Development Workbench - Notifications Oracle Banking Corporate Lending Release 14.7.2.0.0 Part No. F92377-01 [November] [2023]

FINANCIAL SERVICES

# Contents

1	Pref	ace	.3
	1.1	Audience	.3
2	Intro	oduction	.3
ł	How t	o use this Guide	.3
3	Noti	fication – Getting started	.4
	3.1	What is Notification	.4
	3.2	Notification Trigger	.4
4	Noti	fication Development	.4
2	4.1	Pre-request for Notification development and testing	.4
2	4.2	Notification specification	.5
2	4.3	Notification XML development	.5
2	1.4	Notification Process	.5
2	4.5	Development process in Development Workbench	.5
2	4.6	Notification Trigger	.6
2	4.7	Notifications1	12
5	Dep	loy Notification1	18
[	5.1	Notification - Workbench related deployment1	18
[	5.2	Notification Trigger deployment1	18
6	Test	t Notification1	18
6	5.1	Notification flow1	19
6	5.2	Scheduler based notification1	19
(	5.3	MDB based notification flow2	20
6	5.4	Triggering notification and testing2	21

# 1 Preface

This document describes the steps to develop the notification XML and notification trigger using Oracle FLEXCUBE Development Workbench for Universal Banking.

### 1.1 Audience

The Development Workbench Notification Development book is intended for the FLEXCUBE Application Developers who perform the following tasks:

Develop new Notification

To Use this manual, you need conceptual	and working knowledge of the below.
Proficiency	Resources
FLEXCUBE UBS Development	FCUBS-FD01-01-01-Development
overview	Overview Guide
Interface Getting started	FCUBS-FD04-01-01-Interface Getting started
FLEXCUBE Development Workbench for Universal Banking Reference	User manuals
Web service development to have query web service in place	FCUBS-FD02-03-01-RAD Web Service Development

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

# 2 Introduction

# How to use this Guide

The information in this guide includes:

- <u>Chapter 3, "Introduction"</u>
- <u>Chapter 4, "Notification Getting started"</u>
- <u>Chapter 5, "Notification Development"</u>
- <u>Chapter 6, "Deploy Notification"</u>
- <u>Chapter 7, "Test Notification"</u>

# 3 Notification – Getting started

### 3.1 What is Notification

Notification framework in FLEXCUBE UBS is used to communicate the business event happened in FLEXCUBE UBS to external systems. Depending upon the event, the XML message is pushed to external system's asynchronous Queues for their consumption.

# 3.2 Notification Trigger

Notification Triggers is developed to recognize the event and then invoke the notification process. This trigger is developed using Development Workbench.

# 4 Notification Development

## 4.1 Pre-request for Notification development and testing

Following are pre-request for notification development:

- Target FLEXCUBE Environment with Notification framework installed
- Development Workbench link mapped to the FLEXCUBE environment
- Required Query Web services developed and tested

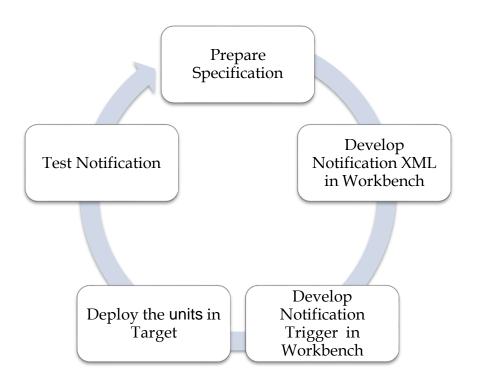


Fig 5.1.1: Development of Notifications

# 4.2 Notification specification

Identify the notification requirement as below

- What is the Notification function ID name for RAD XML (Third character should be N)?
- What is the Notification code?
- What is the Base table in FLEXCUBE UBS that triggers the notification?
  - What operation at base table triggers (insert/update/delete)?
  - What is the where clause for filter?
- What is the query Web service to be used?
  - What is the operation?
  - What are the tags required?

