Development of Call Form Oracle FLEXCUBE Universal Banking Release 14.5.5.0.0 Part No. F56689-01 [May] [2022]



# Contents

1. Prefa	ace	3
1.1	Audience	3
1.2	Related Documents	3
2. Intro	oduction	4
2.1	How to use this Guide	4
3. Over	rview of Call Form	4
4. Scree	en Development	4
4.1	Header Information	5
4.2	Preferences	5
4.3	Data Sources	6
4.4	Data Blocks	7
4.5	Screens	9
4.6	Field Sets	
4.7	Actions	
4.8	Launch Forms	
4.9	Call Forms	
4.10	Summary	
4.11	Preview	
	ching Call Form to Main Function Id	
	erated Units	
6.1	Front End Units	
6.1.1	Language xml	15
6.1.2	SYS JavaScript File	15
6.1.3	Release Type Specific JavaScript File	15
6.2	Data Base Units	15
6.2.1		
6.2.2	System Packages	15
6.2.3	Hook Packages	16
6.3	Other Units	16
6.3.1		
	nsible Development	
7.1	Extensibility in JavaScript Coding	
7.2	Extensibility in Backend Coding	17

# 1. Preface

This document describes the features of a Call Form screen in FLEXCUBE and the process of designing a Call 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:

Proficiency	Resources
FLEXCUBE Functional Architecture	Training programs from Oracle
	Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle
	Financial Software Services.
FLEXCUBE Screen Development	04-Development_WorkBench
	_Screen_Development-I.docx
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

<u>04-Development\_WorkBench\_Screen\_Development-I.docx</u> <u>05-Development\_WorkBench\_Screen\_Development-II.docx</u> <u>14-Development\_of\_Online\_Forms.docx</u>

# 2. Introduction

### 2.1 How to use this Guide

The information in this document includes:

- Chapter 2 , "Introduction"
- Chapter 3, "Overview of Call Form"
- <u>Chapter 4</u>, "Screen Development"
- <u>Chapter 5 , "Generated Units"</u>
- <u>Chapter 5</u>, "Extensible Development"

### 3. Overview of 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

- 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

### 4. Screen Development

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

Action Load	Function Type Parent	📰 🗷 🗐 7 🇐 Function Category Transaction 👻
Function Id CFCTRCHG	Parent Function	Header Template None
Save XML Path CFCTRCHG_I BROWSE	Parent Xml	Footer Template
ch c		
Preferences		
DataSource		
in ListOfValues DataBlocks		
Screens		
FieldSets		
Actions		
CallForms		
LaunchForms Summary		
Gummary		

Fig 4.1 Call Form header Information

Note the following while providing header information.

i) 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: CFCTRCHG, ISCTRSTL etc are valid call form names

- ii) Call Form Category: It has to be either Maintenance or Transaction depending on the functionality and the screens from which it will be invoked
- iii) Footer Template:
   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
- iv) Function Type : Parent and child functionality is supported for call forms

### 4.2 Preferences

Provide the menu details in the Preferences screen

Function Generation									>
					F	× =	¥	9	⇔
Action Load 👻	Function Typ			Tuncaun Gategory Transaction	-				
Function Id CFCTRCHG	Parent Function			Header Template None 🔻					
Save XML Path CFCTRCHG_F	BROWSE Parent Xr	iml		Footer Template	-				
Search	Preferences							9	ŕ
Preferences DataSource DataSource DataBlocks DataBlocks DataBlocks DataBlocks Califorms Califorms Summary	☐ Head Office Fun ☐ Logging Requin ☐ Auto Authorizatic ☐ Tank Modificatio ☐ Field Log Requi ☐ Multi Branch Ace ☐ Excel Export Rec	red on ons irred cess	Module Module Description Branch Program Id Process Code SVN Repository URL Transaction Block Name Transaction Field Name	The ICCF Choose Block	Cont	rol String	+	-	
	Function Id	Module *		Module Description				<b>~</b>	
	CFCTRCHG	CF	<b>1</b>	The ICCF				Ŧ	

Fig 4.2 Call Form Preferences

Note the following while providing Preferences for Call 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)* Of the menu details inc generated, only script for SMTB\_MENU and SMTB\_FCC\_FCJ\_MAPPING is required for Call Forms

