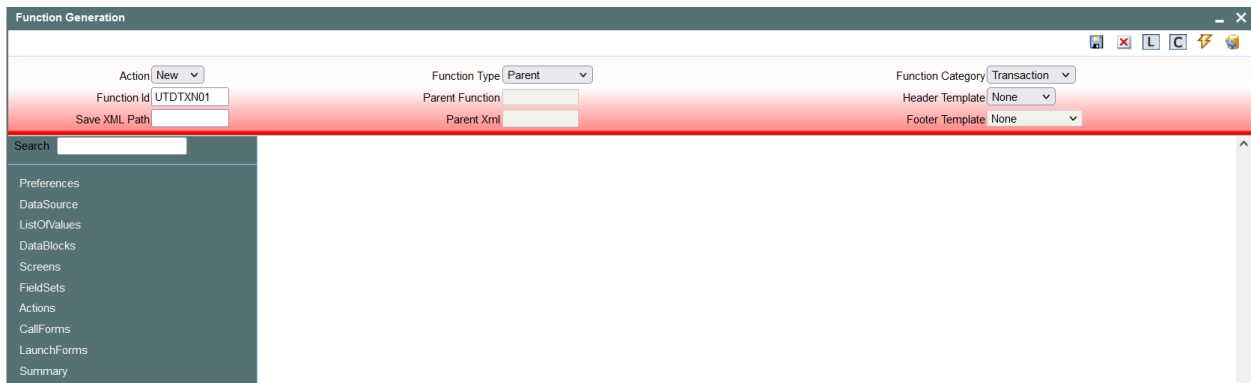
The screenshot shows a software window titled "Function Generation". It features a top header bar with a red gradient. Below this bar, there are several input fields and dropdown menus. On the left, there is a "Search" bar and a list of menu items: Preferences, DataSource, ListOfValues, DataBlocks, Screens, FieldSets, Actions, CallForms, LaunchForms, and Summary. The main area contains fields for "Action" (set to "New"), "Function Id" (set to "UTDTXN01"), "Save XML Path", "Function Type" (set to "Parent"), "Parent Function", "Parent Xml", "Function Category" (set to "Transaction"), "Header Template" (set to "None"), and "Footer Template" (set to "None").

Fig: Online Form header Information

Note the following while providing header information.

- i) Name of the Online form :
Online Form name has to have the third character as 'D'. Ideally, the length of the name should be 8 characters.
Example: *UTDTXN01 etc. are valid online form names*
- ii) Online Form Category:
Function Category has to be **Transaction**
- iii) Footer Template:
For Transaction screens, footer template has to be selected as **NONE**. System does not provide any default template for transaction screens; hence developer has to design the footer portion of the screen manually. Developer has to make sure that footer designed has generic fields like transaction status (TXNSTAT), authorization status (AUTHSTAT) etc.
For Online Process Flow Screens footer template should be selected as **PROCESS**.
- iv) Function Type :
Parent and Child functionality is supported for Online forms.

4.2 Preferences

Provide the menu details in the Preferences screen

ORACLE FLEXCUBE Development Workbench for Universal Banking — Mozilla Firefox

whf00iqw:8088/FCJ/RADTool142GA/RADLoginServlet#nogo

ORACLE FLEXCUBE Development Workbench for Universal Banking

Admin

Windows Options Sign Out

Function Generation

Action Load Function Type Parent Function Category Transaction

Function Id UTDTXN01 Parent Function Header Template: None

Save XML Path JN01_RAD.xml BROWSE Parent Xml Footer Template: None

Search

Preferences

Preferences

DataSource

ListOfValues

DataBlocks

Screens

FieldSets

Actions

CallForms

LaunchForms

Summary

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 UT

Module Description Unit Trust

Module Group

Branch Program Id

Process Code

SVN Repository URL

Menu Details Parameter Value Mappings

Control String + -

Function Id	Module *	Module Description
UTDTXN01	UT	Unit Trust
UTSTXN01	UT	Unit Trust

Fig: Online Form Preferences

Note the following while providing Preferences for Online Forms.

- Module name :
Module name is a mandatory field and has to be provided. It is recommended that the first two letters of the function id is kept as same as the module name. Naming of the generated package will be derived from the module code maintained
- Script for the following tables will be generated by Workbench (menu details) which are essential for launching of an Online screen.
 - SMTB_MENU
 - SMTB_FCC_FCJ_MAPPING
 - SMTB_FUNCTION_DESCRIPTION
 - SMTB_ROLE_DETAILSType string of the Online screens will be generated as 'O' in **smtb_menu** table.
- Transaction specific action codes has to checked in the control string whichever applicable
Example: LIQUIDATE, ROLLOVER, REVERSAL etc.

