# Uploading Records from Upload Table

Oracle Banking Enterprise Limits and Collateral Management

Release 14.7.0.0.0

Part No.

F75024-01

[November] [2022]

FINANCIAL SERVICES

## Contents

1.	Prefa	ce	3
	1.1	Audience	3
	1.2	Related Documents	3
2.	Intro	Juction	3
	2.1	How to use this Guide	3
3.	Over	view of Bulk Upload of Records	4
4.	Uploa	ad Framework	4
4	4.1	Naming Convention:	5
4	4.2	Process Table	
4	4.3	Upload Tables	6
	4.3.1	Guide Lines	6
4	1.4	Trigger on Upload Table	8
	4.4.1	Guide Lines	
4	4.5	Upload Adapter Package	9
5.	ODT	Capabilities	
:	5.1	Configuration of Upload Table Details in RADXML	9
	5.2	Generated Units	14
	5.3	Upgrade Capabilities	14
6.	Misce	ellaneous	15
(	5.1	Appending Data	15

## 1. Preface

This document describes standard framework in FLEXCUBE for uploading records from upload tables.

### 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>

## 2. Introduction

### 2.1 How to use this Guide

The information in this document includes:

- <u>Chapter 2 , "Introduction"</u>
- Chapter 3, "Overview of Bulk Upload of Records"
- Chapter 4 , "Upload Framework"

• <u>Chapter 5 , "ODT Capabilities"</u>

## 3. Overview of Bulk Upload of Records

Bulk upload of Records to FLEXCUBE through upload tables is commonly used for uploading data from an external system periodically.

Data is populated in the upload tables through Macro Excel Upload or any other utility. (Note: Data Population in upload tables should be taken care by custom team .ODT tool does not provide feature for data population )

Thereafter upload routine is invoked from the Screen **CVDUPLOD** for the particular function id. Upload routine processes each record from upload tables. Status of processing will be updated in a **process table** for monitoring purpose.

Upload routine should follow the **same flow** as that of Gateway/FLEXCUBE User Interface to ensure integrity and consistency for records uploaded through different routines. This necessitates the need for a standard framework for uploading records from upload

tables.

A standard framework for the same has been developed using ODT which is described in the sections below.

# 4. Upload Framework

Upload Framework supports upload of both maintenance and transaction screens.

Different steps involved in Bulk Upload can be enlisted as:

- 1) Data is populated in the upload tables through Macro Excel Upload or any other utility.
- 2) *Trigger on Master Upload Table* would insert entries into a *process table* with Upload Status as 'U' (Unprocessed). One entry would be inserted into *process table* for each record. Function id would also be updated in the process table along with other information.
- 3) Upload routine is invoked for a particular function id by the user from **CVDUPLOD** screen/stub.
- 4) On invoking the routine, system would process all the unprocessed records from the *process table* for the particular function id. This would be done using a cursor on process table
- 5) An adapter package converts the upload table types to base table type data. Then it invokes the main package of the function id.
- 6) After processing of each record , *process table* columns for uploaded status, error code etc would be updated by the system

From the above steps, we can derive at the components required for a particular function id to be brought under this framework.

- 1) Process Table
- 2) Upload Tables
- 3) Trigger on Master Upload Table
- 4) Adapter package for Upload Routine

5) Wrapper code in CVDUPLOD screen processing logic to call the adapter package based on the function id

### 4.1 Naming Convention:

Framework does not enforce a standard naming convention for upload tables. Existing upload tables can be re-used in this framework. But if any *new* upload table is introduced, it is recommended to follow naming convention as illustrated below: *Fourth Letter of base table to be replaced with U Example: Base Table Name: STTM\_CUSTOMER Upload Table Name: STTU\_CUSTOMER* 

Recommended to follow naming convention for consulting / client developed *Upload Table Name:* Table name \_U\_EXTGBL

## 4.2 Process Table

For Uploading, each record is processed from a cursor on process tables. This is common across all function ids.

