Development of Call Forms Oracle FLEXCUBE Investor Servicing Release 14.5.1.0.0 [September] [2021]

FINANCIAL SERVICES



Table of Contents

1. Prefa	ace	3
1.1	Audience	3
1.2	Related Documents	
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	-
4.2	Preferences	5
4.3	Data Sources	
4.4	Data Blocks	7
4.5	Screens	
4.6	Field Sets1	
4.7	Actions1	
4.8	Launch Forms1	
4.9	Call Forms	
4.10	Summary	
4.11	Preview	
	ching Call Form to Main Function Id1	
	erated Units1	
6.1	Front End Units	
6.1.1	Language xml1	5
6.1.2	SYS JavaScript File1	5
6.1.3	Release Type Specific JavaScript File1	5
6.2	Data Base Units1	5
6.2.1		
6.2.2	System Packages1	5
6.2.3	Hook Packages1	6
6.3	Other Units1	6
6.3.1		
0.0.0		
	nsible Development1	
7.1	Extensibility in JavaScript Coding	
7.2	Extensibility in Backend Coding1	7

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.pdf</u> <u>05-Development_WorkBench_Screen_Development-II.pdf</u> <u>14-Development_of_Online_Forms.pdf</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"
- Chapter 4 , "Screen Development"
- Chapter 5 , "Generated Units"
- <u>Chapter 5 , "Extensible Development"</u>

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	📓 🗷 🗏 🚱
Function Id CFCTRCHG	Parent Function	Header Template None
Save XML Path [CFCTRCHG_] BROWSE	Parent Xml	Footer Template
ch		
Preferences		
DataSource		
ListOfValues		
🛅 DataBlocks		
Creens Screens		
FieldSets		
Actions CallForms		
LaunchForms		
Summary		

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										_ >
						l	×		¥ (3 🔿
Action Load 🔻	Function Typ	e Parent 👻]		Function Categor	y Transaction 👻				
Function Id CFCTRCHG	Parent Functio	n			Header Templat	e None 👻				
Save XML Path CFCTRCHG_F	BROWSE Parent Xn	1			Footer Templat	e 🗸				
Search	Preferences									9
Preferences DataSource DataSource DataBlocks DataBlocks DataBlocks Gamerical Screens Actions CallForms CallForms Summary	Head Office Fun Logging Require Auto Authorizatio Tank Modificatio Field Log Require Multi Branch Acc	d n ns ed ess		Module Module Description Branch Program Id Process Code SVN Repository URL Transaction Block Name Transaction Field Name	The ICCF			String I		1
	Function Id		Module *		Mo	dule Description	Sincion			
	CFCTRCHG	¢F		×1	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

Function Generation				. ×	= 1	7	_ >
Action Load V Function Id CFCTRCHG Save XML Path CFCTRCHG J BROW Search Dat	Function Type Parent Parent Function Parent Xmt ta Source Details		Function Category Transaction	-		- (9
Preferences DataSource Construct_CHG Construct_CHG Construct_CHG Construct_CHG Construct_CHARGE_ASSOC Construct_CHARGE_LIOD_MASTER CONSTRUCT_CHARG	Data Source CSTBS_CONTRACT_CHG Master Ves Relation Type One To One Muttl Record No PK Cols CONTRACT_REF_NO-LATEST_EVENT_ PK Types VARCHAR2-NUMBER Upload Table	Parent Relation Where Clause Default Order By Type	Normal Mandatory				
Fig 4.3 /	Adding data sources and maint	taining 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

			🔚 🗶 🗏 🐬 🍕 🔿
Action Load	Function Type Parent Parent Function BROWSE Parent Xmt Data Source Field Details	Function Category Header Template Footer Template	None -
DataSource DataSource CSTBS_CONTRACT_CHC CONTRACT_REF_NO LATEST_EVENT_SEQ_NO CTIBS_CHARGE_ASSOC CIFES_CHARGE_LIQD_MASTEF ListOfAues Screens FieldSets Addons CallForms CallForms Summary	Column Name CONTRACT_REF_NO Block Name BLK_CHARGES	Data Type VARCHAR2 MaxLength 16 Upload Table Column I Not Required in Upload Tables	

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 Gen

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

×

Function Generation			-	×
			🖫 🗷 🗏 🔗 🍕	⇔
Action Load - Function Id CFCTRCHG Save XML Path CFCTRCHG_F	BROWSE Block Properties	Function Type Parent Parent Function Parent Xml	Function Category Transaction Header Template None Footer Template Footer Template Footer Template Footer Template	ľ
Preferences DataSource DataSource DataBlocks DataBlocks DataBlocks DataBlocks DataBlocks DataBlocks DataBlocks DataBlocks DataBlockCHARGE_ASSOCIATION DataBlockCHARGE_APPLICATION DataBlockCHARGE_LIQUIDATION DataBlockCHARGE_H	Block Name Block Title Parent Relation Type Block PK Fields	BLK_CHARGES	XSD Node Annotation Master Block Yes V Mutil Record No Vormal V Block Type Normal V Mormal V	

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				_ :
				🖫 🗷 🗏 7 🧐 🔿
Action Load -	Function Type Parent Parent Function	t	Function Category Transac Header Template None	ction 👻
Save XML Path CFCTRCHG_F	Parent Xml		Footer Template	•
Search	Block Field Properties			- xi 🗔 🗐
Preferences DataSource ListOV/alues DataBlocks DataBlo	Field Name LATEVINSEONO Field Label LBL_LATEVINSEONO DataSource CSTBS_CONTRACT_CHG Column Name LATEST_EVENT_SEO_NO Data Type Volume Test Volume Parent Field Related Field LOV Name Fieldset Name Fieldset Name Fieldset Name Fieldset Vame Fieldset Vame Fieldset Vame Cutatom Attributes Events Related Field	11010 0120		Required Visible Read Only Calender Text Popup Edit Required Uppercase Only CLOV Validation Required Input by LOV Only Not Required In Xsd Report Parameter
	Attribute Name	Attribute Value	Active Po	sition A