#### Example;

- Notification function ID name *STNCUMOD*
- Notification code NOTIF\_CA\_CUSTACC\_MOD
- Base table STTM\_CUST\_ACCOUNT
  - Operation DELETE
  - Filter Account class type in (S, U)
- Web service to be used *FCUBSAccService* 
  - Operation *QueryCustAcc*
  - Request node Cust-Account-IO

## 4.3 Notification XML development

Notification RAD XML development creates the following files:

- RAD XML
- SPC
- SQL
- Static Data

#### 4.4 Notification Process

There will be one trigger for the base table of notification and in case of multiple notifications sharing the same base table, there will be no new triggers created. Instead the same trigger created on the base table will be reused. This approach reduces the number of triggers being used for notifications.

## 4.5 Development process in Development Workbench

The notification development process in Workbench is split into two steps:

- 1. Notification Triggers
- 2. Notification Filter Procedure

The first step is to create notification triggers for base tables. The trigger generated from Workbench will be inserting key details into a static notification log table. The following details will be captured:

- Trigger code: A unique value to for a notification trigger.
- Base Table: The base table on which, the trigger is built.
- When Clause: A simple when clause for the notification trigger.

The second step is to capture details of notifications and generate the notification filter procedure. The following details are captured:

- Notification code: A unique value to identify a notification.
- Description: Meaningful description of the notification.
- Gateway Service:

#### 4.6 Notification Trigger

After successful login to Development Workbench click on Notification Trigger option in the tree as shown below:

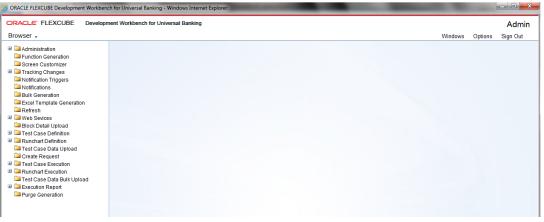


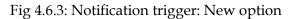
Fig 4.6.1: Notification trigger

Notification Trigger		- ×
		🧧 🛛   🖬 📴 🍃   🖗 🥥 🔿
Trigger Code *	Base Table *	1
Description Firing Time	Before  PK Types *	
Each Record	Yes v Data Types	
Selected Columns	Notification Codes	
Trigger When Clause		
Trigger Logic( Set \$NC	OTIFY To Y/N)	
	А	
	*	

Fig 4.6.2: Notification trigger options

Notification Trigger we have two options - Add a new Trigger or Modify Existing one. **New:** 

Notification Trigger	_ ×
	🎴 🗙   🖬 🌃 🎦   🖗 🎯 🔿
Trigger Code * TRG_CUSTADDR Base Table * MSTM_CUST_ADDRESS	
Description Trigger for Customer Address Maintenanc PK Cols * CUSTOMER_NO~LOCATION~MEDIA	
Firing Time After v PK Types * VARCHAR2-VARCHAR2-VARCHAR2-VARCHAR2	
Each Record Yes T Data Types	
Selected Columns Notification Codes Tripoer When Clause (new.auth_stat=A)	
Trigger When Clause (new.auth_stat='A')	
Trigger Logic(Set \$NOTIFY To Y/N)	
IF NVL(:oldonce_auth, N) ↔ Y THEN A	
ELSE	
L_Operation := "UPDATE"; END IF;	



**Trigger Code:** A unique value to for a notification trigger. Follow the naming conversion it should start with **TRG\_XXXX**. This is mandatory field. This attribute signifies the trigger code created as part of trigger creation step in OTD. Each notification will be linked to a trigger code.

**Description** : Information field. Meaningful description of Trigger is to be given.

**Firing Time** : Specify when trigger needs to fired. We can create only BEFORE and AFTER triggers for tables. (INSTEAD OF triggers are only available for views; typically they are used to implement view updates.) (After/Before).

