Development Workbench - Notifications Oracle Banking Trade Finance Release 14.5.2.0.0 [August][2021]



# Contents

1	Pref	ace3
-	1.1	Audience3
2	Intro	oduction3
ł	How to	o use this Guide3
3	Noti	fication – Getting started4
	3.1	What is Notification4
	3.2	Notification Trigger4
4	Noti	fication Development4
4	4.1	Pre-request for Notification development and testing4
4	1.2	Notification specification5
4	1.3	Notification XML development5
4	1.4	Notification Process
4	1.5	Development process in Development Workbench5
2	1.6	Notification Trigger
4	1.7	Notifications
5	Dep	loy Notification18
ļ	5.1	Notification - Workbench related deployment18
ļ	5.2	Notification Trigger deployment
6	Test	Notification
(	5.1	Notification flow
(	5.2	Scheduler based notification
(	5.3	MDB based notification flow20
(	5.4	Triggering notification and testing21

## 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 nee	d conceptual and workin	ig 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

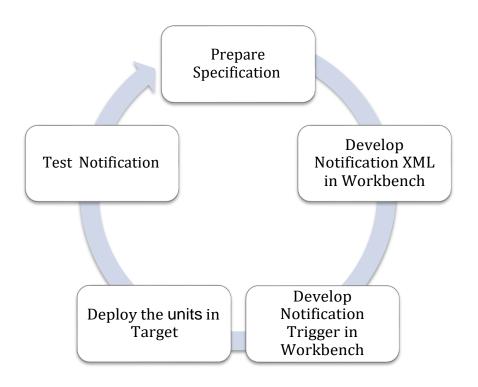


Fig 5.1.1: Development of Notifications

### 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;

0

- 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

5

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:



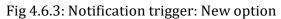
#### Fig 4.6.1: Notification trigger

Notification Trigger					_ ×
					፼ ዀ   ፼ ଭ ⇔
Trigger Code *		Base Table *		×E	
Description		PK Cols *			
Firing Time Each Record	Before V	PK Types * Data Types			
Selected Columns		Notification Codes			
Trigger When Clause	2	Notification obdes			
Trigger Logic( Set \$NC	DTIFY 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:** 

tification Trigger			_ >
Trigger Code + Description Firing Time Each Record Selected Columns Trigger When Clause	TRG_CUSTADDR Trigger for Customer Address Maintenand After v Yes v (new.auth_stat="A)	Base Table * MSTM_CUST_ADDRESS PK Cols * CUSTOMER_INO-LOCATION-MEDIA PK Types * VARCHAR2-VARCHAR2 Data Types Notification Codes	I 🛃 🍃   7 🥡 🤇
I_Ope ELSE	old.once_auth, N') ↔ Ƴ THEN ration := 1NSERT; ration := 'UPDATE';		
		*	



**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 Paskages		Meta Data		1	Dihers	*
RadsML Sorsen Xml System JS	yskall Package Main Package Spac Main Package Body Notikcation Triggers Upload Package Spac Upload Package Body	Kernel Package Spec Kernel Package Body Cluster Package Body Cluster Package Body Clustom Package Body	Menu Defails Datasource Defails LOV Details Brock Defails Screen Details Call form Defails Summary Details	Section Cal	el Defalis K: FK Columns ction Call Forms eway Defalis dication Defalis ction Parameters pe Defalis	Xads Xad With An Screen Him Upload Tabl Upload Tabl	nolations I e Trigger es Definition	
SLNo		File Name		FI	е Туре	Ste	itus	~
1 GW	TR#_TRG_OUSTADOR Irg			TRG				
2 OW	TM_NOTIFICATION_TRIGGERSTRG_C	SUSTADOR INC		INC				
3 TRO	_CUSTADOR_RAD xml			RADXML				
	Error Description	Error Code RD-SAVE-007						
							Generate	- Exit

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:

cation Trigger								-	-	-
						×	I	12	8	ģ
Trigger Code *		<b>*</b> =	Base Table *							
Description		$\sim$	PK Cols *							
Firing Time	Before 👻		PK Types *							
Each Record	Yes 💌		Data Types							
Selected Columns			Notification Codes							
rigger When Clause		$\sim$								
				*						