# iii) 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

### 4.3 Data Sources

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

Action Load  Function Id CFCTRCHG Save XML Path CFCTRCHG   BROWSE	Function Type Parent		Function Category Transaction	1		
Search Data Source Det	Parent Xml		Header Template None 👻	•	4	 Lý)
Preferences     DataSource     DataSource     DataSource     DataSource     DataSource     DataSource     DataSource     Maste     Relation Type     CFTBS_CHARGE_ASSOC     DataBlocks     Suprens     CallForms     CallForms     Summary	CSTBS_CONTRACT_CHG Yes One To One No CONTRACT_REF_NO-LATEST_EVENT_ VARCHAR2-NUMBER	1,00	Normal V Mandatory			

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

			<b>.</b>	× =	V 🧃	⇒
Action Load  Function Id CFCTRCHG Save XML Path CFCTRCHG_F	Function Type Parent Function BROWSE Parent Xml		Function Category Transaction   Header Template None   Footer Template			
Search  Preferences  DataSource CONTRACT_REF_NO CONTRACT_REF_NO CONTRACT_REF_NO CONTRACT_REF_NO CONTRACT_REF_NO CONTRACT_REF_NO CONTRACT_PLN CTBS_CHARGE_ASSOC CONTRACT_CARGE_LIQD_MASTEI CONTRACT_CARGE_CARGE_LIQD_MASTEI CONTRACT_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_CARGE_	Column Name Block Name Field Name	Data Type MaxLength Upload Table Column	VARCHAR2	Refr	esh — G	2

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

Function Ger

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

×

Function Generation								_ >
					 ×	= 1	۶ 🧃	⇔
	BROWSE	Function Type Parent		Function Category Header Template N Footer Template				
Search  Preferences  DataSource DataSource DataBlocks  DataBlocks	Block Properties Block Name Block Title Parent Relation Type Block PK Fields	BLK_CHARGES	≥∎ ble €			* -		

Fig 4.5 Defining Data Blocks and maintaining its properties

Note the following while creating data blocks

- i) Master Data Source has to be a single entry data source.
- ii) Logical Relationships with the parent has to be maintained for all data sources.
- 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) Provide Xsd node name if the block is normal and is required in gateway request

Add block fields to the data block as required.

Function Generation				_ >
				🔚 🗷 🗏 🌮 🍕 🔿
Action Load  Function Id CFCTRCHG	Function Type Par Parent Function	rent -	Function Category Transaction Header Template None -	
Save XML Path CFCTRCHG_F	BROWSE Parent Xml		Footer Template	-
Search	Block Field Properties			- 🛚 🗔 🦃 🕯
Preferences DataSource ListOfValues DataBlocks DataBlocks DLCHARGES CONREFNO D. ELK_CHARGE_ASSOCIATION DLK_CHARGE_ASSOCIATION DLK_CHARGE_LIQUIDATION DLK_CHARGE_S DLK_CHARGE_LIQUIDATION DLK_CHARGE_S DLK_CHARGE_	Field Name + LATEVINSEQNO Field Label LBL_LATEVINSEQNO DataSource CSTBS_CONTRACT_CH Column Name + LATEST_EVENT_SEQ_NO Data Type + Number ~ Display Type + Database Item ~ Parent Field ~ Related Field LOV Name Fieldset Name FLD_FCCREF Custom Attributes Events Related Field	TICIG OILC	ATEVNSEONO	Required  Visible  Read Only  Calender Text  Popup Edit Required  Uppercase Only  LOV Validation  Required  Input by LOV Only  Not Required In Xsd  Report Parameter
	Attribute Name	Attribute Value	Active Positio	

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 Call Form screen; as launch form can be used for query only screens

