

Oracle FLEXCUBE Universal Banking ® 12.0.2 Development of Online Forms

August 2013



Contents

1.	Preface	3
1.1	Audience	3
1.2	Related Documents	3
2.	Introduction.....	3
2.1	How to use this Guide	3
3.	Overview of Online Form.....	4
4.	Screen Development.....	4
4.1	Header Information	5
4.2	Preferences	5
4.3	Data Sources	6
4.4	Data Blocks.....	8
4.5	Screens	10
4.6	Field Sets.....	12
4.7	Actions	13
4.8	Launch Forms	14
4.9	Call Forms.....	15
4.9.1	Sub System Pickup/Processing	15
4.10	Summary	16
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	20
5.1.3	Release Type Specific JavaScript File	20
5.2	Data Base Units.....	20
5.2.1	Static Scripts	20
5.2.2	System Packages.....	20
5.2.3	Hook Packages.....	21
5.3	Other Units	21
5.3.1	Xsd.....	21
6.	Extensible Development	21
6.1	Extensibility in JavaScript Coding	21
6.2	Extensibility in Backend Coding.....	22

1. Preface

This document describes the features of Online Forms in FLEXCUBE and the process of designing a 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.pdf*](#)
[*05-Development_WorkBench_Screen_Development-II.pdf*](#)

2. Introduction

2.1 How to use this Guide

The information in this document includes:

- [Chapter 2 , “Introduction”](#)
- [Chapter 3 , “Overview of 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
LCDAVMNT	- 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.

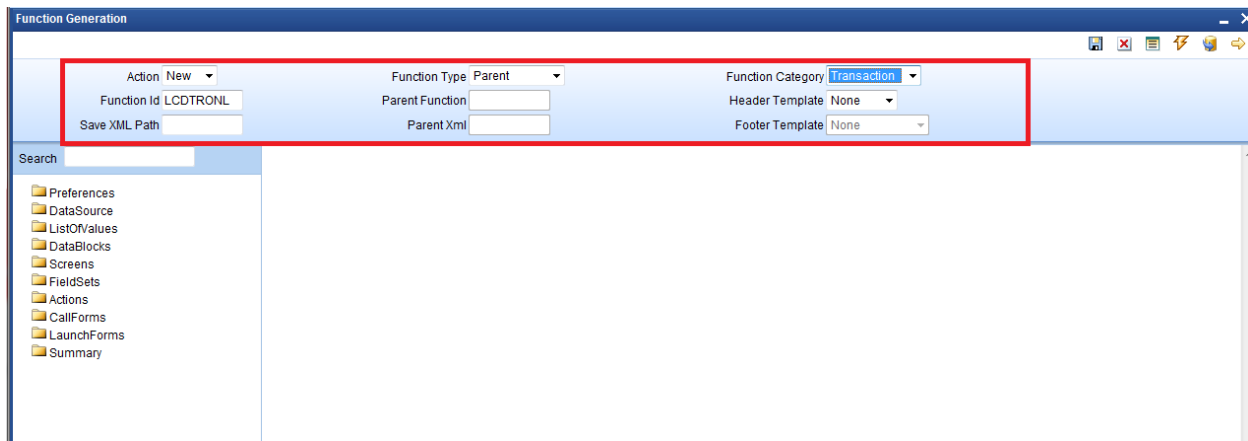
4. Screen Development

Design and development of a Online Form function id is similar to any other function Ids. This section briefs the steps in designing a 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 contains several input fields and dropdown menus. A red rectangle highlights the following fields: 'Action' (set to 'New'), 'Function Type' (set to 'Parent'), 'Function Category' (set to 'Transaction'), 'Function Id' (set to 'LCDTRONL'), 'Parent Function' (empty), 'Header Template' (set to 'None'), 'Save XML Path' (empty), 'Parent Xml' (empty), and 'Footer Template' (set to 'None'). On the left side of the window, there is a 'Search' bar and a list of folders: 'Preferences', 'DataSource', 'ListOfValues', 'DataBlocks', 'Screens', 'FieldSets', 'Actions', 'CallForms', 'LaunchForms', and 'Summary'.

Fig 4.1 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: LCDTRONL, BCDTRONL 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

Fig 4.2 Online Form Preferences

Note the following while providing Preferences for Online Forms.

