Development Workbench - Notifications Oracle FLEXCUBE Investor Servicing Release 14.5.2.0.0 [November] [2021]

FINANCIAL SERVICES



Table of Contents

1	Pref	ace	3
1	1.1	Audience	3
2	Intro	oduction	3
H	low to	o use this Guide	3
3	Noti	fication – Getting started	4
3	3.1	What is Notification	4
3	3.2	Notification Trigger	4
4	Noti	fication Development	4
2	4.1	Pre-request for Notification development and testing	4
2	1.2	Notification specification	5
2	1.3	Notification XML development	
2	1.4	Notification Process	5
2	4.5	Development process in Development Workbench	6
2	1.6	Notification Trigger	6
2	1.7	Notifications 1	.2
5	Dep	loy Notification 1	.9
5	5.1	Notification - Workbench related deployment 1	.9
5	5.2	Notification Trigger deployment 1	.9
6	Test	Notification	20
6	5.1	Notification flow	20
6	5.2	Scheduler based notification 2	20
6	5.3	MDB based notification flow 2	21
6	5.4	Triggering notification and testing 2	23

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

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

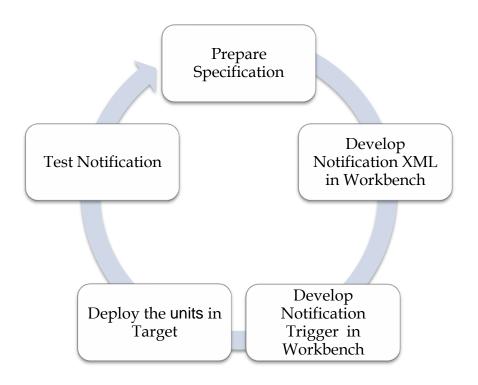


Fig 5.1.1: Development of Notifications

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
 - \circ 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:

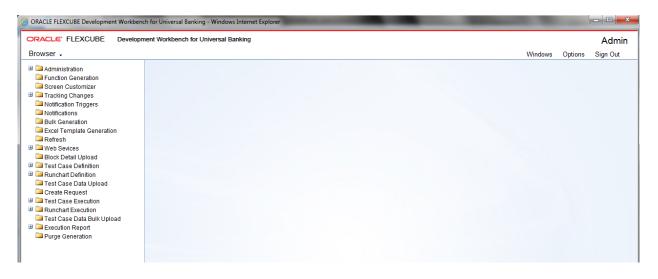
Trigger code: A unique value to for a notification trigger. **Base Table:** The base table on which, the trigger is built. **When Clause:** A simple when clause for the notification trigger.

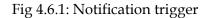
The second step is to capture details of notifications and generate the notification filter procedure. The following details are captured:

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:





Notification Trigger				_ ×
			a 7	9 4
Trigger Co Descripti Firing Tir Each Recc Selected Colum Trigger When Clau	ion PK Cols * me Before * ord Yes * Data Types			
Trigger Logic(Sei	tsNotify To Y/N)			

Fig 4.6.2: Notification trigger options

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

New:

Notification Trigger			_			_ ×
		×	I (I	8 6	1 7	9
	TRG_CUSTADDR Base Table * MSTM_CUST_ADDRESS Trigger for Customer Address Maintenand PK Cols * CUSTOMER_NO-LOCATION-MEDIA After * PK Types * Yes * Data Types					
Selected Columns	(new.auth_stat='A')					
Trigger When Clause	(new.auth_stat='A')					
I_Oper ELSE	DTIFY To YN) Jidonce_auth, N) ⇔ Y THEN aiton := TNSERT; ation := UPDATE;					

Fig 4.6.3: Notification trigger: New option

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

Front-End	Files System Packages	Hook Packages	h	deta Dala			Others	^
] RadXML] Screen Xml] System JS	Main Package Spec Main Package Body Volitication Triggers Upload Package Spec Upload Package Body	Kernel Package Spec Kernel Package Body Cluster Package Spec Cluster Package Body Custom Package Spec Custom Package Body	Menu Defails Datasource Defails LoV Details Block Defails Screen Details Amendable Details Call form Defails Summary Details	Block Funct Gales	I Defailis ; PK Columns Ition Call Forms way Defailis cation Defailis tion Paraméters & Defailis	🛄 Upload 1		
SLNo		File Name		File	Туре		Status	*
	GWTR#_TRG_CUSTADDR.lrg		TR	G			-	
	GWTM_NOTIFICATION_TRIGGERSTRG_0	OUSTADOR INC	INC	;		Generaled	*	
	TRG_CUSTADDR_RAD xml		RAI	DXML			*	
nformation	Error Description	Error Code	<					
! Re	quest successfully 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.