4.3 Data Sources

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

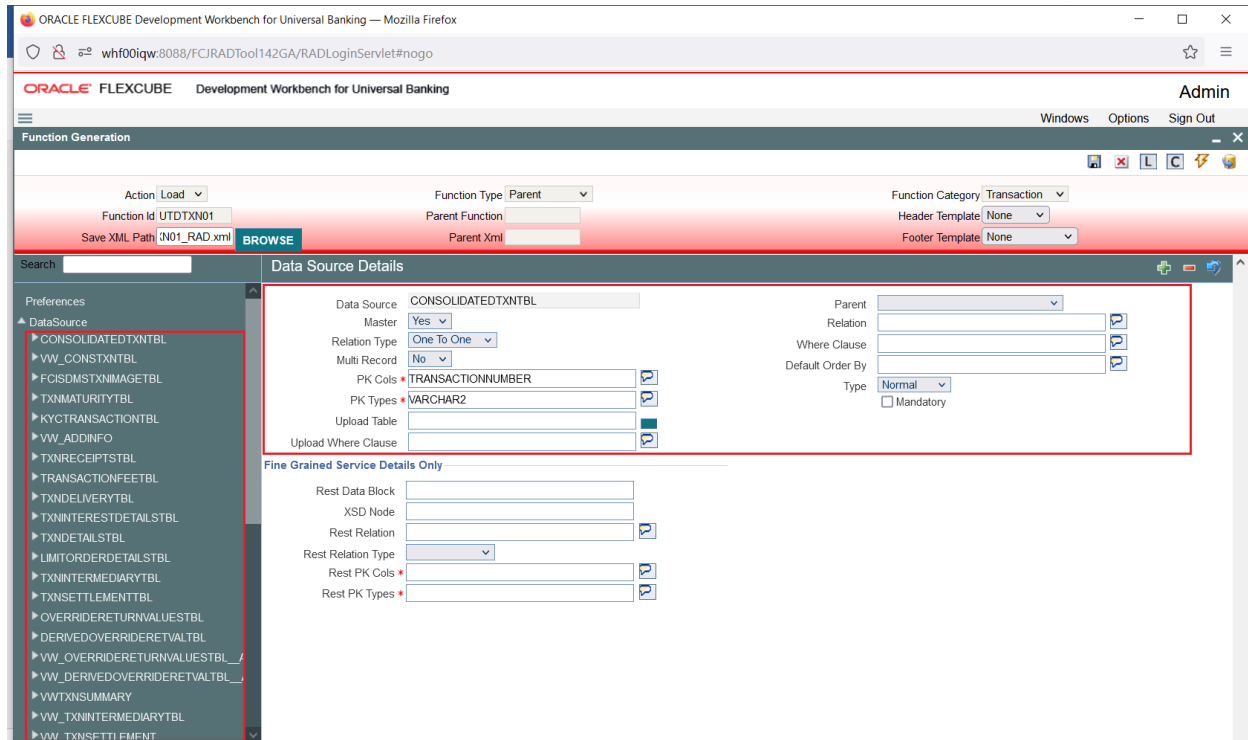


Fig: Adding data sources and maintaining properties

Note the following while creating data sources

- i) Master Data Source has to be a single entry data source.
- ii) Logical Relationships has to be maintained for all data sources except the parent
- iii) Provide PK Cols and PK types for all data sources.
If data source is a multi-record block, then make sure it has at least one more pk than its parent which helps to uniquely identify each record of multi record block
- iv) **Minimize the use of views in the data sources.** For transaction screens, system generated upload logic (fn_sys_upload_db) is not called within the system package. It is up to the developer to decide whether the system generated code can be used or not. **If views are used in data sources, then this function should not be used by the developer.**
- v) Usually for Online forms, a separate view can be used for summary purpose. This view will have all the fields required to be displayed in the summary.
Example: UTDTXN01_SUMMARY

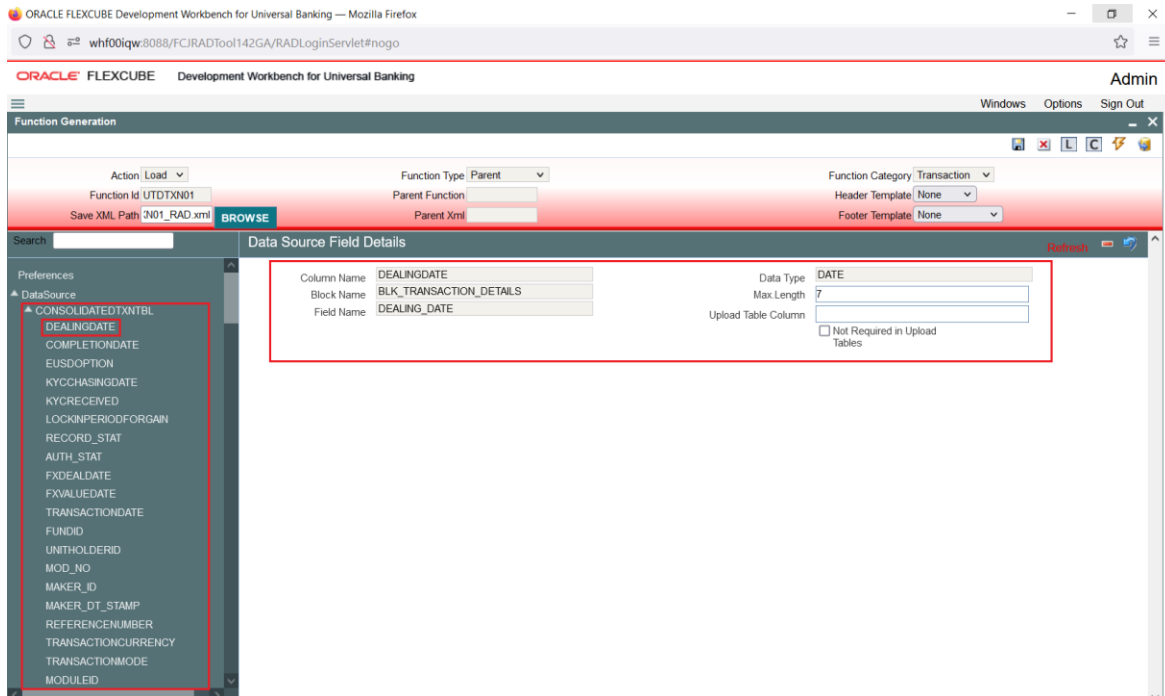


Fig: Adding data sources fields and its properties

Max length of the data source field can be modified as per requirement

4.4 Data Blocks

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

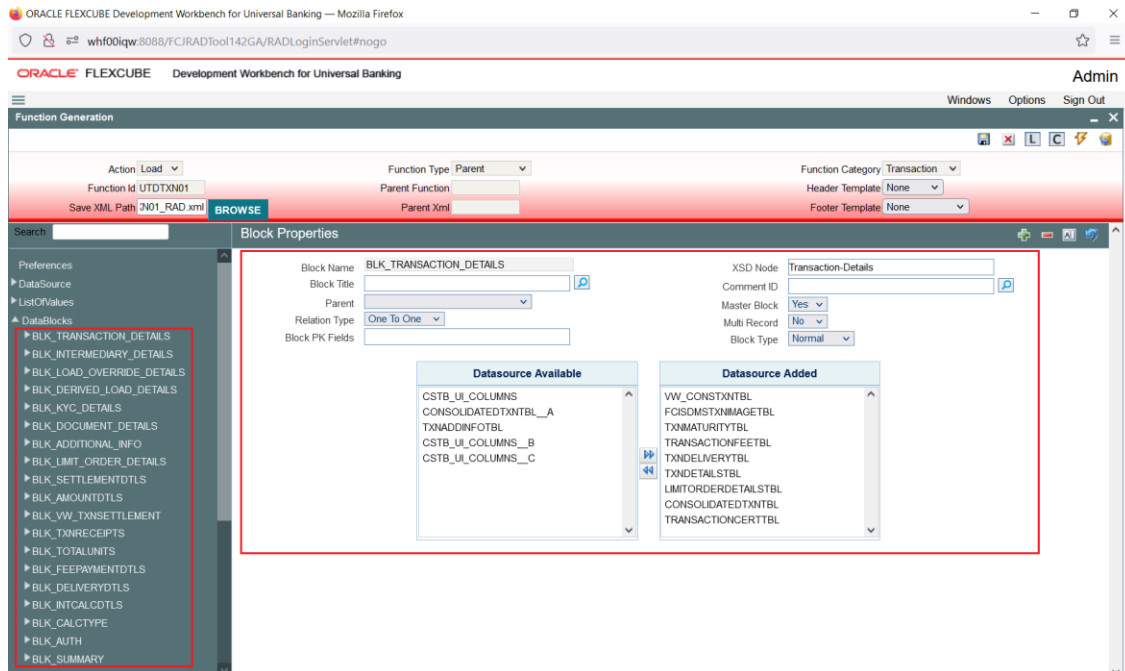


Fig: Defining Data Blocks and maintaining its properties

Note the following while creating data blocks

- i) Master Data block has to be a single entry data source.
- ii) Provide Xsd node name if the block is normal and is required in gateway request
- iii) Block order and block field order can be changed by re arranging blocks and block fields in the browser tree (drag and drop). **Note that all units will have to be regenerated if block or block field order is changed** (including xsd's)
- iv) Related currency fields should be placed above the amount field in the tree

Add block fields to the data block as required.