**Each Record:** specify for each row required or not. If FOR EACH ROW option is specified, the trigger is row-level; otherwise, the trigger is statement-level. (**Yes/No**)

**Base Table:** The base table on which, the trigger is built. This is mandatory field. Select a valid table from available LOV next to the field.

**Pk Cols:** Enter Primary key fields of table in tilde (~) separated format. This is mandatory field.

**Pk Types:** Enter Primary key type of the corresponding primary key field. This is mandatory field.

#### Selected Columns and Data Types: Defunct

**Trigger When Clause:** A simple when clause for the notification trigger. A trigger restriction can be specified in the WHEN clause, enclosed by parentheses. The trigger restriction is a SQL condition that must be satisfied in order for Oracle to fire the trigger. This condition cannot contain sub queries. Without the WHEN clause, the trigger is fired for each row.

**Notification Codes:** If the trigger is associated with a specific notification code, then the particular notification code has to be provided in the field. If the trigger is shared across many Notifications, field can be left empty

nerate Rad Files Front End Files RadXML	System Packages							
	System Packages							
		Hook Paokages	N	ieta Data			Olturs	*
Screen Xmi	Main Package Spec Main Package Body Nofikication Trogors Upload Package Spec	Kernel Package Spac Kernel Package Body Cluster Package Body Cluster Package Body Clustom Package Body Clustom Package Body	Menu Datalis Datasource Datalis LOV Details Elock Datalis Screen Details Amendable Details Call form Datalis Surmary Details	Label Dela Block PK C Function C Galeway D V Nolification Function P Purge Dela	olumns all Forms etalls i Detalls aramoters	Screen H Upload T Upload T	Annolations Himi (able Trigger (ables Definition (able Definition	×
SLNo		File Name		File Type			Status	
1 GW1	R#_TRG_CUSTADOR lrg		TRO	3				
2 GW1	IM_NOTIFICATION_TRIGGERSTRG_C	USTADOR INC	INC					
3 TRG	_OUSTADOR_RAD xml		RAD	)XAL				
Information		×						
mormation			-					
E	rror Description	Error Code						
	ccessfully Processed	RD-SAVE-007						
								-

Fig 4.6.4: Notification trigger: Generation

On successful save Notification Trigger will generate two files (gwtr#\_<trg-code>.trg and GWTM\_NOTIFICATION\_TRIGGERS\_\_<trg-code>.INC) user needs to compile them in FLEXCUBE schema.

Modify:

fication Trigger		-
		9
Trigger Code *		
	Base Table *	
	PK Cols *	
Firing Time Before 👻	PK Types *	
Each Record Yes -	Data Types	
Selected Columns	Notification Codes	
Trigger When Clause		
TriggerLogic(Set \$NOTIFY To Y/N)		

Fig 4.6.5: Notification trigger: Modification

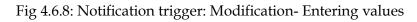
ication Trigger							×	ττ	1 15	
			igger Code		×	1				
Trigger Code *		<b>A</b> \$	igger code		^					
Description			Trigger Code							
Fining Time										
Each Record	Belore -									
Selected Columns										
					Search Reset					
Ingger When Clause					<< < 1nf5 > >>					
Trigger Logic( Set 1HC	TIFY TO YAN (			Trigger Code	*					
			CUST_AC_BRN_TFR							
			LOAN_BRN_TRFR							
			TD_AC_BRN_TFR							
			TRG_APP_DETAIL							
			TRG_BLKDELMSTR							
			TRG_BRTMMSTR							
			TRG_CATDET							
			TRG_CFRAMSTR							
			TRG_CHBK							
			TRG_CLAC							
			TRG_CLTMPRD							
			TRG_CONT							
			TRG_CSTBCTRT							
			TRG_CSTMPRD							
			TRG_CUST							
					*					

Fig 4.6.6: Notification trigger: Modification-Selecting Trigger name

Notification Trigger					- 1
				🗎 🔀	🖬 📅 🏠   🖓 🎯 ৰ
					Execute Query
Trigger Code *	TRG_CUSTADDR	Base Table *			
Description	<b>P</b>	PK Cols *			
Firing Time	Before -	PK Types *			
Each Record	Yes 🔻	Data Types			
Selected Columns		Notification Codes			
Trigger When Clause					
Trigger Logic( Set \$NO	(IFY TO Y/N)				
			w.		