#### 4.5 Screens

Design the screen layout based on the requirement

Action Load  Function Id CFCTRCHG Save XML Path CFCTRCHG	BROWSE	Function Type Parent Parent Function Parent Xml	<b>•</b>		Function Cates Header Temp Footer Temp			
arch Preferences DataSource DataSource DataStorks DataBlocks Screens CVS_CFCTRCHG DHADER	Screen Details Screen Name Screen Title Screen Size Exit Button Type	CVS_CFCTRCHG LBL_CHARGE_DETAILS Medium Default Ok Cancel	7E 	I⊽ Main Screen I⊽ Visible			- 4	
🗉 🚞 BODY							-	
	Argument	Name Source Block	Source Field	Argument Value	Target Block	Target Field	Active	^
CallForms	CONTREF	<b>_</b>	-		BLK_CHARGES -	CONREFNO -	Yes 👻	_
🞴 LaunchForms 🔁 Summary	ESN ESN				BLK_CHARGES V	LATEVNSEQNO V	Yes •	

Fig 4.7 Designing Screens and providing Screen Properties

Note the following while creating screens

- i) One Screen should be identified as the main screen; if multiple screens present
- ii) In the function id ,where the call form is called :

For the button (which launches call form) events, the main screen of the call form has to be mentioned

#### iii) Screen Arguments :

Screen Arguments has to be provided 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.

Add Tabs, sections and partitions as per the screen design

Function Generation								. ×
				8	×	1	4	⇔
Action Load  Function Id CFCTRCHG Save XML Path CFCTRCHG		Function Type Parent Parent Function Parent Xml	ŀ	unction Category Transaction   Header Template None   Footer Template				
Search	Tab Details			Dependent F	ields 🚽	• (	x 9	_1
Preferences DataSource DataSource DataSlocks CVS_CFCTRCHG SEC_APPL SEC_APPL SEC_APPL SEC_APPL SEC_APPL SEC_APPL SEC_APPL SEC_LIDD SI = FredSets Actions CullForms LaunchForms Summary	Screen Name Tab Name Tab Label Tab Type	CVS_CFCTRCHG TAB_MAIN LEL_ALL Data V	visible ≥					

Fig 4.8 Creating Tabs and maintaining Properties

Note the following when creating tabs and sections for the screen

i) 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

Function Generation					-
				II 🗶 🛛	I 7 🧃 ╡
Action Load  Function Id CFCTRCHG Save XML Path CFCTRCHG_F	BROWSE	Function Type Parent  Parent Function Parent Xml		Function Category Transaction   Header Template None   Footer Template	
Search	Section Details				- 🛛 🌍
<ul> <li>Preferences</li> <li>DataSource</li> <li>ListOfValues</li> <li>DataBlocks</li> <li>Screens</li> <li>CVS_CFCTRCHG</li> </ul>	Section Name SEC Section Label	MAIN	♥ Visible		+-
🗉 🚞 HEADER	Partition Details		Partition Name	Width Sub-partitions	±=
<ul> <li>BODY</li> <li>TAB_MAN</li> <li>SEC_ASOC</li> <li>SEC_ASOC</li> <li>SEC_LIOD</li> <li>FOOTER</li> <li>FieldSets</li> <li>Actions</li> <li>CallForms</li> <li>LaunchForms</li> <li>Summary</li> </ul>		PARL_MAIN_1 PARL_MAIN_2		View         Sup-partuoins           50 ▼         ▼           33 ▼         0 ▼	*

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

unction Generation				
				🖫 🗶 🗏 🖗 🗐
Action Load 👻	Function Type Parent	Func	tion Category Transaction	<b>*</b>
Function Id CFCTRCHG	Parent Function	Hea	der Template None 👻	
Save XML Path CFCTRCHG_F	ROWSE Parent Xml	Foo	ter Template	•
earch F	Fieldset Properties			- 🛛 🗳
Preferences	Fieldset Name FLD_FCCREF	Screen Name CVS_CFCTRCHG	•	🗖 Horizontal Fieldset
DataSource     DistofValues	Fieldset Label	Screen Portion Body	<b>~</b>	ReadOnly
DataBlocks	Data Block BLK_CHARGES -	Tab Name TAB_MAIN	<b>~</b>	Navigation Button
Screens	Multi Record No 👻	Section Name SEC_MAIN	-	Visible
CVS_CFCTRCHG	View Type Single 👻	Partition Name PARt_MAIN_1	-	
	Fieldset Height	Number Of Rows		
DODY     TAB_MAIN				
SEC_ASSOC	Data Block Fields	Field Set Fields	Subpartition Name	
Dec_APPL			- -	
SEC_LIQD				
FOOTER  FIEldSets		CONREFNO		
		ÞÞ		_
FST_CHARGE_ASSOCIATION				
ST_CHARGE_APPLICATION		44		
E FST_CHARGE_LIQD_MASTER				
CallForms				
a Californis				
Summary				