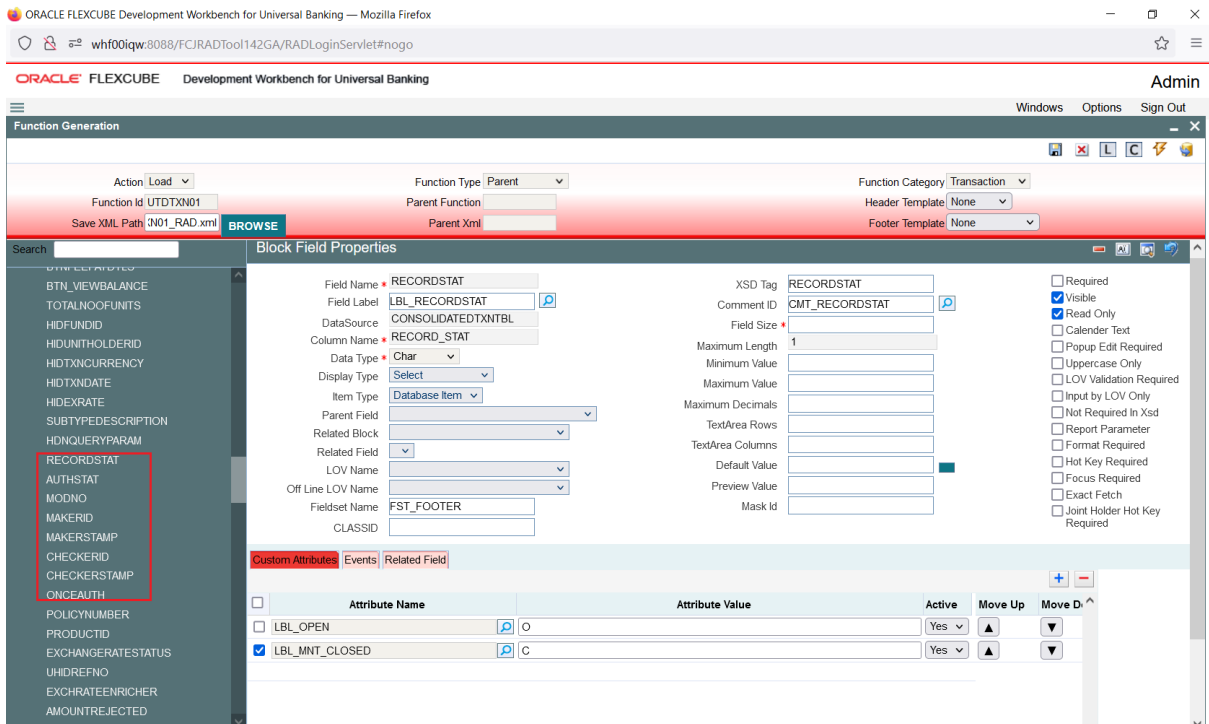
The screenshot shows the 'Block Field Properties' dialog in the Oracle Flexcube Development Workbench. The 'Field Name' is 'FUNDID', 'Field Label' is 'LBL_FUNDID', and 'DataSource' is 'CONSOLIDATEDTXNTBL'. The 'Column Name' is 'FUNDID' and 'Data Type' is 'Varchar2'. The 'Display Type' is 'Lov'. The 'Item Type' is 'Database Item'. The 'Parent Field' is 'LOV_FUND'. The 'Related Block' is 'BLK_TRANSACTION_DETAILS'. The 'Related Field' is 'FUNDID'. The 'LOV Name' is 'LOV_FUND'. The 'Off Line LOV Name' is 'FUNDID'. The 'Fieldset Name' is 'FST_TRANSACTION_INFO'. The 'CLASSID' is empty. The 'XSD Tag' is 'FUNDID' and 'Comment ID' is 'CMT_FUND_ID'. The 'Field Size' is '6'. The 'Maximum Length' is '6'. The 'Minimum Value' is empty. The 'Maximum Value' is empty. The 'Maximum Decimals' is empty. The 'TextArea Rows' is empty. The 'TextArea Columns' is empty. The 'Default Value' is empty. The 'Preview Value' is empty. The 'Mask Id' is empty. The 'Required' checkbox is checked. The 'Visible' checkbox is checked. The 'Read Only' checkbox is unchecked. The 'Calendar Text' checkbox is unchecked. The 'PopUp Edit Required' checkbox is unchecked. The 'Uppercase Only' checkbox is checked. The 'LOV Validation Required' checkbox is checked. The 'Input by LOV Only' checkbox is unchecked. The 'Not Required In Xsd' checkbox is unchecked. The 'Report Parameter' checkbox is unchecked. The 'Format Required' checkbox is unchecked. The 'Hot Key Required' checkbox is unchecked. The 'Focus Required' checkbox is unchecked. The 'Exact Fetch' checkbox is unchecked. The 'Joint Holder Hot Key Required' checkbox is unchecked.

Query Column	Block Name	Return Field Name
FUNDID	BLK_TRANSACTION_DETAILS	FUNDID
FUNDIDENTIFICATIONNUM	BLK_TRANSACTION_DETAILS	FUNDIDENTIFICATIONNUM
FUNDNAME	BLK_TRANSACTION_DETAILS	FUNDNAME
FUNDBASECURRENCY	BLK_TRANSACTION_DETAILS	FUNDBASECURRENCY

Fig: Attaching Block Fields and maintaining its properties

Note the following while attaching block fields to data blocks

- i) In case the field is not required in XSD, check not Required XSD
- ii) Ensure that Related Block and Field are given for Amount Fields
- iii) Minimize the use of query data sources by using DESC fields wherever possible.
Note: Query data sources is rarely required for an Online Form screen; as launch form can be used for query only screens
- iv) Master block should contain reserved field names like AUTHSTAT, RECORDSTAT, ONCEAUTH, MODNO, MAKERID, CHECKERID, MAKERDTSTAMP and CHECKERDTSTAMP are added as part of the footer of the screen.