Fig 4.6.7: Notification trigger: Modification- Entering values

Notification Trigger								_ ×
				×	I I7	2	17	ⓐ ⇒
				-		j 🖬 :	V	19 V
Trigger Code *	TRG_CUSTADDR		MSTM_CUST_ADDRESS					
Description	Trigger for Customer Address Maintenanc		CUSTOMER_NO~LOCATION~MEDIA					
Firing Time	After 👻	PK Types *	VARCHAR2~VARCHAR2~VARCHAR2					
Each Record	Yes 👻	Data Types						
Selected Columns		Notification Codes						
Trigger When Clause	(new.auth_stat='A')							
Trigger Logic( Set \$NO								
IF NVL(: I Ope	old.once_auth, 'N') <> 'Y' THEN ration := 'INSERT';		<u> </u>					
ELSE								
LOpe END IF;	ration := 'UPDATE';							
210 11,								
			-					
								I



									1
rate Rad Files									
									*
Front-End	d Files	System Packages		Paokages		Meta Data		Others	
		Main Package Spec	Kernel Pack		Menu Delalis	Label		Xsds	
Screen Xml		Main Package Body	Kernel Pack		Datasource Defails		PK Columns	Xsd With Annolatio	ons
System JS		Nohibcation Triggers	Cluster Pad		LOV Cretails		on Call Forms	Screen Himi	
		Upload Package Spec Upload Package Body	Cluster Paci		Eleck Delails		ay Defails ation Defails	Upload Table Trig	
		C Obioan Lackade Bonk	Custom Pac		Amendable Details		on Parameters	Archive Table Defi	
				chage booy	Call form Defails	Purge		1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	oneoni
					Summary Details	and a wey a			
									+
SLNO			File Name			File 1	lype 	Status	^
	GWTR#1	RG_CUSTADOR Irg				TRG		Generated -	
	GWTM_N	OTIFICATION_TRIGGERSTRG_C	USTADOR INC			INC		Generalad +	
	TRG_CUS	STADDR_RAD.xml				RADIXML		Generaled 👻	
	TRG_CUS	STADDR_RAD.xml				RADXML		Generaled *	
nformation	TRG_CUS	STADDR_RAD xml		×		RADXML		Generalad 👻	
formation	TRG_CUS	STADDR_RAD xml		×		RADXML		Generalad •	
formation		STADOR_RAD xml	Error (			RADXML		Generaled *	
	Error I					RADXML		Generalsd •	
	Error I	Description		Code		RADXML		Generaled	
	Error I	Description		Code		RADXML		Generaled +	
	Error I	Description		Code		RADXML		Generalad •	
	Error I	Description		Code		RADXML		Generalid *	
	Error I	Description		Code		RADXML		Generalisd *	
	Error I	Description		Code		RADXML		Generaled •	
	Error I	Description		Code		RADXML		Generaled •	~
	Error I	Description		Code		RADXML			Jerate Exit
	Error I	Description		Code		RADXML			nerate Exit
	Error I	Description		Code		RADXML			verate Exit

Fig 4.6.9: Notification trigger: Modification-Successful Generation

# 4.7 Notifications

Notifications Screen will be used to create new notification or modify existing notification; here we capture notification information for notification codes. We save notification details into xml.