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

Preferences Screen Name CVS_CFCTRCHG ✓ Main Screen	a
Screen Details Preferences DataSource DataSource DataBlocks Screen Title Ext Button Type Default Ok Cancel	Χ. 🗔
Preferences Screen Name CVS_CFCTRCHG ☞ Main Screen □ DataSource Screen Title LBL_CHARGE_DETAILS ✓ Visible □ DataBlocks Screen Size Medium ✓ □ Screens Exit Button Type Default Ok Cancel ✓	AL 💽
B □ DataSource Screen Title LBL_CHARGE_DETAILS Visible Correen Size Screen Size Exit Button Type Default Ok Cancel Visible	
BODY Description: Description:	+ -
and a regulation of the second	*
Califorms	
a Launchforms ELSN Y BLK_CHARGES Y LATEVNSEONO Y Yes Y	

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				_ ×
				🔚 🗷 🗏 7 🧐 🔿
Action Load	BROWSE	Function Type Parent Parent Function Parent Xml	•]]	Function Category Transaction Header Template None Footer Template
Search	Tab Details			Dependent Fields 🐥 🗕 🗷 🗐 🥤
Preferences DataSource ListOfValues DataBlocks Screens GCASCCHCHG GHADER GCASCC SEC_ASSCC SEC_ASSCC SEC_ASPL SEC_LIOD GHADER GEGSets Actions Califorms Summary	Screen Name Tab Name Tab Label Tab Type	CVS_CFCTRCHG TAB_MAIN LBL_ALL Data V	visible 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					-
					1 77 🧐 🤫
Action Load Function Id CFCTRCHG Save XML Path CFCTRCHG_F	BROWSE	Function Type Parent Parent Function Parent Xml		Function Category Transaction V Header Template None V Footer Template V	
Search	Section Details				- 🛛 🌍
Preferences DataSource DataSource DataSource DataBlocks Screens	Section Name SEC_ Section Label	MAIN	Visible Collapse		
CVS_CFCTRCHG HEADER	Partition Details				+-
BOPY BOPY BEC ASSOC SEC ASSOC SEC ASSOC SEC APPL SEC ASSOC SEC APPL FIeldSets Actions Califorms LaunchForms Summary	Partition SI No	PARLMAIN_1 PARLMAIN_2	Partition Name	Vidth Sub-partitions 50 v v 33 v 0 v	

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		Function Category Transaction	*
Function Id CFCTRCHG	Parent Function		Header Template None 👻	
Save XML Path CFCTRCHG_F	ROWSE Parent Xml		Footer Template	•
earch	Fieldset Properties			- 🛛 🗳
Dependences	Fieldset Name FLD_FCCREF	Screen Name CVS	S_CFCTRCHG -	🗂 Horizontal Fieldset
⊞ DataSource ≣ ListOfValues	Fieldset Label	Screen Portion Bod	y 👻	ReadOnly
DataBlocks	Data Block BLK_CHARGES -	Tab Name TAB	MAIN -	Navigation Button
Screens	Multi Record No 👻	Section Name SEC	C_MAIN -	Visible
CVS_CFCTRCHG	View Type Single -	Partition Name PAR	Rt_MAIN_1 -	
	Fieldset Height	Number Of Rows		
EODY TAB_MAIN				
SEC_ASSOC	Data Block Fields	Field S	Set Fields Subpartition Name	
SEC_APPL			· · ·	
SEC_LIQD				
B DOTER		CONREFNO		
FIEIDSETS				
FST_CHARGE_ASSOCIATION				
FST_CHARGE_APPLICATION		44		
DFST_CHARGE_LIQD_MASTER				
Actions				
CallForms				
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 👻		Function Type Paren	t v	Function Category Transaction					
Function Id CFCTRC	IG	Parent Function		Header Template None -					
Save XML Path CFCTRC	IG_F BROWSE	Parent Xml			Footer Template	•			
h	Form Actions								
Preferences DataSource DataSource DataSource DataBlocks DataBlocks DataBlocks DataBlocks CallForms CallForms Summary	XSD Type Identi Operation			Service Name	CFCTRCHG	73			
							+-		
	Web Service	Action Code	Oper	ration Code	Action Stage Type	Amendables	^		
	~	QUERY				Amendables			
		NEW				Amendables			
		MODIFY				Amendables			
		AUTHORIZE				Amendables			
	~	DELETE				Amendables			
		CLOSE REOPEN REVERSE				Amendables			
						Amendables			
						Amendables			
		ROLLOVER				Amendables			
		CONFIRM				Amendables			
		LIQUIDATE				Amendables			
		SUMMARYQUERY			V				

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							
🛯 🖣 1 of 1 🕨 🕅	Go to Page						==
Creation ESN	Compone	ent *	Rule		cription	Consider as Discour	*
							-
•						Þ	
Charge Application							
I							==
Creation ESN	Component *	Tag Currency	Tag Amount	Ch	arge Currency	Charge Amount	*
							-
<							
Charge Liquidation							
I≪ 1 of 1 ▶ ▶I	Go to Page						==
Event Sequence Numbe	r Component	Charge Currer	ncy Charge An	nount	Liquidated		*
					\checkmark		
							-
•			III			4	
							F -24
						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

ii) 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 Forms [September] [2021] Version 14.5.1.0.0

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], [2021], 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.