- i) 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
- ii) Script for the following tables will be generated by Workbench (menu details) which are essential for launching of an Online screen.
 1. SMTB_MENU
 2. SMTB_FCC_FCJ_MAPPING
 3. SMTB_FUNCTION_DESCRIPTION
 4. SMTB_ROLE_DETAILS
 Type string of the Online screens will be generated as 'O' in *smtb_menu* table.
- iii) 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.

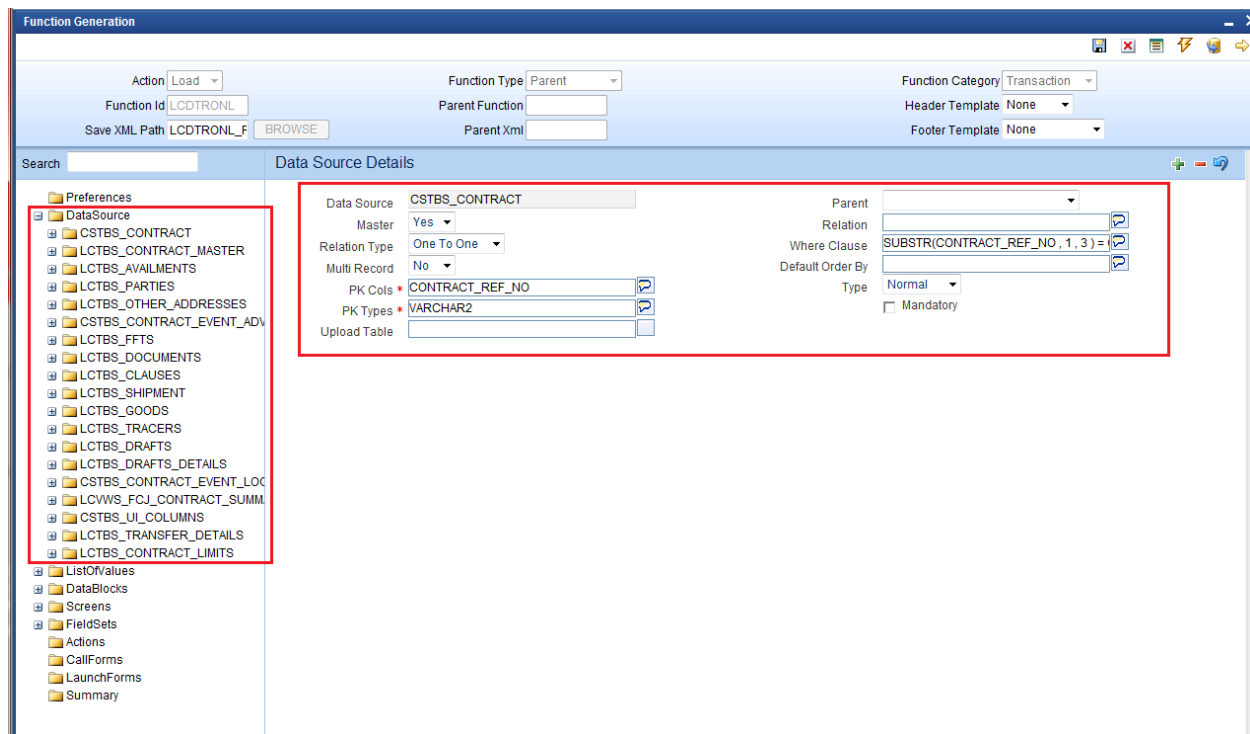


Fig 4.3 Adding data sources and maintaining properties

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: LCVWS_FCJ_CONTRACT_SUMMARY