	Notification Function		Action None -	Save XMI	L Path		
Notification Code	*		м	lodule			
Description			Module Descr	ription			
Notification Xsd			Base	Table *			
Firing Time			PK	Cols *			
Filter Type	Ψ		PK	Types *			
Gateway Service				Full Screen Reply			
Gateway Operation				HO Only			
Sateway IO Request							
Type XSD Name	Y To Y/N & Refer Current Record as						
				~			
Web Service Ta	igs				+-		
Web Service Ta Order	igs Xsd Field	Table Field	Data Type	T Maximum Length	+-		
		Table Field	Data Type		+-		

Fig 4.7.1: Notification Screen

**Action:** We can choose either new or Load action. New to create a new notification and Load is used to modify the existing one.

**Save Xml Path:** Specify the path to save notification xml. This would be considered only if the Save Mode is Client and Work Directory is specified as \$CURRENT\_DIRECTORY

**Notification Function:** Specify the notification function-id name. *Conventions:* 

*Maximum 8 chars.* 3<sup>*rd</sup></sup> <i>letter must be 'N'. Example: FTNCONON*</sup>

**Notification Code:** Enter the notification code to which we need to capture values. This is Mandatory field.

#### Recommended Convention for Notification Codes:

NOTIF\_<Module Code>\_<Description> Example: NOTIF\_LD\_CONTRACT This is the notification indicating that a LD contract has been created/modified

**Description:** Information field. Meaningful description of the Notification has to be provided in the field

Module: This attribute signifies the module on which the notification is based.

**Module Description:** Information field. Module Description which would be defaulted from Module LOV

**Notification XSD**: Notification XSD name will have to be provided in the corresponding Field. Naming convention to be followed while naming Notification XSD is as follows

[Module Name] – [Notification Description] – Notif.xsd Example: FT-Contract-Notif.xsd

Notification XSD has to be provided only if no Gateway Web Service Query Operation is configured to the Notification

Base Table: Select the base table on which trigger needs to be applied.

**Firing Time**: Indicates the Operation on the base Table for which Notifications has to be sent. Options available are Insert, Update or Both

Filter Type: This attribute can take the following values.

- 1. Where clause
- 2. Plsql block

Pk Cols: Enter Primary key columns of the Base Table.

Pk Types: Enter Primary key field Data Types.

Provide details of Gateway Service, Operation, Type XSD Name and Full Screen Reply if a Query Web Service has to be mapped to the Notification

**Gateway Operation:** The gateway operation name to execute query for the mentioned Service.

Gateway Service: The gateway service to be used to get the full screen response.

Gateway IO Request: The gateway IO request node to be used in querying operation.

**Type XSD Name:** This field has to be entered if Notification is mapped to a Service and Request. Name of the Master Type XSD for the service and operation has to be provided here. This can be found in include portion of the Request Msg XSD of particular Service-Operation

Example: LC-Contract-Types.xsd

**Full screen Reply:** This attribute decides whether full screen or primary key notification response to be sent. This is applicable only if gateway Service details are provided

HO only: This attribute is used to send notification only from head office.

**Filter Logic:** The filter logic which decides whether the notification needs to be sent or not. This can be a simple where-clause on base table or a complex pl/sql block.

**Web service Tags:** The columns selected from base table as part of web service tags, will be used to send the full screen notification response. These tags defines the elements of Notification Xml when no Query service is mapped to it:

Front-End Files	System Packages	Hook Packages		Meta Data	Others
RadXML Screen Xml System JS	☑ Main Package Spec ☑ Main Package Body Notification Triggers Upload Package Spec Upload Package Body	Kernel Package Spec     Kernel Package Body     Cluster Package Spec     Cluster Package Body     Custom Package Body     Custom Package Body	Menu Details Datasource Details LOV Details Block Details Screen Details Amendable Details Call form Details Summary Details	Label Details Block PK Columns Unction Call Forms Gateway Details V Notification Details Function Parameters Purge Details	Xsds     Xsd With Annotations     Screen Htmi     Upload Table Trigger     Upload Tables Definition     Archive Table Definition
SLNo	9	ile Name	File Type		Status
SLNo	F	ile Name	File Type		Status