There are 2 process tables

• CSTB\_EXT\_CONTRACT\_STAT :

This is the process table for all transaction Function ids.

Name	Туре	Nullable
BRANCH_CODE	VARCHAR2(3 CHAR)	Ν
SOURCE	VARCHAR2(20 CHAR)	Ν
PRODUCT_CODE	VARCHAR2(4 CHAR)	Y
COUNTERPARTY	VARCHAR2(35 CHAR)	Υ
EXTERNAL_INIT_DATE	DATE	Υ
MODULE	VARCHAR2(2 CHAR)	Υ
EXTERNAL_REF_NO	VARCHAR2(20 CHAR)	Ν
IMPORT_STATUS	VARCHAR2(1 CHAR)	Y
CITICUBE_REF_NO	VARCHAR2(16 CHAR)	Y
POST_IMPORT_STATUS	CHAR(1 CHAR)	Υ
EXPORT_STATUS	CHAR(1 CHAR)	Υ
USER_ID	VARCHAR2(12 CHAR)	Υ
JOBNO	NUMBER(2)	Y
CONTRACT_REF_NO	VARCHAR2(16 CHAR)	Y
ERR_CODE	VARCHAR2(11 CHAR)	Y
ERR_MESSAGE	VARCHAR2(255 CHAR)	Y
ACTION_CODE	VARCHAR2(10 CHAR)	Y
FUNCTION_ID	VARCHAR2(8 CHAR)	Y
EXTERNAL_SEQ_NO	NUMBER(22)	Ν
UPLOAD_ID	VARCHAR2(16 CHAR)	Y

Here a particular record from upload Tables would be picked by combination of

EXTERNAL\_REF\_NO, EXTERNAL\_SEQ\_NO, BRANCH\_CODE and SOURCE . Columns like EXPORT\_STATUS, CONTRACT\_REF\_NO, ERR\_CODE and ERR\_MESSAGE would be updated by the system after processing. UPLOAD\_ID signifies the thread of execution. Upload routine can be invoked in

#### • **STTB\_UPLOAD\_MASTER** :

This is the process table for all maintenance Function ids.

multiple threads if multiple upload ids are present.

Name	Туре	Nullable
MAINTENANCE_SEQ_NO	VARCHAR2(16 CHAR)	Ν
BRANCH_CODE	VARCHAR2(3 CHAR)	Ν
SOURCE_CODE	VARCHAR2(15 CHAR)	N
MAINTENANCE_TYPE	VARCHAR2(15 CHAR)	Υ
UPLOAD_STATUS	CHAR(1 CHAR)	Υ
UPLOAD_INITIATION_DATE	DATE	Y
USER_ID	VARCHAR2(12 CHAR)	Υ
ACTION_CODE	VARCHAR2(15 CHAR)	Υ
SOURCE_SEQ_NO	NUMBER	Ν
UPLOAD_ID	VARCHAR2(16 CHAR)	Υ

Here a particular record from upload Tables would be picked by combination of MAINTENANCE\_SEQ\_NO, SOURCE\_SEQ\_NO, BRANCH\_CODE and SOURCE\_CODE.

UPLOAD\_STATUS would be updated by the system after processing a record. UPLOAD\_ID signifies the thread of execution. Upload routine can be invoked in multiple threads if multiple upload ids are present

### 4.3 Upload Tables

Each Data source in the function id, if required, should be mapped to corresponding Upload Tables in ODT.

• Data Source Column Mapping

Mapping of Upload Table Columns to Base Table Columns has to be done after proper analysis. Avoid including internal processing columns/invisible field columns etc to upload Table. This will reduce complexity of upload table.

#### 4.3.1 Guide Lines

Some guidelines for mapping upload table/columns with base table/columns are enlisted below:

- Master Data Source of the Function Id should always be a mapped to an Upload Table (except in case of some call forms where it is not feasible). This upload table would be referred to as **Master Upload Table**
- Upload Tables should be mapped only to **Normal** Data Sources as per ODT configuration. For query, in only and summary data sources; upload tables are not required

• More than one data source in the Function Id can be mapped to a single upload table. Note that all the base tables should one to one relationship with each other in this scenario.