4.5 Screens

Design the screen layout based on the requirement

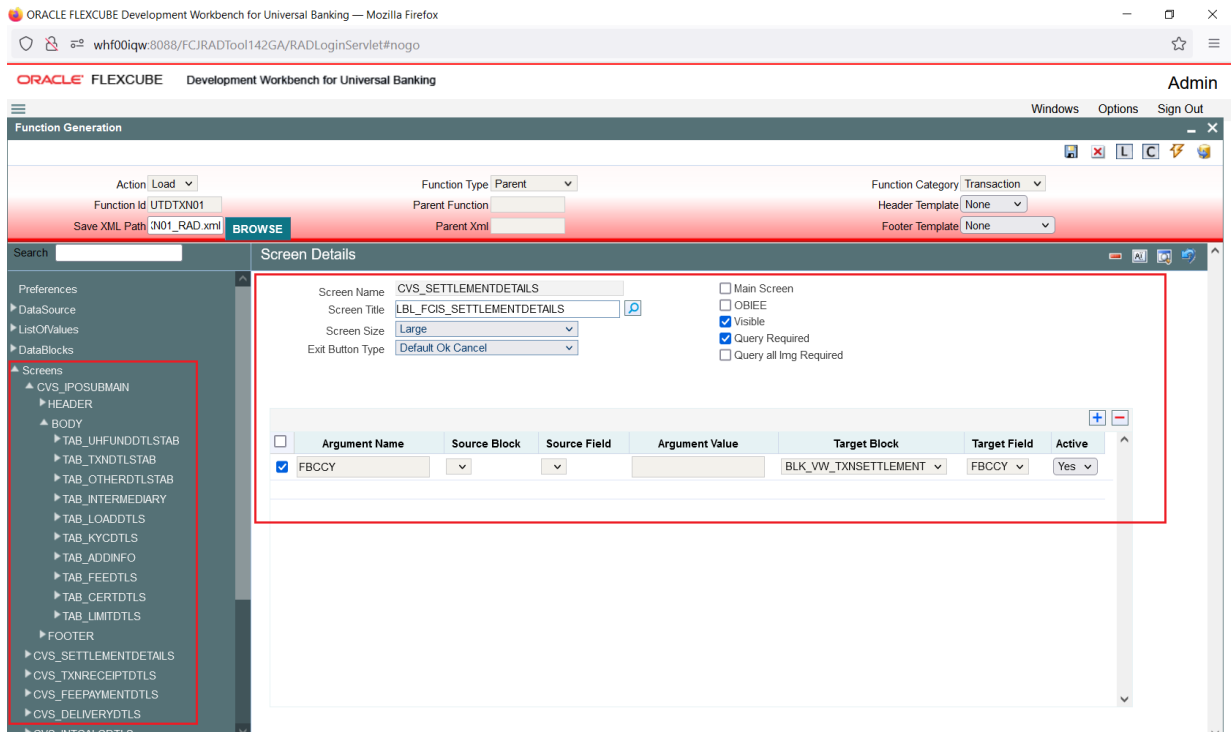


Fig: Designing Screens and providing Screen Properties

Note the following while creating screens

- One Screen should be identified as the main screen.

Add Tabs, sections and partitions as per the screen design

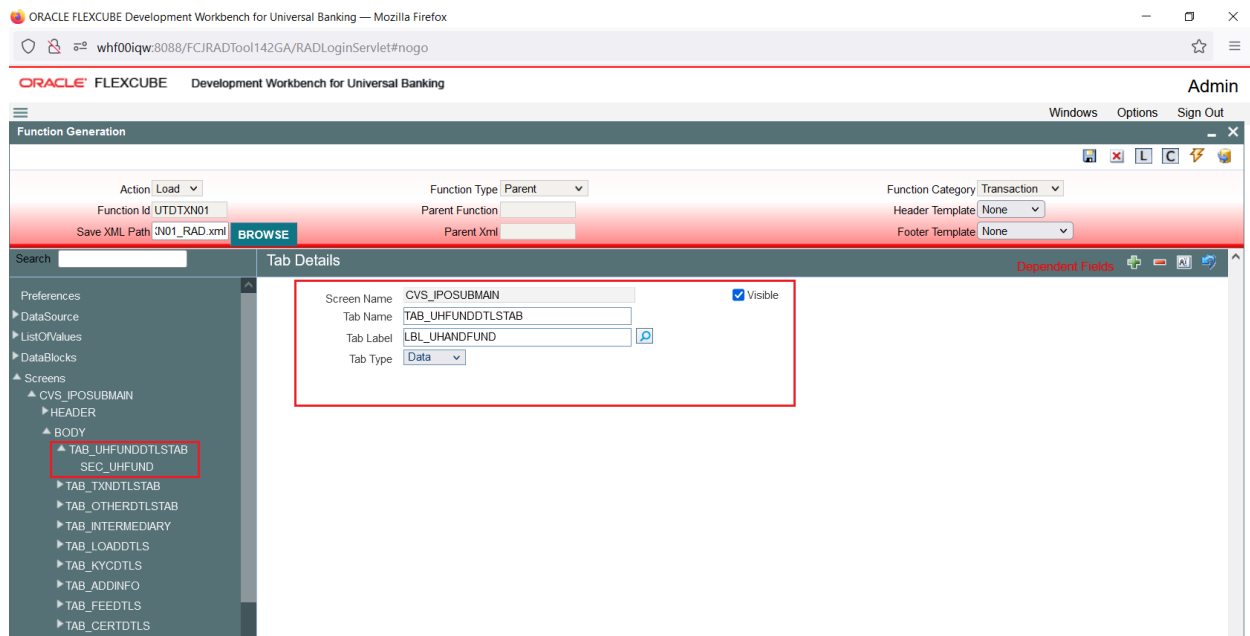


Fig: Creating Tabs and maintaining Properties

Note the following when creating tabs and sections for the screen

- 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
- Normally Online forms are large screens with multiple tabs. In this case, all the tabs needs to be used .TAB_HEADER should contain the header information. TAB_MAIN should be the first tab in the body .Other tabs has to be added in the body portion as required
- Footers are often designed by the developer for Online forms. Provide sections in TAB_FOOTER as required. **Note that in large screens ,footer supports 4 partitions while other portions support 3 partitions**