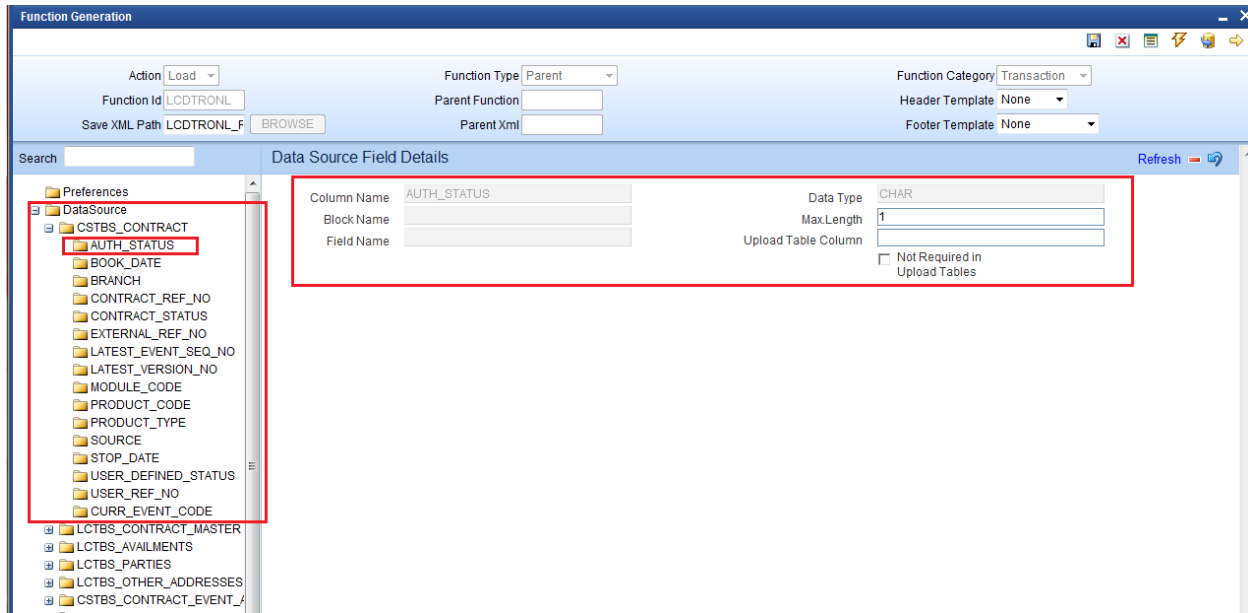


Fig 4.4 Adding data sources fields and its properties

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

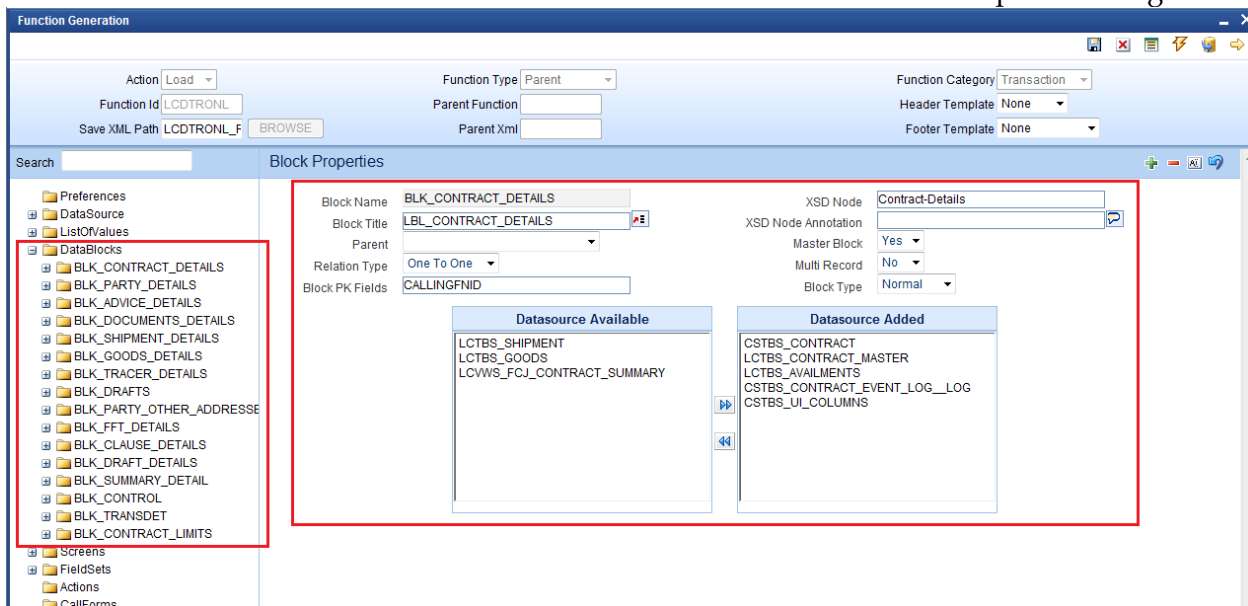


Fig 4.5 Defining Data Blocks and maintaining its properties

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 'Function Generation' window with the 'Block Field Properties' tab active. On the left, a tree view shows the hierarchy of blocks, with 'PRTYTP' selected under 'BLK_CONTRACT_DETAILS'. The main area displays the properties for 'PRTYTP'. The 'Field Name' is 'PRTYTP', 'Field Label' is 'LBL_PRTYTP', 'DataSource' is 'LCTBS_PARTIES', 'Column Name' is 'PARTY_TYPE', 'Data Type' is 'Varchar2', 'Display Type' is 'Lov', 'Item Type' is 'Database Item', 'Parent Field' is empty, 'Related Block' is empty, 'Related Field' is empty, 'LOV Name' is 'LOV_PART_TYPE', 'Off Line LOV Name' is empty, 'Fieldset Name' is 'FST_CONTRACT_PARTIES', 'XSD Tag' is 'PRTYTP', 'XSD Annotation' is empty, 'Field Size' is '3', 'Maximum Length' is '3', 'Minimum Value' is empty, 'Maximum Value' is empty, 'Maximum Decimals' is empty, 'Text Area Rows' is empty, 'Text Area Columns' is empty, 'Default Value' is empty, 'Preview Value' is empty, 'Mask Id' is empty. Checkboxes on the right indicate 'Required' and 'Visible' are checked, while others are unchecked. Below the properties, the 'Return Fields Mapping' table shows two mappings: 'PARTY_TYPE' from 'BLK_PARTY_DETAILS' to 'PRTYTP' and 'ITEM_VAL_DESC' from 'BLK_PARTY_DETAILS' to 'PARTYDESC'.

Fig 4.6 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 a Online Form screen; as launch form can be used for query only screens
- iv) Master block should contain reserved field names like TXNSTAT, AUTHSTAT and SUBSYSSTAT (this is not shown) as shown in the figure. These are reserved field names which are essential for an online form. These will be used by FLEXCUBE Infra while processing. Normally TXNSTAT and AUTHSTAT are added as part of the footer of the screen