*Example:* Both CSTB\_CONTRACT and FXTB\_CONTRACT\_MASTER can be mapped to the same upload table, say, FXTB\_UPLOAD\_MASTER.

- If the master data source is a common table used across many functions (*Example: CSTB\_CONTRACT*), try grouping it with any of its child table; so that the master upload table is unique for the function id. *Example: Both CSTB\_CONTRACT and FXTB\_CONTRACT\_MASTER can be mapped to the same upload table, say, FXTB\_UPLOAD\_MASTER.*
- It is recommended to provide the same column names to both base table column and upload table column. This avoids complexity to both developer and user.

COLUMN NAME	Remarks
SOURCE_CODE	Specifies External Source
SOURCE_REF / MAINTENANCE_SEQ_NO	Specifies External Reference Number. SOURCE_REF is used for Contract upload tables while MAINTENANCE_SEQ_NO for maintenance upload tables
SOURCE_SEQ_NO	Specifies Source Sequence Number.
BRANCH_CODE	Specifies branch Code
	This column is required only in Upload Master table. This is mandatory if same upload master tables are used for multiple function ids. <i>Example : Parent and</i>
FUNCTION_ID	Child Functions
	This is required only in Upload Master Table. If not present , then only NEW Operation would be supported by upload framework for the Function
ACTION_CODE	Id.
	This is required only in Upload Master Table. Different values can be inserted for this column in batches to process upload routine in multiple threads. This is an optional column; mostly used in
UPLOAD_ID	transaction screens
UPLOAD STATUS	Optional. Required only if Upload Table and Master Table are the same.eg: PC contract. This is used mostly in transaction screens
	Module Code of the Function Id.
MODULE	Optional; used mostly in transaction screens
	Optional. Specifies the Source Operation code. This needs to present in only master upload table, if any. If not present , then system would try to derive the default SOURCE_OPERATION for particular action
SOURCE OPERATION	code

• Apart from Mapped Columns from Base Table, Upload Table should have a standard set of columns as defined below

SOURCE\_CODE, SOURCE\_REF / MAINTENANCE\_SEQ\_NO, SOURCE\_SEQ\_NO and BRANCH\_CODE form the composite primary key for any master upload table. For detail upload tables, the 4 columns mentioned above along with unique identifier for the record, if any, forms the primary key FUNCTION\_ID, ACTION\_CODE, UPLOAD\_ID, UPLOAD\_STATUS, MODULE and SOURCE\_OPERATION are optional columns in Master Upload Table

- For Transaction screens, EXTERNAL\_REF\_NO of upload table has to be mandatorily mapped to a base table column. This is required to derive the reference number in case of any modify operation. Most often, this column can be found in CSTB\_CONTRACT.
- More than one data source can be mapped to the same upload table differentiated by *upload table where clause*.
   For Instance; if two different legs of a transaction (buy and sell) of a deal are captured by two data sources in function id (same table with different aliases); then one upload table can be used for both the tables. Upload where clause for both these data source should be such that the adapter picks proper data to base table data types Note that in this scenario, both the data sources should not be directly related to each other. Difference has to be noted between this scenario and the case where 2 data sources with one to one relationship is mapped to same upload table.
- For call form function ids, master data source would often be a view for propagating record key to the call form. In such instances master data sources should not have any upload table mapped to it. *Example: CSTBS\_CONTRACT\_ADV is the master data source for Advice Call form but data is uploaded only in CSTB\_CONTRACT\_EVENT\_ADVICE. Hence upload table should not be mapped to CSTBS\_CONTRACT\_ADV*

Sample Master Upload Table definitions are attached:

## 4.4 Trigger on Upload Table

Triggers would be created on Master Upload Table to insert records into Upload Process Tables on insert of records in upload tables. For Uploading, each record is processed from a cursor on process tables.