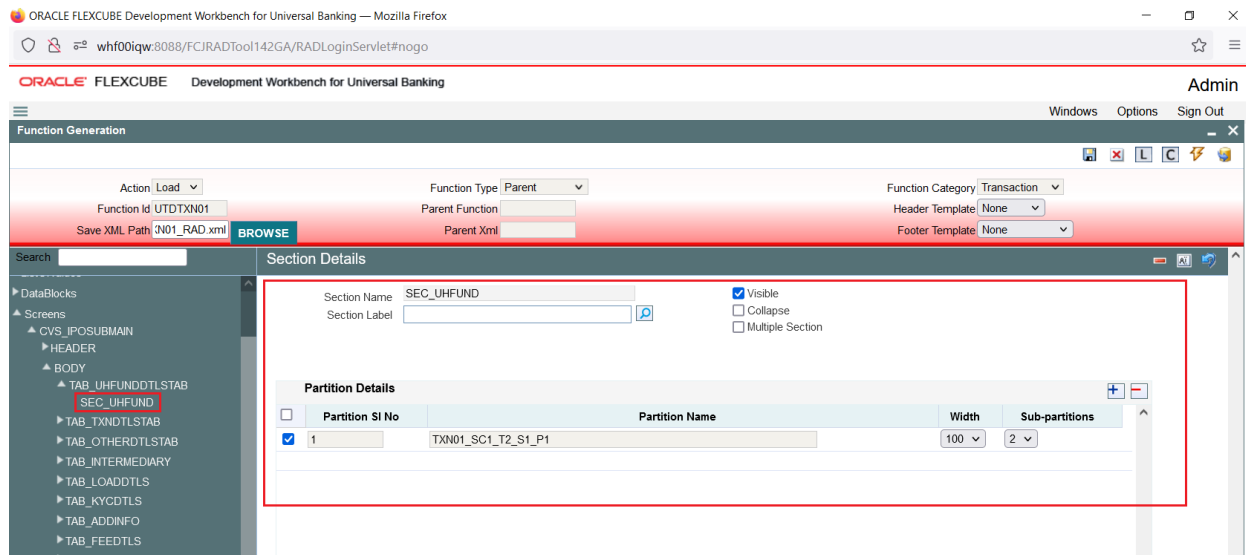


Fig: Section Properties

Multiple Screens can be designed if required.

4.6 Field Sets

Create Field sets and attach the fields to the field sets as required

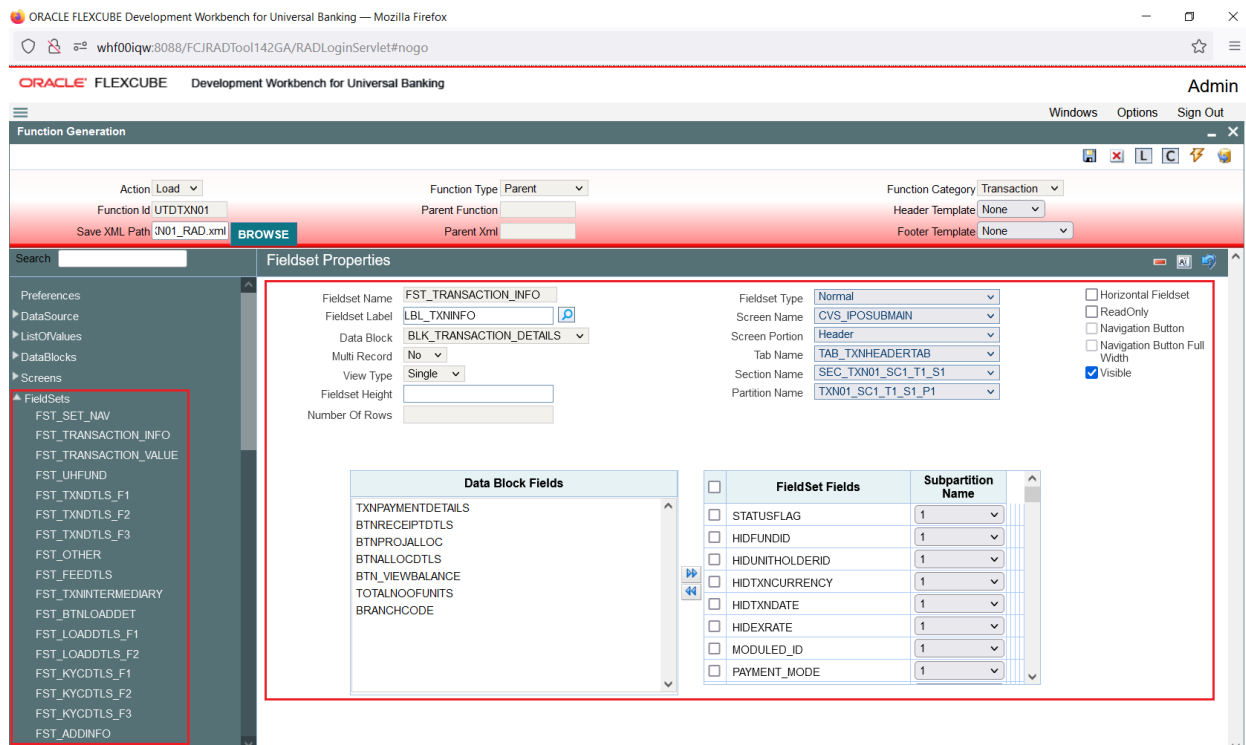


Fig: Field Set Properties

Note the following when attaching field to a field set

- i) If a field is not required in the screen, but kept as hidden and value defaulted; then **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 accessing the field.

4.7 Actions

Mention the web service and amendable information in Actions Screen

ORACLE FLEXCUBE Development Workbench for Universal Banking

Function Generation

Action Load: Load Function Id: UTDTXN01 Function Type: Parent Parent Function: Parent Xml: Save XML Path: N01_RAD.xml BROWSE Function Category: Transaction Header Template: None Footer Template: None

Search: Form Actions

XSD Type Identifier: IPOSubscription Operation Id: IPOSubscription Service Name: FCISIPoSubscription XSD Module Folder: Rest Service Name: IPOSubscription

☒ Rest Enabled ☐ Fine Grained Service Required

Web Service	Action Code	Operation Code	Action Stage Type	Rest Enabled	Amendables	Comment Coc
<input checked="" type="checkbox"/>	QUERY	QueryIPOSubscription	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Amendables	Comments
<input checked="" type="checkbox"/>	NEW	CreateIPOSubscription	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Amendables	Comments
<input checked="" type="checkbox"/>	MODIFY	ModifyIPOSubscription	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Amendables	Comments
<input checked="" type="checkbox"/>	AUTHORIZE	AuthorizeIPOSubscription	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Amendables	Comments
<input checked="" type="checkbox"/>	DELETE	DeleteIPOSubscription	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Amendables	Comments
<input type="checkbox"/>	CLOSE		<input type="checkbox"/>	<input type="checkbox"/>	Amendables	Comments
<input type="checkbox"/>	REOPEN		<input type="checkbox"/>	<input type="checkbox"/>	Amendables	Comments
<input checked="" type="checkbox"/>	REVERSE	ReverseIPOSubscription	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Amendables	Comments
<input type="checkbox"/>	ROLLOVER		<input type="checkbox"/>	<input type="checkbox"/>	Amendables	Comments
<input type="checkbox"/>	CONFIRM		<input type="checkbox"/>	<input type="checkbox"/>	Amendables	Comments
<input type="checkbox"/>	LIQUIDATE		<input type="checkbox"/>	<input type="checkbox"/>	Amendables	Comments
<input type="checkbox"/>	SUMMARYQUERY		<input type="checkbox"/>	<input type="checkbox"/>	Amendables	Comments