COLUMN NAME	BLOCK FIELD NAME
CONTRACT_STATUS	TXNSTAT
AUTH_STATUS	AUTHSTAT
SUBSYSTEM_STAT	SUBSYSSTAT

Function Generation

Action: Load Function Id: LCDTRONL Save XML Path: LCDTRONL_F BROWSE

Function Type: Parent Parent Function: Parent Xml:

Function Category: Transaction Header Template: None Footer Template: None

Search: Block Field Properties

Field Name: CONSTAT Field Label: LBL_CONSTAT DataSource: CSTBS_CONTRACT Column Name: CONTRACT_STATUS Data Type: Char Display Type: Text Item Type: Database Item Parent Field: Related Block: Related Field: LOV Name: Off Line LOV Name: Fieldset Name: FST_PRODDDET

XSD Tag: CONSTAT XSD Annotation: Field Size: 1 Maximum Length: 1 Minimum Value: Maximum Value: Maximum Decimals: TextArea Rows: TextArea Columns: Default Value: Preview Value: Mask Id:

Required Visible Read Only Calendar Text Popup Edit Required Uppercase Only LOV Validation Required Input by LOV Only Not Required In Xsd Report Parameter

Custom Attributes Events Related Field

Attribute Name	Attribute Value	Active	Position
----------------	-----------------	--------	----------

4.5 Screens

Design the screen layout based on the requirement

Function Generation

Action: Load Function Id: LCDTRONL Save XML Path: LCDTRONL_F BROWSE

Function Type: Parent Parent Function: Parent Xml:

Function Category: Transaction Header Template: None Footer Template: None

Search: Screen Details

Screen Name: CVS_MAIN Screen Title: LBL_CONTRACT_DETAILS Screen Size: Large Exit Button Type: Default Cancel

Main Screen Visible

Argument Name	Source Block	Source Field	Argument Value	Target Block	Target Field	Active
CONTREF				BLK_CONTRACT_DETAILS	CONREFNO	Yes
ESN				BLK_CONTRACT_DETAILS	LATEVNSEQNO	Yes