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

Fig 4.6.5: Notification trigger: Modification

cation Trigger			×	1 15	3a I 1	 ••• ×
	Trigger Code	×				
Trigger Code * Description Fining Time Before *	Trigger Code					
Each Record Yes * Selected Columns	Search	Reset				
ingger When Clause		> >>				
Trigger Logic (Sel \$NOTIFY To Y/N)	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					-
			×	I 17	1 F 🌒
				E	xecute Query
				_	
Trigger Code * TRG_CUSTADDR	Base Table '	:			
Description	PK Cols *				
Firing Time Before 👻	PK Types '				
Each Record Yes *	Data Types				
Selected Columns	Notification Codes				
Trigger When Clause					
Trigger Logic(Set \$NOTIFY To Y/N)					
		<u>^</u>			
		-			

Fig 4.6.7: Notification trigger: Modification- Entering values

			_ ×
•	× 1 12	1 7	9 0
Trigger Code * TRG_CUSTADDR Base Table * MSTM_CUST_ADDRESS			
Description Trigger for Customer Address Maintenand P PK Cols * 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 \$NOTIFY To Y/N)			
IF NVL(:old.once_auth, 'N) ⇔ 'Y' THEN			
LOperation := TINSERT; ELSE			
I_Operation := 'UPDATE';			
END IF;			
•			

Fig 4.6.8: Notification trigger: Modification- Entering values

erote Rad Files						
Front-En	f Filen System Packagen	Hook Paokages	N	leta Dala	Others	
] RadXML] Screen Xml] System JS	Main Package Spec Main Package Body Nobliccation Tinggers Upload Package Spec	Kernel Package Spec Kernel Package Body Oluster Package Spec Oluster Package Body Oustom Package Spec Oustom Package Body	Menu Details Datasource Details Datasource Details LOV Details Block Details Screen Details Amendable Details Call form Details Summary Details	Label Details Block PK Columns Function Call Forms Catleway Details Notification Details Function Parameters Purge Details	XSds Xad With Annatalions Screen Himti Upload Table Trigger Upload Tables Ordinition Archive Table Definition	
SLNo		File Name		File Type	Status	^
	GWTR#_TRG_CUSTADDR Irg		TRO	G	Generaled +	
	GWTM_NOTIFICATION_TRIGGERSTRG	CUSTADOR INC	INC	;	Generaled -	
	TRG_CUSTADDR_RAD xml		RAD	DXML	Generaled -	
	Error Description equest successfully Processed	Error Code ARD-SAVE-007				

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 Maintenance							89	- ×
	Notification Function		Action None -	Sa	ave XML Path			
Notification Code * Description Notification Xsd Firing Time Filter Type Gateway Service Gateway Operation Gateway IO Request Type XSD Name Filter Logic(Set SNOTIFY		SCURRENT_RECORD)	Module Des Bas	Module	λ]]]		
Web Service Tag	s			v	+-			
Order	Xsd Field	Table Field	Data Type	Maximum Length				
					÷			

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^{*rd*} *letter must be 'N'. Example: FTNCONON*

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

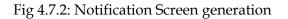
Full screen Reply: This attribute decides whether full screen or primary key notification response to be sent. This is applicable only if gateway Service details are provided

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 Body Cluster Package Body Cluster Package Spec Clustom Package Spec Custom Package Body	Menu Details Datasource Details LOV Details Sick Details Screen Details Amendable Details Call form Details Summary Details	Label Details Block PK Columns Function Call Forms Gateway Details Votification Details Function Parameters Purge Details	 Xsds Xsd With Annotations Screen Himi Upload Table Trigger Upload Tables Definition Archive Table Definition 	
SI.No	F	ïle Name	File Type		Status	
SI.No	F	ile Name	File Type		Status	
SLNo	F	ile Name	File Type		Status	
SLNo	F	ile Name	File Type		Status	
SLNo	F	ile Name	File Type		Status	
SI.No	F	ile Name	File Type		Status	
SLNo	F	ile Name	File Type		Status	Exi