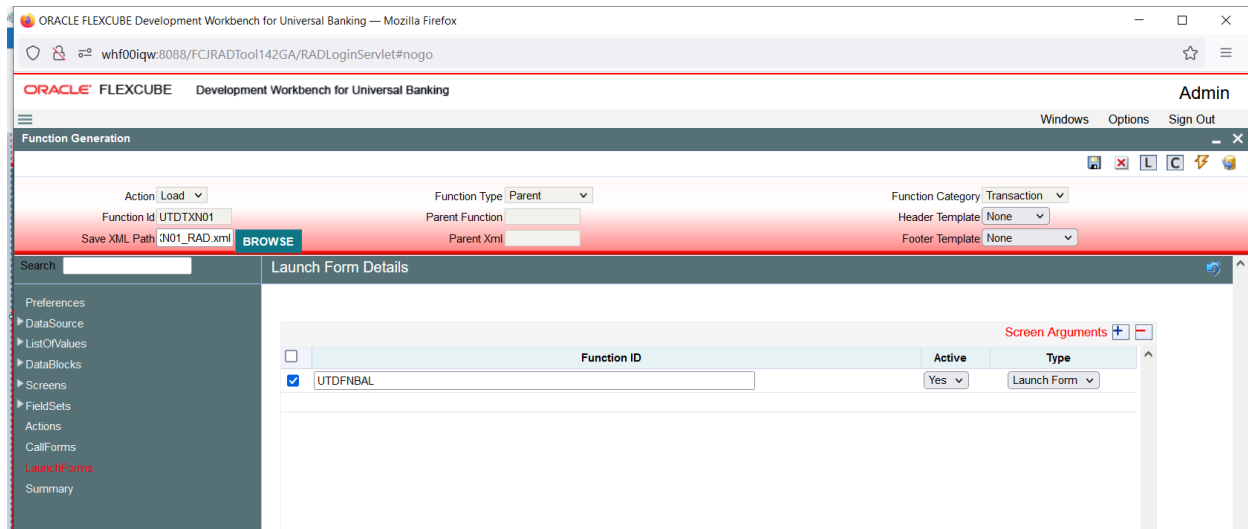
Fig: Actions Screen

Note the following while maintaining web services and amendable information

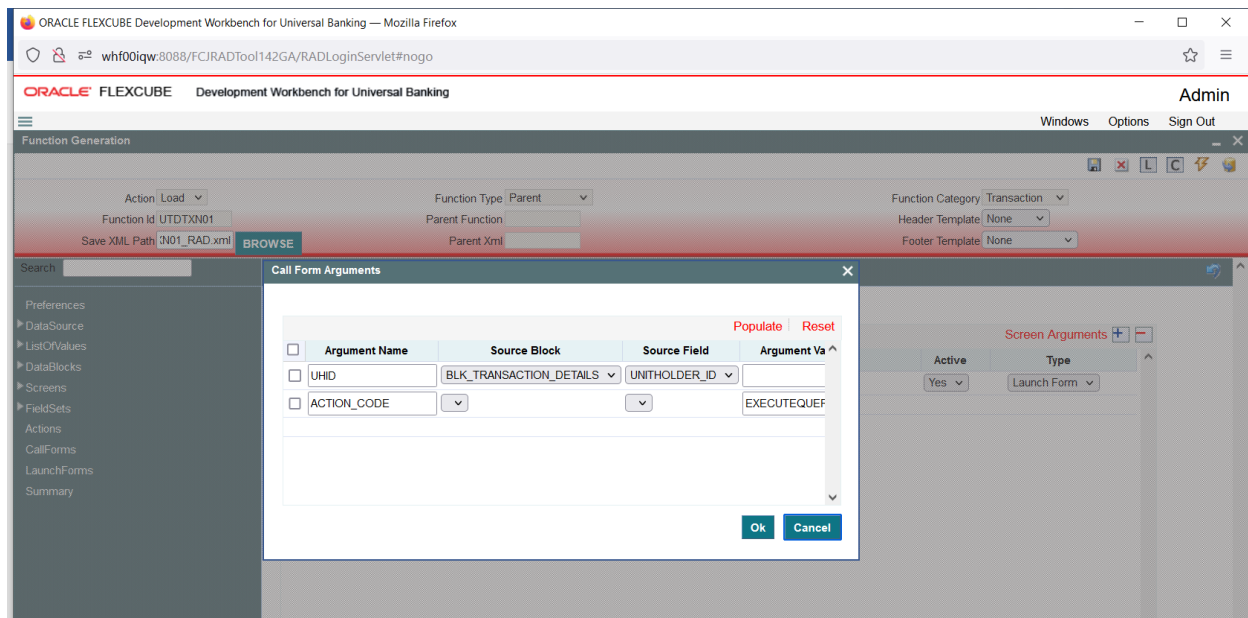
- i) Online forms will generate Type XSD and Message XSD. Operation specific message xsd's will be generated.
Example: for the example given in the figure, name of the xsd generated will be UT-FCISITransaction-Types.xsd (Type XSD for UT)
UT-QueryTransaction-Res-Full-MSG.xsd (Create Message XSD for UT)
UT-QueryTransaction-Req-IO-MSG.xsd (Create Message XSD for UT)
UT-CreateTransaction -Res-Full-MSG.xsd (Create Message XSD for UT)
UT-CreateTransaction -Res-PK-MSG.xsd (Create Message XSD for UT)
- ii) Operation Id and Operation Code need be maintained for the above mentioned reason
- iii) Amendable information has to be maintained similar to any other function ids.

4.8 Launch Forms

Launch Forms can be attached to Online form screen.



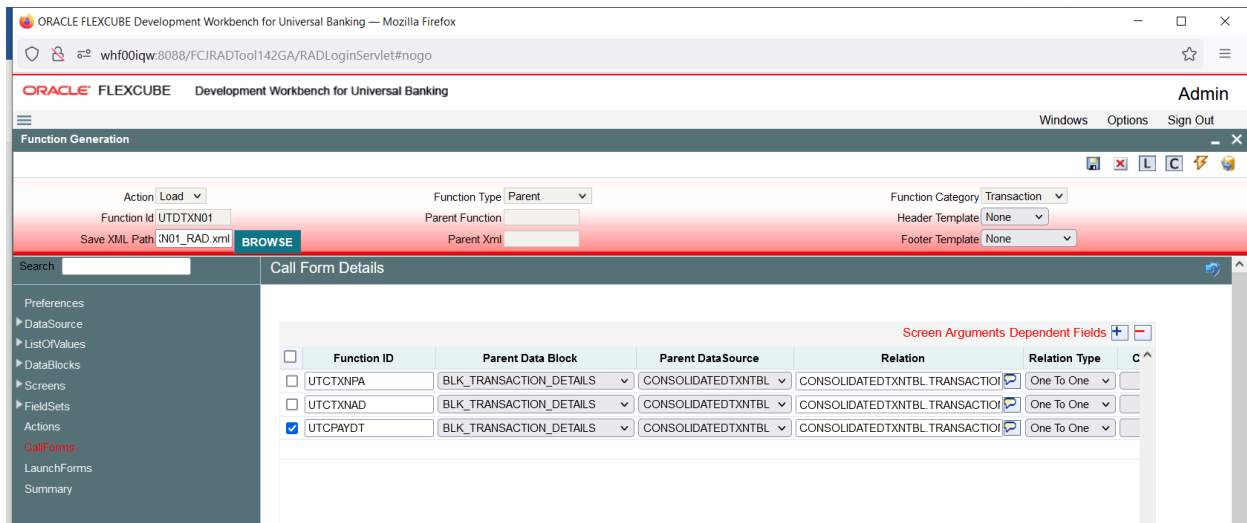
Screen Arguments should be maintained for the launch form to query the proper contract record from the main online functions.