Fig 4.7 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

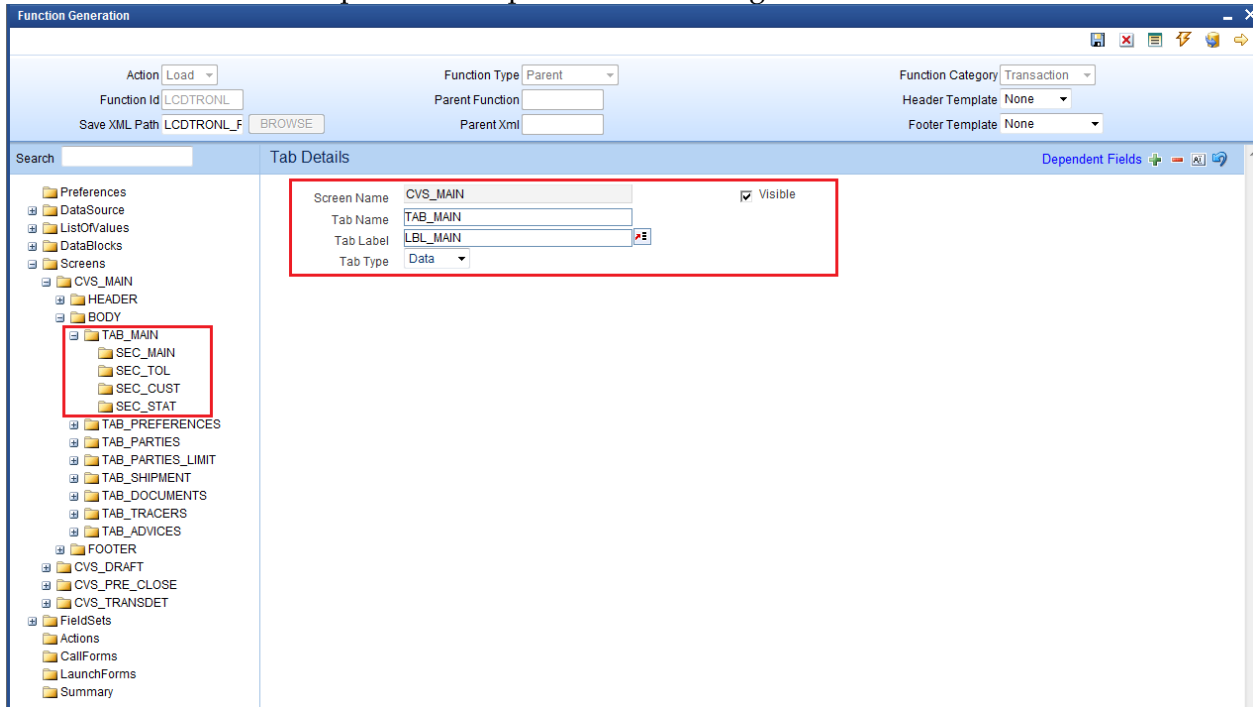


Fig 4.8 Creating Tabs and maintaining Properties

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*

Function Generation

Action: Load | Function Id: LCDTRONL | Save XML Path: LCDTRONL_F | Function Type: Parent | Parent Function: | Parent Xml: | Function Category: Transaction | Header Template: None | Footer Template: None

Search: | Section Details

Section Name: SEC_TOL | Section Label: | Visible: ☒ | Collapse: ☐

Partition SI No	Partition Name	Width	Sub-partitions
1	PART_TOL1	66	2
2	PART_TOL2	33	

Fig 4.8 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

Function Generation

Action: Load | Function Id: LCDTRONL | Save XML Path: LCDTRONL_F | Function Type: Parent | Parent Function: | Parent Xml: | Function Category: Transaction | Header Template: None | Footer Template: None

Search: | Fieldset Properties

Fieldset Name: FST_PROD | Fieldset Label: | Data Block: BLK_CONTRACT_DETAILS | Multi Record: No | View Type: Single | Fieldset Height: | Screen Name: CVS_MAIN | Screen Portion: Header | Tab Name: TAB_HEADER | Section Name: SEC_HEADER2 | Partition Name: PART1 | Number Of Rows: | Horizontal Fieldset: ☒ | ReadOnly: ☐ | Navigation Button: ☐ | Visible: ☒

Data Block Fields	FieldSet Fields	Subpartition Name
BTN_DRAFTS	<input type="checkbox"/> PRDCD	
BTN_COMMISSION	<input type="checkbox"/> BTN_P	
PRITYTYP	<input checked="" type="checkbox"/> CALLINGFID	
CRDLIN		
LINECID		
INCAMDNO		
EVENTCD		
BTN_CHARGES		
BTN_SETT		
BTN_TAX		
BTN_COLLATERAL		
BTN_EVENTS		
BTN_LINKAGES		