Front-End	iles System Packages	Hook Packages	N	feta Dala	Others
2] RadXML Screen Xml System JC	Main Package Spec Main Package Body Nolification Triggers Upload Package Spec Upload Package Body	Kernel Package Spec Kernel Package Body Cluster Package Spec Cluster Package Sody Cluster Package Spec Custom Package Spec Custom Package Body	Menu Detalls Datasource Defails U/V Details Bilock Details Screen Details Armendable Details Cail form Details Surmary Details	Label Details Block PK Columns Function Call Forms Galeway Details Volification Details Function Parameters Purge Details	Xads Xad With Annotations Screen Himi Upload Table Trager Upload Table Definition Archive Table Definition
SI.No		File Name		File Type	Status
1	fipks_Rnconon_main.spc		SPC	>	Generated
?	fipks_itnconon_kernel.spc		SPC	2	Generated -
3	fipks_itnconon_main.sql Inform	ation		×	Generated 👻
1	fipks_ftnconon_kernel.sql				Generated -
5	GWTM_NOTIFICATIONS_MAST	Error Description	Err	or Code	Generated
5	GWTM_NOTIFICATION_TAG_N	Request successfully Processed	RD	-SAVE-007	Generated 👻
,	FTNCONON_RAD xml				Generated
				-	Ŧ
				Ok	Generate

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

Notificatio	n Maintei	iance																					4 7	×
		N	lolificatio	on Functio	n					Action	n Load	-				Lo	ad Scree	n Xml			B	ROWSE		
N	oblication												Mo	dule [
Upload	Phone	option					×		(2)	Choose Fi	ile to Upk	bad									_ X			
											🍌 « M	AIN 🕨 F	T 🕨 RAE	XML		-	4 7	Search RAI	DXML		9			
		pload Fil	0			Brows	e		0)rganize		ew folder								•	0			
		productin								📜 Dov	vnloads ent Places	*	Name					Date n	modified	1	Туре 📩			
														FTDUP_R					/2012 9:44 /2012 9:44	(MA 8 8 AM	XML F XML F			
								RECOR	Ð	📕 Desktı 🧊 Libr			FTC	PRJFT_RA	D.xml				/2012 9:4		XML F			
L								1		<u> p</u> an	raju	E		CAUTH_R CONAU_F					/2012 9:44 /2012 9:44		XML F XML F			
										P Con	nputer /stem (C:)		🗍 FTD	CONON_	RAD.xml			12/23/	/2012 9:44	IS AM	XML F			
										👝 Da	ata (D:)			CXFRA_R/ DSHBD_R					/2012 9:4		XML F XML F			
											VD RW Dri .EXCUBE_		FTD	мсксн_і	RAD.xml			12/23/	/2012 9:44	IS AM	XML F			
											peria L			MT101_R/				12/23/	/2012 9:4	8 AM)	XML F 🔻			
												File nan	ne:				- A	ll Files (*.*))		-			
																		Open		Cance	1			
W																								
				Xsd Fi	eld		Tab											1						
																			-					

Fig 4.7.4: Notification Screen Loading

															×	V	
			Notification Function F1			Actio	n Load 👻			Save Xml Pa	ath FTNCC	NON_RAI	.xml	BROWSE			
	Notificatio	on Code *	NOTIF_FT_CONTRACT	· · · · · · · · · · · · · · · · · · ·				Module	FT			* E					ĺ
	Des	scription	This is the notification in	ndicating that a FT 🔽	5		Module E	Description	Funds Transfer								
	Notificati	tion Xsd					E	ase Table *	CSTB_CONTR	ACT 🔼		_					
	Firin	ng Time	Insert 💌					PK Cols *	CONTRACT_R	EF_NO							
	Filt	lter Type	PIsql Block 🔹	_	_			PK Types *									
	Gateway		FCUBSFTService						Full Screen	Reply							
Gr	Gateway Op		QueryContract		1				HO Only								
	ateway IO R		Contract-Details-IO														
	Type XSD																
			To Y/N & Refer Current R _RECORD.module_cod			iotify := "N";	END IF; RETURN 1	IRUE;	*								
ilter L	IF \$C	CURRENT.	_RECORD.module_cod			iotify := "N";	END IF; RETURN 1	IRUE;	×								
ilter L	IF \$C	VICE Tags	_RECORD.module_cod		FY := 'Y'; ELSE \$N	IOTIFY := 'N';			×		+-						
V	IF \$C Web Serv Order	VICE Tags	_RECORD.module_cod s Xsd Field	e = FT' THEN SNOTI	Table Field		Data T		Maxim	num Length	+-						
V 1 1 1 1	Web Serv Order		RECORD.module_cod	e = FT' THEN SNOTI	Table Field EF_NO	1	Data T VARCHAR2		Maxin		+						
V	Web Serv Order 1 2		_RECORD.module_cod s Xsd Field	e = FT' THEN SNOTI	Table Field EF_NO		Data T		Maxim								