### 4.4.1 Guide Lines

- If column for action code is not present in the master upload table; then Action code column in process table would be updated as NEW
- If upload routine is present for parent and child function ids; then the master upload table would be the same. *In such cases, FUNCTION\_ID column*

*should be present in the master upload table* and the same would be inserted into the process table. Hence the same trigger would hold good for all the child screens.

• There would be no separate trigger for any call form function ids as call form records does not exist independently.

Sample Upload Table Trigger is attached





Maintenance\_opioad\_ringgei.sqr

## 4.5 Upload Adapter Package

Upload Packages would handle type conversions and processing records after conversion.

- Naming Convention Module||' pks '||FunctionId||'\_Ext\_Upload'
  - Example : fxpks\_fxfdtronl\_ext\_upload

Based on structure, upload packages can be broadly classified as

#### 1) Transaction Upload Adapter

Records will be processed based on cursor on CSTB\_EXT\_CONTRACT\_STAT. Code to handle *SUBSYSTAT* will be present.

- 2) Maintenance Upload Adapter Records will be processed based on cursor on STTB\_UPLOAD\_MASTER
- 3) Transaction Call forms Upload Adapter It will be called from Transaction Upload Package. Code to update *SUBSYSSTAT* will be present
- **4) Maintenance Call Forms Upload Adapter** It will be called from Maintenance Upload Packages

## 5. ODT Capabilities

ODT supports extensible upload framework. Upload Framework components can be generated by the tool through configurations.

## 5.1 Configuration of Upload Table Details in RADXML

### **Upload Table :**

In data source definition screen, upload table name for the data source has to be specified.

Function Generation								_ >
						×	V	🧐 🌩
Action Lead  Function Id FXDTRONL Save XML Path FXDTRONL_R	BROWSE	Function Type Parent Parent Function Parent Xml		Function Category Transaction Header Template None  Footer Template None	<ul><li>▼</li></ul>			
Search	Data Source Details							• • ?
Preferences DataSource Contract Unater Contract Unater Contract Event Lo FATBS_CONTRACT UNATER FATBS_CONTRACT_UNATER FATBS_CONTRACT_UNATER FATBS_CONTRACT_OPTION_DTLS FATBS_UNATIO_OPTION_DTLS FATBS_UNATIO_OPTION_DTLS FATBS_UNATIONS	Master Yes Relation Type One Multi Record No PK Cols • CONT PK Types • VARC Upload Table XTB Upload Where Clause	To One  To One Tract_REF_NO	Parent Relation Where Clause Default Order By Type	Normal  Mandatory				
Fi	gure 1: Specifyin	g Upload Table f	or a Data Source in ODT					

- Avoid providing Synonyms in Upload Table field.
- Upload Tables should be mapped only to Normal Data Sources
- The standard set of columns for the upload table can be viewed by clicking on the button next to Upload table.
- SOURCE\_CODE, SOURCE\_REF, SOURCE\_SEQ\_NO and BRANCH\_CODE will be assumed as part of primary key of any upload table. Make note of the guide lines explained in previous section while providing Upload Table Name

#### **Upload Table Standard Columns:**

Refer previous section for the standard columns which are part of the upload table. Default column names are provided in the screen. Source Operation would be not present in the table by default.

Developer can change the column names of the standard columns as desired. This could be useful if existing upload tables are re-used.

*Example* : Name of the column for External reference number can be changed from SOURCE\_REF to EXT\_REF\_NO

ita Source Detai		Table Standard Columns		
Dala Source Masier	CSTBS_CONTRACT			
Relation Type Multi Record	One To One 💌		Columns	Not Required
	CONTRACT_REF_NO	Source code	SOURCE_CODE	
	VARCHAR2	External Reference Number	SOURCE_REF	
Upload Table	FXTB_UPLOAD_CONTRACT	Source Sequence Number	SOURCE_SEQ_NO	
ad Where Clause		Branch Code	BRANCH_CODE	
		Function Id	FUNCTION_ID	
		Action Code	ACTION_CODE	
		Upload Id	UPLOAD_ID	
		Module Code	MODULE	
		Source Operation		
				Ok Cancel