Process to attach launch forms is similar to any other function Id's.

4.9 Call Forms

Call forms can be attached to Online form. Each call form should be mapped to Parent Data Block, Parent Data Source and proper relations should be maintained with parent data source of main online form.



Screen Arguments should be given to each callform. So that the call form will display the respective data of calling main function.

Dependent Fields are required to re default the call form values when the user changes input data in the main form.

Each of the subsystem pickup logic will have to be coded by the developer in release specific packages. Processing logic (sub system pickup) for the attached call forms has to be called from the main form package.

4.9.1 Sub System Pickup/Processing

Subsystem pickup refers to the process of picking up the values in sub systems. Normally values in sub systems will be defaulted based on the data given in the main screen of the online form.

1) Defaulting of sub system

After providing values in the main screen, user may click on any sub system to view or change the value.

On clicking the sub system for the first time, sub system values will be defaulted based on the values provided in the main screen. Action code passed will be **SUBSYS PKP**. The code for defaulting will have to be written by the developer in corresponding hook packages in function **Fn_Post_Subsys_Pickup**

In this case SUBSYSSTAT for all subsystems will go as 'D' and processing done based on this flag for each sub system (call form). Note that SUBSYS PKP action will default values for all subsystems and not only the sub system being launched

Example:

MICTRMIS:D;ISCTRSTL:D;TACTRTAX:D;CSCTRUDF:D;CFCTROCH:D;CSCTRADV:D;FTCCGCLM:D;

If user saves the contract without visiting any call forms, then all the sub systems will be defaulted before saving

2) Uploading of sub system

If after launching the subsystem with defaulted values; User changes the value in subsystem; the new user input values has to be uploaded to the system. Hence while saving, **the**

subsystems which has been modified by user will be uploaded while others will be defaulted.

In this case SUBSYSSTAT for the subsystem which has been modified will go as 'U' .Developer has to write code for processing based on the flag

Example: if user changes MIS details (MICTRMIS) from what was defaulted; then

SUBSYSSTAT will go as

*MICTRMIS:U;ISCTRSTL:D;TACTRTAX:D;CSCTRUDF:D;CFCTROCH:D;CSCTRADV:D;FTC
CGCLM:D;*

3) Re defaulting of sub system

After launching and changing subsystem values; if user changes any values in main screen which are dependent field for the subsystem: subsystem values will have to be defaulted again based on the new main screen values. Hence the sub system will be re defaulted. In this case value entered by the user in subs system will be lost.

In this case SUBSYSSTAT for the subsystem whose dependent fields has been modified will go as 'R'. .Developer has to write code for processing based on the flag

Example: In a Funds Transfer Contract Input Screen, assume that charge subsystem

(CFCTROCH) is dependent on the values entered for debit and credit account. After

launching the sub system and changing the charges manually; if user changes the account

again the charges will have to re defaulted. The manually entered charges will not be

considered. SUBSYSSTAT will go as

*MICTRMIS:U;ISCTRSTL:D;TACTRTAX:D;CSCTRUDF:D;CFCTROCH:R;CSCTRADV:D;FTC
CGCLM:D;*

Values for other subsystems will depend on each of their dependencies.

4.10 Summary

Summary screens can be designed for Online Form if required

ORACLE FLEXCUBE Development Workbench for Universal Banking — Mozilla Firefox

whf00iqw:8088/FC/RADTool142GA/RADLoginServlet#nogo

ORACLE FLEXCUBE Development Workbench for Universal Banking

Admin

Windows Options Sign Out

Function Generation

Action: Load Function Type: Parent Function Category: Transaction

Function Id: UTDTXN01 Parent Function: Header Template: None

Save XML Path: JN01_RAD.xml BROWSE Parent Xml: Footer Template: None

Search

Summary Details

Title: Data Blocks: BLK_SUMMARY Data Source: VWTXNSUMMARY Summary Type: Bulk Authorization Summary Screen Size: Medium

Default Where Clause: TRANSACTIONTYPE = '01' Default Order By: Multi Branch Where Clause: TRANSACTIONTYPE = '01' Main Summary Screen: Criteria Based Search: WebServices Required:

Data Block Fields

PAYMENTMODE

Fields Selected Query Properties

AUTHSTAT [x] Properties

RECORDSTAT [x] Properties

STATUS [] Properties

TRANSACTIONNUMBER [x] Properties

REFERENCENUMBER [x] Properties

FUNDID [x] Properties

4.11 Preview

The figure shows the preview of the Online form Input screen developed

The figure shows the preview of the Online form Summary screen developed

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

5. Generated Units

The following units will be generated for an Online Form screen.
Refer document on generated units on detailed explanation on the same.

5.1 Front End Units

5.1.1 Language xml

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

Example – **UTDTXN01.xml** (UIXML for UT Screen)

5.1.2 SYS JavaScript File

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

Example – **UTDTXN01_SYS.js** (JS for UT Screen)

5.1.3 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

Example – **UTDTXN01_KERNEL.js** (JS for KERNEL Release)

Example – **UTDTXN01_CLUSTER.js** (JS for CLUSTER Release)

Example – **UTDTXN01_CUSTOM.js** (JS for CUSTOM Release)

5.2 Database Units

5.2.1 Static Scripts

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

5.2.2 System Packages

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

Example – *utpks_utdtxn01_main.spc,*
utpks_utdtxn01_main.sql (Main Package for UT screen)

Main package contains functions for:

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

- Resolve Ref Numbers (fn_resolve_ref_numbers)
- Mandatory checks (fn_check_mandatory).
- Product Default (fn_product_default)
- Subsystem Pickup(fn_subsys_pickup)
- Enriching (fn_enrich)
- Default and validation(fn_default_and_validate)
- Uploading into DB tables(fn_upload_db)
- Processing the contract input values(fn_process)
- 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 Online forms. Thus no processing logic within the main package is used

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.