Fig 4.7.5: Notification Screen Loaded

Noti	cotion Code *	Notification Function								¥ i	¥	9
Noti	antion Code *	Notification Function								<u> </u>		
Noti	action Code *		INCONON	Actio	n Load 👻	Save Xml	Path FTNCC	NON_RAD.xml	BROWSE			
Noti		NOTIF_FT_CONTRACT			Module	e FT		* =				
		This is the notification in			Module Description							
	ification Xsd				Base Table	e * CSTB_CONTRACT		_				
	Firing Time	Insert 👻			PK Cols	s * CONTRACT_REF_NO						
	Filter Type	Plsql Block 🔹			PK Types	s * VARCHAR2						
Gate	way Service	FCUBSFTService				Full Screen Reply						
Gatewa	ay Operation	QueryContract	>			HO Only						
Gateway	IO Request	Contract-Details-IO										
	XSD Name		Record as \$CURRENT_RECORD)									
						-						
	Service Tag					Ŧ	+-					
Or	rder	Xsd Field	Table Field		Data Type	- Maximum Length						
Or	rder		Table Field EXTERNAL_REF_NO	۶E	Data Type VARCHAR2	• Maximum Length						
	rder	Xsd Field		25								

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.

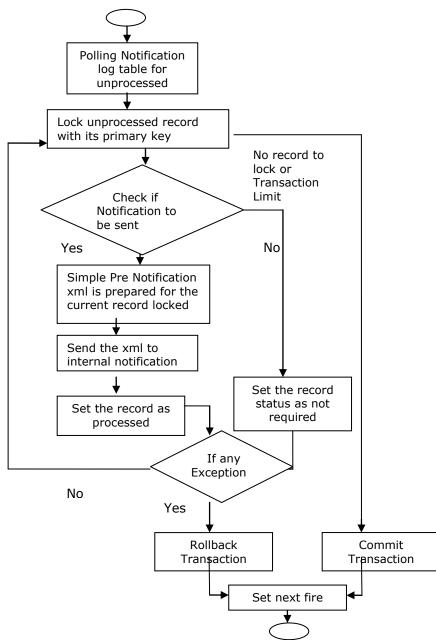
The record is verified against the notification maintenance and checked whether notification is to be sent or not.

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



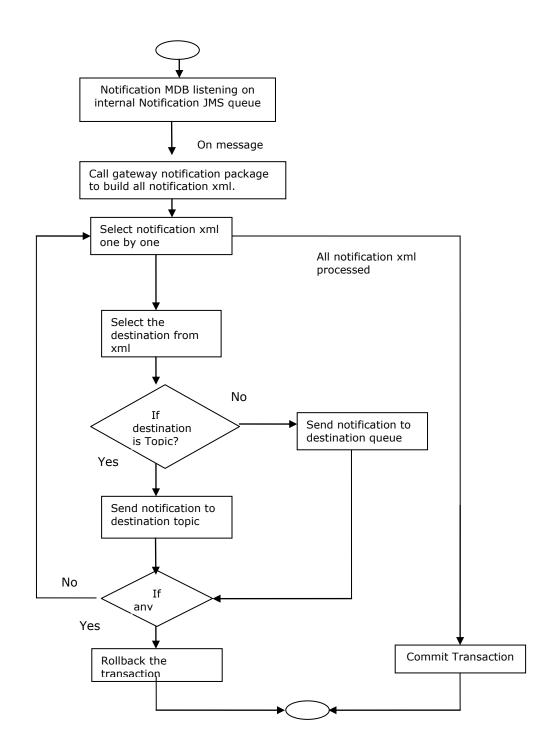
6.3 MDB based notification flow

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] [2021] Version 14.5.2.0.0

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], [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 embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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.