Figure 2: Standard Columns for Master Upload Table of Transaction Screen

BROWSE Parent Xmi		Fooler Template Maint Audit	<b>.</b>
Data Source Details	Table Standard Columns		×
Data Source Details           Data Source         CATMS_AMOUNT_BLOCKS           Master         Yes           Relation Type         One To One           Multi Record         No           PK Cols *         AMOUNT_BLOCK_NO           PK Types         VARCHAR2           Upload Table         CATU_AMOUNT_BLOCKS	Source code External Reference Number Source Sequence Number Branch Code Function Id	Columns SOURCE_CODE MAINTENANCE_SEQ_NO SOURCE_SEQ_NO BRANCH_CODE Column FUNCTION_ID	Not Required
	Action Code	ACTION_CODE	
	Source Operation		
			Ok Cancel

Figure 3: Standard Columns for Master Upload Table of Maintenance Screen

Dala Source Masler	FXTBS_CONT_OPTION_DTLS	Pa Rela	tion CSTBS_CONTRACT		
Relation Type	One To Many 💌	Where Cla	190		
Multi Record	Yes 💌	Table Standard Columns			
	CONTRACT_REF_NO-VERSION_NC				
	VARCHAR2-NUMBER				
	FXTU_CONT_OPTION_DTLS				
load Where Clause			Columns	Not Required	4
		Source code	SOURCE_CODE		
		External Reference Number	SOURCE_REF		
		Source Sequence Number	SOURCE_SEQ_NO		
		Branch Code	BRANCH_CODE		

Figure 3: Standard Columns for Detail Upload Table

### **Upload Table Where Clause:**

If all the records in an upload table is not be mapped to a particular data source; then upload table where clause can be specified to filter the records.

This is applicable only when the data sources involved are not directly related with each other.



Figure 4: Usage of Upload Where Clause for Differentiating data of multiple Data sources

Upload Table Where clause would be applied on the upload table; hence upload table columns should be used in the clause

#### Upload Table Column:

All the data source columns which are included in the RADXML would be assumed to be part of the upload table.

- By default, name of the Upload Table column would be assumed to be same as that of the base table name
- If the name of the upload table column has to be different from base table column; then the same has to be explicitly mentioned in Upload Table Column Field

Function Generation								-	
						× =	V	9	Ş
Action Load		Function Type Parent	-	Function Category Transaction	-				
Function Id FXDTRONL		Parent Function		Header Template None 🔻					
Save XML Path FXDTRONL R	BROWSE	Parent Xml		Footer Template None	-				
	Data Oanna Eista	D-(-i)-				_			
Search EXT	Data Source Field	Details				Ref	resh 🗕	- 5	ľ
SPOT_DATE ^	Column Name	EXTERNAL_EX_RATE	Data Type	NUMBER					
SPOT_LCY_EQLNT	Block Name	BLK CONTRACT MASTER	MaxLength	22					
PREMIUM_DISCOUNT_AMT			Upload Table Column		_	٦			
PREM_DISC_INDICATOR	Field Name		Upload Table Column	Not Required in					
INTEREST_RATE_BOT_CC\ INTEREST_RATE_SOLD_C(				Upload Tables					
DEALER									
LCY_EQUIVALENT									
SPOT_RATE									
BOT_AC									
EVENT_CODE									
SWAP_REFERENCE_NO									
TAX_SCHEME									
SET_SOLD_CCY									
SET_BOT_CCY									
EXTERNAL_EX_RATE									
CUST_SPREAD									
PARENT_EXT_CONTRACT_									
BOT_LEG_RATE									
SOLD_LEG_RATE									
BOT_LEG_PREM_DISC									
SOLD_LEG_PREM_DISC									
LCY_DISCOUNTED_AMOUN									
SPREAD_DEFN INTERNAL_SWAP_REF_NO									
EXCLUDE_FROM_CLS									
CLS_ELIGIBLE									
NETTING MODE									
FX_NETTING_CUSTOMER									
RISK_RATE									
WEIGHTED_RISK_AMT									
LIMITS_TRACKING_REQUIF									
TRACK_SETTLEMENT_RISH									
TRACK_PRE_SETTLE_RISK									
PRE_SETTLE_RISK_LINE 👻									
۰ III ا									•