Fig 4.7.2: Notification Screen generation

Front-E	nd Files System	Packages	Hook Packages		Meta Dala		Others	
RadXML I Main Package     Streen Xmi I Main Package     System JS I Notification T     Upload Packa     Upload Packa		Body Igers Je Spac	dy Z Kamel Package Body ers Cluster Package Spec Spec Cluster Package Body Body Clustorn Package Spec Custorn Package Body		Menu Delails     Label Delails       Datasource Delails     Block FK Columna       LOV Details     Function Call Forms       Brock Delails     Calavary Delails       Screen Details     Ranendable Details       Arrendable Details     Function Parameters       Call form Details     Summary Details		Upload Table Trigger	
SLNo			File Name		File Type		Status	~
	fipks_finconon_main.spc			S	PO		Generated +	
	flpks_ltnconon_kernel.spc			3	PO		Generated 💌	
	fipks_itnconon_main.sqi	Information			×		Generated 👻	
	flpks_itnconon_kernei.sqi						Generated *	
	GWTM_NOTIFICATIONS_MAS	1	Error Description	E	irror Code		Generated 👻	
	GWTM_NOTIFICATION_TAG_		Request successfully Processed	R	RD-SAVE-007		Generated *	
,	FTHCONON_RAD ami				Ok		Generated *	Exit

Fig 4.7.3: Notification Screen Generation Successful

#### Modifying an Existing Notification RADXML

The process of modifying an existing Notification RADXML is illustrated in the images below

Notification Function	Action Load 👻	Load Screen Xml	BROWSE
ation Code 🔻	Module		
Description	Choose File to Upload		
	🔾 🗸 🖓 « MAIN 🕨 FT 🕨 RADXML	- 49 Search RADXML	ρ
Upload File Browse	Organize 🔻 New folder	8== ¥ 🗔 😧	
Upicad File	Downloads ^ Name	Date modified Type	<u>^</u>
	Recent Places CSCFTDUP_RAD.xr		=
	ECORD Desktop		-
	a papraiu		
	ETDCONAU_RAD.x		
	System (C:)		
	DVD RW Drive   ETDMCKCH RAD		
	FLEXCUBE_Kerr     TOMCKCH_KADJ     Xperia L     Y		-
	File name:	✓ All Files (*.*)	
		Open Cancel	
Order Ksd Field Table F		umum Length	

Fig 4.7.4: Notification Screen Loading

										×	3
		Notification Function F	TNCONON	Actio	n Load 👻	Save Xml F	ath FTNCONON	_RAD.xml	BROWSE		
, j	Notification Co	de * NOTIF_FT_CONTRACT	r I		Module	TT	*=				
	Descript	ion This is the notification i	ndicating that a FT 🖓		Module Description	Funds Transfer	2				
	Notification >	(sd			Base Table	* CSTB_CONTRACT					
	Firing Ti	me Insert 👻			PK Cols	* CONTRACT_REF_NO					
	Filter Ty	/pe Pisql Block 🔹	_		PK Types	* VARCHAR2					
	Gateway Serv		*=			Full Screen Reply					
Ga	Gateway Operat		×E			HO Only					
Gat	ateway IO Requ	est Contract-Details-IO									
	Type XSD Na										
	Logic( Set \$NO	TIFY To Y/N & Refer Current F	Record as \$CURENT_RECORE de = 'FT' THEN \$NOTIFY := 'Y'; EL:		END IF; RETURN TRUE;						
ilter L	Logic( Set \$NO	TIFY To Y/N & Refer Current F RENT_RECORD.module_coo			END IF; RETURN TRUE;	×					
V	Logic( Set \$NO IF \$CURF	TIFY To Y/N & Refer Current F RENT_RECORD.module_coo	de = 'FT' THEN \$NOTIFY := 'Y'; EL	3E \$NOTIFY := 'N';		×	+-				
V	Use Service Order	TIFY To Y/N & Refer Current F RENT_RECORD.module_coor	de = 'FT' THEN \$NOTIFY := 'Y'; EL	35 \$NOTIFY := 'N'; d	Data Type	Maximum Length	+-				
V	Web Service Order	TIFY To Y/N & Refer Current F RENT_RECORD.module_coor Tags Xsd Field SOURCEREFNO	de = FT THEN \$NOTIFY := Y; EL Table Fiel EXTERNAL_REF_NO	d pe	Data Type VARCHAR2	Maximum Length					
ilter L	Web Service Order	TIFY To Y/N & Refer Current F RENT_RECORD.module_coor	de = 'FT' THEN \$NOTIFY := 'Y'; EL	35 \$NOTIFY := 'N'; d	Data Type	Maximum Length					