Fig 4.9 Field Set Properties

Not the following when attaching field to a field set

i) 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

### 4.7 Actions

Mention the web service and amendable information in Actions Screen

Action Load -						🖁 🗶 🗏			
Evention Information		Function Type Parer	nt 👻	Function Category Transaction					
Function Id CFCTRCHG		Parent Function		Header Template None 👻					
Save XML Path CFCTRCHG_F	BROWSE	Parent Xml			Footer Template	•			
h	Form Actions						1		
Preferences	XSD Type Ider	ntifier TxnChgDtls		Service Na	me CFCTRCHG	75			
DataSource ListOfValues DataBlocks Screens	Operation Id								
FieldSets							+-		
CallForms	Web Service	Action Code	Ope	eration Code	Action Stage Type	Amendables Amendables	^		
눹 LaunchForms 🔁 Summary		NEW				Amendables			
		MODIFY				Amendables			
		AUTHORIZE				Amendables			
		DELETE				Amendables			
		CLOSE				Amendables			
		REOPEN			Amendables				
		REVERSE			Amendables				
		ROLLOVER				Amendables			
		CONFIRM				Amendables			
						Amendables			
		SUMMARYQUERY				Amendables			

Fig 4.10 Actions Screen

Note the following while maintaining web services and amendable information

i) 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: for the example given in the figure, name of the xsd generated will be SubSys-TxnChgDtls-Types.xsd* 

- ii) Operation Id and Operation Code need not be maintained for the above mentioned reason
- iii) Amendable information has to be maintained similar to any other function id's.

### 4.8 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.

### 4.9 Call 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

# 4.10 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

### 4.11 Preview

The figure shows the preview of the cal form screen developed

🔶 Charge Details									
Contract Reference *									
Charge Association									
I ≪ 1 of 1 🕨 🕨	Go to Page					[	==		
Creation ESN	Compone	nt * Rule		Description		Consider as Discour	*		
							-		
•									
Charge Application									
I≪ 1 of 1 ▶ ▶I	Go to Page								
Creation ESN	Component *	Tag Currency	ncy Tag Amount		arge Currency	Charge Amoun	*		
							-		
•	۲.								
Charge Liquidation									
I	Go to Page						==		
Event Sequence Number	Component	Charge Curre	ncy Charge An	nount	Liquidated		*		
					$\checkmark$				
							-		
•						•			
						Ok	Exit		

Fig 4.10 Call Form Screen Preview

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

# 5. Attaching 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 04-Development\_WorkBench \_Screen\_Development-I.docx for detailed explanation Note that 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.

# 6. Generated Units

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

### 6.1 Front End Units

#### 6.1.1 Language xml

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

### 6.1.2 SYS JavaScript File

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

### 6.1.3 Release Type Specific JavaScript File

This file won't be generated by the Tool. It has to be manually written by the developer if he has to write any code specific in that release

### 6.2 Data Base Units

### 6.2.1 Static Scripts

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

i) Menu Details

Scripts for SMTB\_MENU and SMTB\_FCC\_FCJ\_MAPPING 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
- iii) Lov Details
- iv) Amendable Details
- v) Label details
- vi) Screen Details
- vii) Block details
- viii) Data Source Details

### 6.2.2 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

### 6.2.3 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.

## 6.3 Other Units

### 6.3.1 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

# 7. Extensible Development

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

## 7.1 Extensibility in JavaScript Coding

For release specific JavaScript coding, code has to be written in release specific JavaScript file.

It follows the naming convention as : (Function Id)\_(Release Type).js *Example: Code in 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 ()*

# 7.2 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 that 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 form 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 form 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



Development of Call Form [May] [2022] Version 14.5.5.0.0

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

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

Copyright © 2007, 2022, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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