Figure 5: Changing Column Name for Standard Columns in Upload Tables

• If any of the columns included in the data source in RADXML is not required in Upload table; then the same has to specified by selecting the checkbox **Not Required in Upload Tables** 

Function Generation			_ ×
			77 🧃 🔿
Action Load 👻	Function Type Parent	Function Category Transaction	
Function Id FXDTRONL	Parent Function	Header Template None 👻	
Save XML Path FXDTRONL_F	BROWSE Parent Xml	Footer Template None	
Search SPREAD_D	Data Source Field Details	Refr	esh 🗕 🏟 🤺
PREMIUM_DISCOUNT_ANT PREM_DISC_INDICATOR INTEREST_RATE_SOLD_CC DEALER EFFECTVE_INT_RATE LCY_BOUNALENT BOT_AC EVENT_COOLE SWAP_REFERENCE_NO TA_SCHEME SET_SOLD_CCY SET_SOLD_CCY SET_SOLD_CCY SET_SOLD_CCY SETTERNAL_SXARTE DOT_LCS_RATE BOT_LCS_RATE BOT_LCS_RATE BOT_LCS_RATE BOT_LCS_RATE DOLD_RAND_REF_NO EXCLUDE_RKNU_CLS CLS_ELIGIBLE NETTING_MODE RKK_RATE WEIGHTED_RISK_LINE TRACK_SETLE_RISK_LINE WEIGHTED_RISK_LINE	Bick_CONTRACT_MASTER Field Name	Data Type CHAR Max Length Upload Table Column V Dolad Tables Volcad Tables	

Figure 6: Data Source Column not required in upload Table. Explicitly specifying the same

## 5.2 Generated Units

1) Upload Adapter Package

Naming Convention : Module||'\_pks\_'||FunctionId||'\_Ext\_Upload'

- 2) Triggers on Upload table
- 3) Upload Table DDL

If existing upload tables are being used, DDL scripts can be ignored. Drop scripts for the table would be generated in a separate file.

### 5.3 Upgrade Capabilities

Normal ODT upgrade feature is supported in Upload table configurations as well. Customizations can be done on the configuration maintained in ODT Customizations can be done to:

- i) Change the Upload Tables mapped.Map new Upload tables/Remove existing table mapping
- ii) Modify/Remove/Add Upload table column names
- iii) Configure upload tables for entire screen if it is not provided by engineering and if bank needs the same

#### *Changes done as part of customizations would be retained during ODT Refresh.* Any new mappings done by the engineering team would reflect after Refresh. After Refresh all the artifacts has to be regenerated including upload table definitions.

# 6. Miscellaneous

## 6.1 Appending Data

In certain scenarios, only the data which has to be appended would be uploaded to the upload tables. Requirement would be to append this data to the existing data in the table. This feature is not supported by the standard FLEXCUBE framework *Example : Upload of Floating Rates for a Currency* 

#### **FLEXCUBE Framework :**

In FLEXCUBE, if any multi Record Block has to be modified then the complete data of the block has to be sent. FLEXCUBE will derive the records modified, deleted or added in the block according to the data sent and updates the tables accordingly. *Hence if only the data to be appended to the block is sent, then the existing records would be treated as deleted and hence deleted from the tables.* 

#### Solution:

To handle this scenario, the following approach is recommended: Handle this case based on the source operation parameter in the custom package. *Developer can either* 

- 1) skip all the system functions and write code to upload data in custom package for particular source operation OR
- 2) append the existing data to screen object instance before start of processing ( preferably in pre\_check\_mandatory) for the particular source operation



Uploading Records from Upload Tables [November] [2022] Version 14.7.0.0.0

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.