Fig 4.7.5: Notification Screen Loaded

	ition Maintei												_	-
											<b>.</b>	×	13	4
			Notification Function	NCONON		Actio	n Load 👻		Save Xml Path	FTNCONON_RAD.xml	BROWS		-	
r	Notificatior	n Code *	NOTIF_FT_CONTRACT		1		Modu	e FT		×E				
			This is the notification in				Module Descriptio							
	Notificatio				j			e * CSTB_CONTR/	ACT 🚬					
	Firin	g Time	Insert 👻				PK Co	s * CONTRACT_R	EF_NO					
	Filte	er Type	Plsql Block 🔹				РК Туре	s * VARCHAR2						
	Gateway S	Service	FCUBSFTService		<b>*</b> =			Full Screen	Reply					
Ga	Gateway Ope	eration	QueryContract		<b>*</b> =			HO Only						
Gat	ateway IO Re	equest	Contract-Details-IO		]									
		NOTIFY T	To YIN & Refer Current R			Notify := 'N';	END IF; RETURN TRUE;	•						
er Li	Logic(Set \$	NOTIFY T	_RECORD.module_cod			NOTIFY := 'N';	END IF; RETURN TRUE;	×						
v	Logic(Set \$	NOTIFY T	_RECORD.module_codd		)TIFY := 'Υ'; ELSE \$	NOTIFY := 'N'		×						
v N	Ucgic(Set S IF \$CL Web Servi Order	NOTIFY T	_RECORD.module_codd	e = FT' THEN \$NO	TIFY := 'Y'; ELSE \$ Table Field		Data Type		- Hand State Stat	-				
V 1	Logic(Set S IF \$CL Web Servi Order 1	INOTIFY T	_RECORD.module_codd	e = FT'THEN \$NO	Table Field REF_NO	×E	Data Type VARCHAR2	64						
iter Li	Under Version Servi Web Servi Order 1 2		_RECORD.module_codd	e = FT' THEN \$NO	Table Field 		Data Type							

Fig 4.7.6: Notification Screen Loaded and Modified

# 5 Deploy Notification

#### 5.1 Notification - Workbench related deployment

Compile the following files in Target FLEXCUBE UBS Database schema

- Notification Main Package generated from ODT
- Hook Packages
- GWTM\_NOTIFICATION\_TAG\_MAP\_\_\_<Notification Function ID>\_.INC
- GWTM\_NOTIFICATIONS\_MASTER\_\_\_<Notification Function ID>\_.INC

## 5.2 Notification Trigger deployment

Compile the following files in Target FLEXCUBE UBS Database schema

- GWTM\_NOTIFICATION\_TRIGGERS\_TRIG\_CONTRACT.INC
- GWTR#\_TRIG\_CONTRACT.TRG

# 6 Test Notification

This section explains the run time notification flow and testing steps.

### 6.1 Notification flow

The notification process occurs as two parts:

- 1. Oracle JOBs created using FCJ Scheduler framework that sends data required for notification to an internal JMS queue.
- 2. Gateway MBD that lists on internal JMS queue, that picks the notification XMLs and prepare full web service response and send to external system queues.