Fig 4.9 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

Web Service	Action Code	Operation Code	Action Stage Type	Amendables
<input checked="" type="checkbox"/>	QUERY	QueryContract	<input checked="" type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	NEW	CreateContract	<input checked="" type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	MODIFY	ModifyContract	<input checked="" type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	AUTHORIZE	AuthorizeContract	<input type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	DELETE	DeleteContract	<input type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	CLOSE	CloseContract	<input type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	REOPEN	ReopenContract	<input checked="" type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	REVERSE	ReverseContract	<input type="checkbox"/>	Amendables
<input checked="" type="checkbox"/>	ROLLOVER	RolloverContract	<input checked="" type="checkbox"/>	Amendables
<input type="checkbox"/>	CONFIRM		<input type="checkbox"/>	Amendables
<input type="checkbox"/>	LIQUIDATE		<input type="checkbox"/>	Amendables
<input type="checkbox"/>	SUMMARYQUERY		<input checked="" type="checkbox"/>	Amendables

Fig 4.10 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

LC-Contract-Types.xsd (Type XSD for LC Contract)

LC-CreateContract-Req-Full-MSG.xsd (Create Message XSD for LC Contract)

LC-CreateContract-Req-IO-MSG.xsd (Create Message XSD for LC Contract)

LC-CreateContract-Res-Full-MSG.xsd (Create Message XSD for LC Contract)

LC-CreateContract-Res-PK-MSG.xsd (Create Message XSD for LC Contract)

- 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.

The screenshot shows the 'Function Generation' window. On the left is a tree view with categories like Preferences, DataSource, ListOfValues, DataBlocks, Screens, FieldSets, Actions, CallForms, LaunchForms (highlighted with a red box), and Summary. The main area is titled 'Launch Form Details'. It contains a table with columns 'Function ID' and 'Active'. The table lists three items: CSDEVENT, MSDALMSG, and MSDMSPRV (checked). A red box highlights the table and a 'Screen Arguments' button in the top right corner of the table area.

Function ID	Active
CSDEVENT	Yes
MSDALMSG	Yes
MSDMSPRV	Yes

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

This screenshot shows the same 'Function Generation' window, but with the 'Call Form Arguments' dialog box open. The dialog has a 'Populate' button and a 'Reset' button. It contains a table with columns 'Argument Name', 'Source Block', 'Source Field', and 'Argument Value'. The table lists two arguments: 'CONTRF' with source 'BLK_CONTRACT_DETAILS' and field 'CONREFNO', and 'ACTION_CODE' with value 'EXECUTEQUERY'. A red box highlights the dialog box.

Argument Name	Source Block	Source Field	Argument Value
CONTRF	BLK_CONTRACT_DETAILS	CONREFNO	
ACTION_CODE			EXECUTEQUERY

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.

The screenshot shows the 'Function Generation' window with the 'Call Form Details' tab selected. The window includes a search bar, a left sidebar with a tree view (where 'CallForms' is highlighted), and a main table. The table has columns for 'Function ID', 'Parent Data Block', 'Parent Data Source', 'Relation', and 'Dependent Fields'. The 'Dependent Fields' column is expanded to show 'Relation Type' for each row. The 'Function ID' column has a checkbox for each row, with 'LCCILUTL' checked. The 'Parent Data Source' column has a dropdown arrow for each row. The 'Relation' column has a dropdown arrow for each row. The 'Dependent Fields' column has a dropdown arrow for each row.

Function ID	Parent Data Block	Parent Data Source	Relation	Dependent Fields
<input type="checkbox"/> CFCTRCOM	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> CFCTRCHG	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> ISCTRSTL	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> LCCTRCLT	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> TACTRTAX	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> CSCTRLNK	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> CSCTRUDF	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To Many
<input type="checkbox"/> MICTRMIS	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> LCCBCLNK	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> CSCTRSPT	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> BCCTRPRF	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> CSCOFACF	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> BCCBRDET	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input type="checkbox"/> CSCDOCTR	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One
<input checked="" type="checkbox"/> LCCILUTL	BLK_CONTRACT_DETAILS	LCTBS_CONTRACT_MASTER	LCTBS_CONTRACT_MASTER.CONT	One To One

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

Dependant 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 .

i) 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 **SUBSYSPKP** .