5.2.3 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.

Example – utpks_utdtxn01_kernel.spc, utpks_utdtxn01_kernel.sql (Kernel Package)
 utpks_utdtxn01_cluster.spc, utpks_utdtxn01_cluster.sql (Cluster Package)
 utpks_utdtxn01_custom.spc, utpks_utdtxn01_custom.sql (Custom Package)

5.3 Other Units

5.3.1 Xsd

Only Type XSD and message XSD will be generated for an Online Form function Id. This type xsd will be used in the type xsd of any function which uses the particular online form.

Example – UT-FCISTransaction-Types.xsd (Type XSD for UT)
 UT-CreateTransaction -Req-Full-MSG.xsd (Create Message XSD for UT)
 UT-CreateTransaction -Req-PK-MSG.xsd (Create Message XSD for UT)
 UT-CreateTransaction -Res-Full-MSG.xsd (Create Message XSD for UT)
 UT-CreateTransaction -Res-PK-MSG.xsd (Create Message XSD for UT)

6. Extensible Development

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

6.1 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 UTDTXN01_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 ()

6.2 Extensibility in Backend Coding

For online forms, generated code does not provide any business logic. Insert statements won't be present as part of generated code in online packages. Developer has to write the business logic in release specific packages (or make call I to server functions from release specific packages).

Hooks will be provided in the following stages

Resolving reference numbers

- Checking mandatory fields
- Defaulting and validating
- Uploading to db
- Process
- Subsystem pickup
- Enrich
- Product Default
- Query

Note that the system generated code for uploading; defaulting etc. (*fn_sys_default_and_validate, fn_sys_upload_db etc.*) won't be called by the main package in online flow. If it is required, developer has to call it explicitly from release specific packages.

Note that in online flow, upload to base tables happens first and processing is done on the inserted data after uploading. After processing, the response type will be build

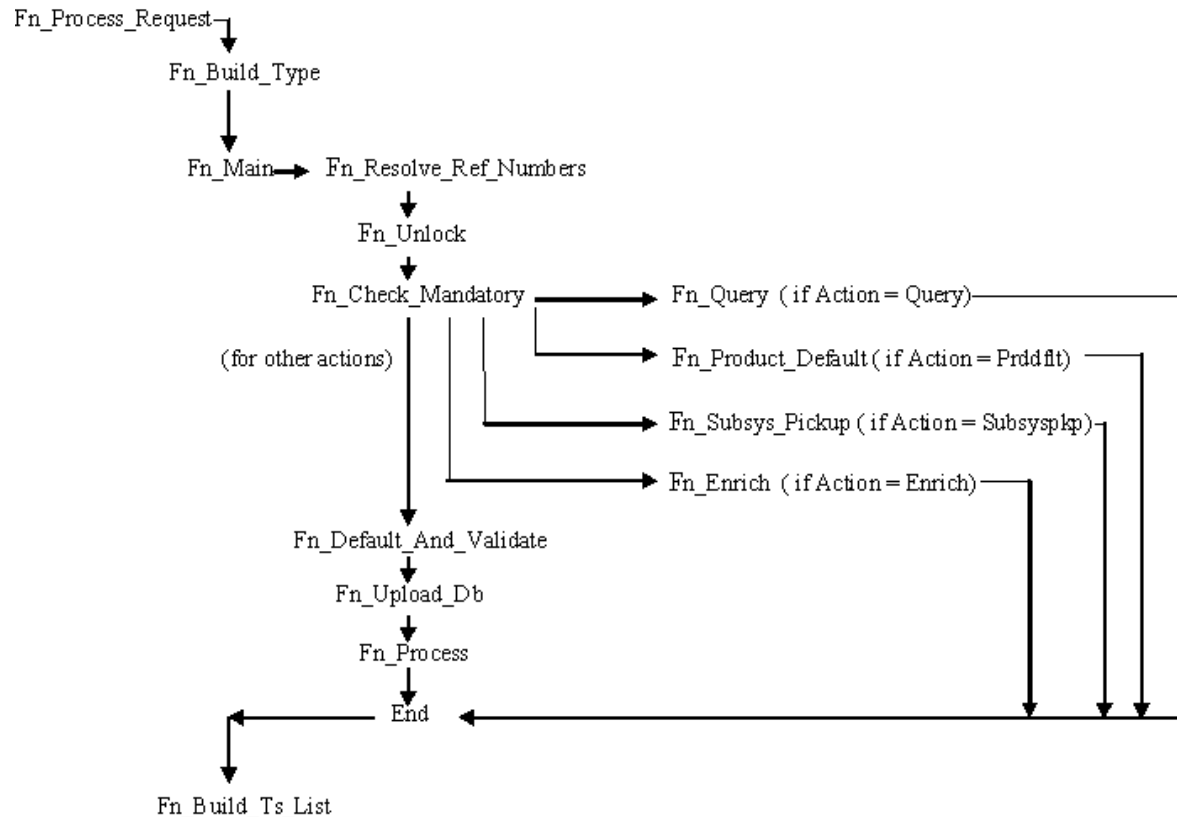


Fig Flow of control in an Online main package

Release specific code has to be written in the Hook Packages generated. Different functions available in the Hook Package of an Online 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_resolve_ref_numbers

4) Fn_post_resolve_ref_numbers

This function validates the reference number. It is called from fn_resolve_ref_numbers of the main package.

5) Fn_pre_unlock

6) Fn_post_unlock

This function holds the contract level validations and modification logic for existing contract. It is called from fn_unlock of main package.

7) Fn_pre_check_mandatory

8) Fn_post_check_mandatory

Any mandatory checks can be validated here. It is called from fn_chchk_mandatory of main package.

9) Fn_pre_query

10) Fn_post_query

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

11) Fn_pre_product_default

12) Fn_post_product_default

This function has the logic to default the values for the contract based on the product maintenance. It is called from fn_product_default of main package.

13) Fn_pre_subsys_pickup

14) Fn_post_subsys_pickup

This function does the subsystem pickup for the subsystem's (call form's) as per product maintenance for the contract. It is called from fn_subsys_pickup of main package.

15) Fn_pre_enrich

16) Fn_post_enrich

After product default, user can default others values. That logic can be put here. it is called from fn_enrich of main package.

17) Fn_pre_default_and_validate

18) 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.

19) Fn_pre_upload_db

20) Fn_post_upload_db

Any logic while uploading data to tables can be written here. It is called from fn_upload_db of main package.

21) Fn_pre_process

22) Fn_post_process

These hook functions are specific to transaction online form screens. This function should have the call to all the server functions which process the input data for the contract as per the functionality. These are called from fn_process of the main package.



Development of Online Forms
[May] [2023]
Version 14.7.0.1.0

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

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

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

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

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

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

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

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

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