### 6.2 Scheduler based notification

The Notification Process in FLEXCUBE can be done using the jobs scheduler as follows:

The trigger generated from Workbench will be inserting key details into a static notification log (STTB\_NOTIFICATION)

Once Job is triggered, a request is sent to EJB layer from job execution class and the notification log table will be polled for unprocessed records.

Each unprocessed record is locked.

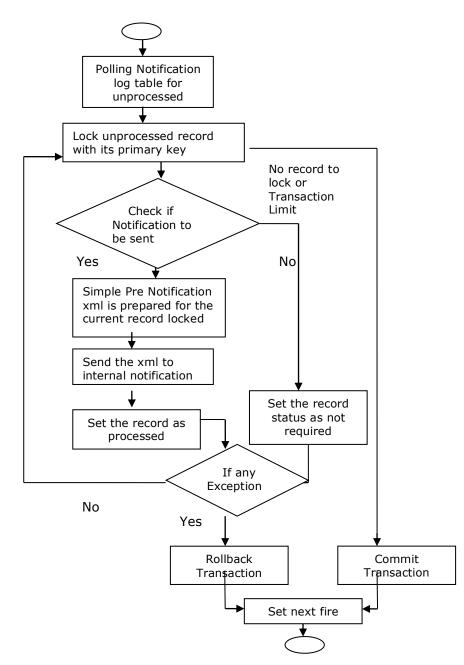
The record is verified against the notification maintenance and checked whether notification is to be sent or not.

If notification is to be sent, pre notification message xml is built and it is sent to internal NOTIFY\_QUEUE(JMS queue) configured in Gateway layer.

The job is then rescheduled to fire next time based on the previous execution.

Refer Gateway Installation documents on how to setup the Queues.

Flow Chart for Notification Flow in Scheduler

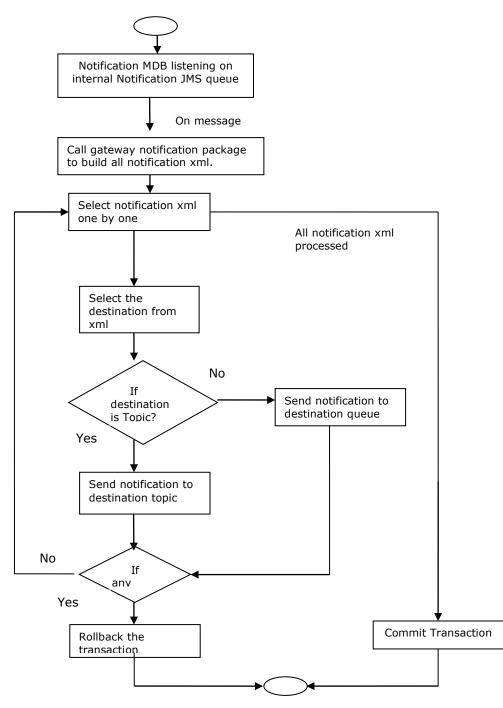


#### 6.3 MDB based notification flow

Notification processes in MDB are as follows:

- 1. Notification MDB listens on the internal NOTIFY\_QUEUE(JMS queue)
- 2. On any message received, the MDB identifies which schema to connect using the JNDI name being present as part of the message xml.
- 3. Gateway notification processing package is called from MDB to build notifications.
- 4. In MDB, the notifications built is processed and sent to the destination specified in corresponding notification.
- 5. In case of exception the transaction is rolled back.
- 6. If all notifications are successfully processed, transaction is committed.

#### Flow Chart for Notification Flow in MDB



#### 6.4 Triggering notification and testing

Follow the below steps to test notification:

- Simulate a case where base table under goes data change.
- Check record populated at STTB\_NOTIFICATION table
- Check Notification message
   GWTBS\_NOTIFICATIONS\_LOG.NOTIFICATION\_MESSAGE



Development Workbench - Notifications [November] [2023] Version 14.7.2.0.0

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, 2023, 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.