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 SUBSYSPKP action will default values for all subsystems and not only the sub system being launched

Example:

MICTRMIS:D;ISCTRSTL:D;TACTRTAX:D;CSCTRUDE: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

ii) 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;CSCTRUDE:D;CFCTROCH:D;CSCTRADV:D;FTCCGCLM:D;

iii) 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 sub 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;CSCTRUDE:D;CFCTROCH:R;CSCTRADV:D;FTCCGCLM: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

Function Generation

Action

Load

Function Id

LCDTRONL

Save XML Path

LCDTRONL_F

BROWSE

Function Type

Parent

Parent Function

Parent Xml

Function Category

Transaction

Header Template

None

Footer Template

Search

Summary Details

Title

LBL_SUMMARY

Data Blocks

BLK_SUMMARY_DETAIL

Data Source

LCVWS_FCJ_CONTRACT_SUM

Summary Type

Summary

Summary Screen Size

Medium

Default Where Clause

MODULE_CODE='LC'

Default Order By

Multi Branch Where Clause

MODULE_CODE = 'LC' AND EXISTS (SEL

Main Summary Screen

WebServices Required

☐

Data Block Fields

Custom Buttons

Fields Ordering

Data Block Fields

CLOSDT
SETLMTH
EFFDT
CIFID
MAXCONTAMT
MAXLIABAMT
CURRAVAIL
OSLIAB

Fields Selected

☐ AUTHSTAT
☐ CONSTAT
☐ CONREFNO
☐ PRDCD
☐ CONTCCY
☐ CONTAMT
☐ USEREFNO
☐ EXTREFNO

Query

☒
☒
☒
☒
☒
☒
☐
☐

LOV Name

4.11 Preview

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

Product Details

Product Code * P Contract Reference Operation Code *
 Product Description User Reference Source Code FLEXCUBE
 Product Type Source Reference Version Number Of

LC Details

Currency * Customer * Issue Date
 Contract Amount * Customer Name Effective Date
 Positive Tolerance Party Type * Tenor
 Negative Tolerance Dated Expiry Date
 Max Amount Customer Reference Expiry Place
 Liability Tolerance License Expiry Date ☐ Auto Closure
 Liability Amount Amount Closure Date
 Tolerance Text Liability Stop Date
 Pre-Advice Date
 Reference To Pre-advice

Credit

Type Credit Available With
 Mode Details

Guarantee Details

Type of guarantee
 Guarantee

Revolving Detail

Revolves in ☐ Automatic Reinstatement
 Units ☐ Cumulative
 Frequency Next Reinstatement Date
 Remarks

Reimbursement Undertaking

Undertaking Expiry Date Availed Undertaking Amount
 Undertaking Amount

Status

Navigation Bar: Drafts | Commission | Charges | Settlement | Tax | Collateral | Events | Linkage Details | Fields | MIS | Transfer Details | BC Linkages | Split Settlement | Loan Preference | Brokerage | All Messages | Documents | Message Preview | Import License

Footer: Maker Date Time Status
 Checker Date Time Authorization Status
 Exit

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

Fig 4.10 Online Form Summary Screen Preview

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 a 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 – LCDTRONL.xml (uixml for LC Contract 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 – LCDTRONL_SYS.js (JS for LC Contract 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 – LCDTRONL_KERNEL.js (JS for KERNEL Release)

Example – LCDTRONL_CLUSTER.js (JS for CLUSTER Release)

Example – LCDTRONL_CUSTOM.js (JS for CUSTOM Release)

5.2 Data Base Units

5.2.1 Static Scripts

The following static scripts generated are required for the proper functioning of a 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 – Lcpks_Lcdtronl_Main.spc,

Lcpks_Lcdtronl_Main.sql (Main Package for LC Contract)

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 - Lcpks_Lcdtronl_Kernel.spc, Lcpks_Lcdtronl_Kernel.sql (Kernel Package)
Lcpks_Lcdtronl_Cluster.spc, Lcpks_Lcdtronl_Cluster.sql (Cluster Package)
Lcpks_Lcdtronl_Custom.spc, Lcpks_Lcdtronl_Custom.sql (Custom Package)

5.3 Other Units

5.3.1 Xsd

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

Example - LC-Contract-Types.xsd (Type XSD for LC Contract)
LC-CreateContract-Req-Full-MSG.xsd (Create Message XSD for LC Contract)
LC-CreateContract-Req-IO-MSG.xsd (Create Message XSD for LC Contract)
LC-CreateContract-Res-Full-MSG.xsd (Create Message XSD for LC Contract)
LC-CreateContract-Res-PK-MSG.xsd (Create Message XSD for LC Contract)

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 LCDTRONL_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 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

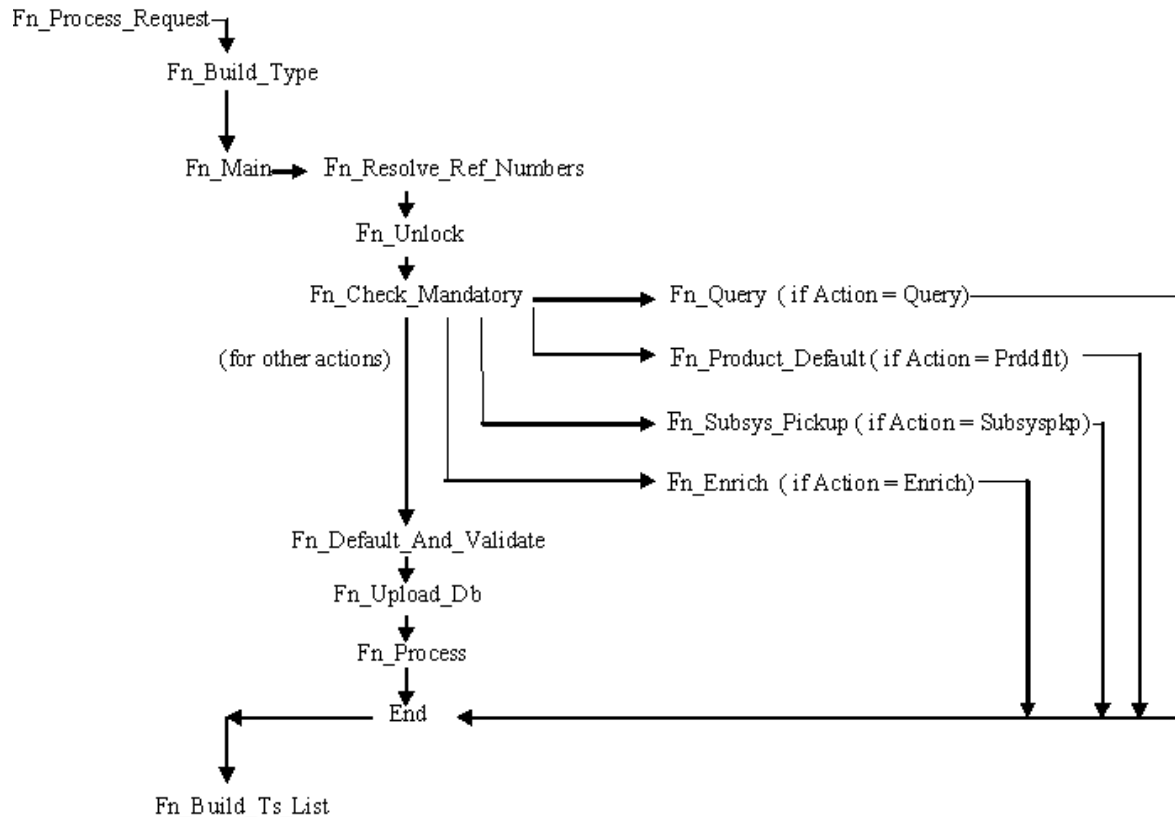


Fig 4.10 Flow of control in an Online main package

Release specific code has to be written in the Hook Packages generated. Different functions available in the Hook Package of a 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 form 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
August 2013

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
[www.oracle.com/ financial_services/](http://www.oracle.com/financial_services/)

Copyright © 2012-2013 Oracle Financial Services Software Limited. All rights reserved.

No part of this work may be reproduced, stored in a retrieval system, adopted or transmitted in any form or by any means, electronic, mechanical, photographic, graphic, optic recording or otherwise, translated in any language or computer language, without the prior written permission of Oracle Financial Services Software Limited.

Due care has been taken to make this document *Development of Online Forms* and accompanying software package as accurate as possible. However, Oracle Financial Services Software Limited makes no representation or warranties with respect to the contents hereof and shall not be responsible for any loss or damage caused to the user by the direct or indirect use of this *Development Of Online Forms* and the accompanying Software System. Furthermore, Oracle Financial Services Software Limited reserves the right to alter, modify or otherwise change in any manner the content hereof, without obligation of Oracle Financial Services Software Limited to notify any person of such revision or changes.

All company and product names are trademarks of the respective companies with which they are associated.