Fig 4.6.5: Notification trigger: Modification

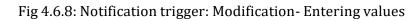
Notification Trigger						×
					<b>X</b>	1112 2017 8
		Trigger Code		×		
Trigger Code *		×5				
Description		Trigger Code				
Firing Time						
Each Record	Yes +					
Selected Columns				Search Reset		
Trigger When Clause						
				<< < 1nf5 > >>		
Trigger Logic( Sel SNC	TIFY TO YAN 3		Trigger Code	<b>^</b>		
		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				
				-		
		L				

Fig 4.6.6: Notification trigger: Modification-Selecting Trigger name

Notification Trigger		_ >
		📔 🗷   🖬 📝 🦕 🖓 🤤 🗧
		Execute Query
Trigger Code *	TRG_CUSTADDR Ase Table *	
Description	PK Cols *	
Firing Time	Before  PK Types *	
Each Record	Yes 💌 Data Types	
Selected Columns	Notification Codes	
Trigger When Clause		
Trigger Logic( Set \$NO		
	×	

Fig 4.6.7: Notification trigger: Modification- Entering values

Trigger Code * [TRG_CUSTADDR Base Table * MSTM_CUST_ADDRESS Description Trigger for Customer Address Maintenand® PK Cols * CUSTOMER_NO-LOCATION-MEDIA Fining Time Address Maintenand® PK Types * VARCHAR2-VARCHAR2-VARCHAR2-VARCHAR2	ification Trigger				_ ×
Each Record Columns Data Types	Trigger Code * [TRG_CUS] Description Trigger for Firing Time After • Each Record Yes • Selected Columns Trigger When Clause (new.auth_ Trigger Logic( Set SNOTIFY To Y/N IF NVL/cold.onc= auth_ L_Operation := TVP] L_SE	Sustomer Address Maintenand stat=A) ) h, N) ⇔ Υ THEN ERT;	PK Cols * PK Types * Data Types	CUSTOMER_NO~LOCATION~MEDIA	- ×



ierate Rad Filos									
Front-End	Files	System Packages	Hook Pa	okages		Meta Dala		Others	ſ
] RadXHL   Screen XmF   System JS		Main Package Spec     Main Package Body     Nolification Triggers     Upload Package Spec     Upload Package Body	Kernel Packag Ciusler Packa Ciusler Packa Ciusler Packa Custom Packa Custom Packa	e Body ge Spac ge Body ge Spac	Menu Delalis Datasource Delalis LOV Details Block Delalis Screen Details Amendable Details Call form Delalis Surmmary Details	Galeway Del	umins Forms alls relaits amelers	Iml	
SLNo			File Name			File Type		Status	,
	GWTR#_	TRG_CUSTADOR Irg				TRG		-	
	GWTM_H	IOTIFICATION_TRIGGERSTRO_C	JSTADOR INC			INC		-	
3	TRG_CL	ISTADDR_RAD xml				RADXML		-	
Information	Error	Description	Error Co	de 🔺					
! Rec	quest succes	sfully Processed	RD-SAVI	E-007					
								Generate	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.

							X I V	
	Notification Function		Action None -	Save XML Path		Unci (		
Notification Code	*		Мо	dule				
Description			Module Descrip	ption				
Notification Xsd			Base T					
Firing Time			PK	Cols *				
Filter Type			PK T)	ypes *				
Gateway Service				Full Screen Reply				
Gateway Operation	1			HO Only				
Gateway IO Request	t							
Type XSD Name	Y To Y/N & Refer Current Record a							
				*				
Web Service Ta	ags			*				
Web Service Ta	ags Xsd Field	Table Field	Data Type	▼ <b>±</b> 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> 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     Custom Package Spec     Custom Package Body     Custom Package Body	Menu Details Datasource Details LOV Details Block Details Amendable Details Call form Details Surmary Details	Label Details Block PK Columns Function Call Forms Gateway Details Violification Details Function Parameters Purge Details	V Xsds Sad With Annotations Screen Hml Upload Table Trigger Upload Tables Definition Archive Table Definition
SI.No		File Name	File Type		Status

Fig 4.7.2: Notification Screen generation

Front-En	d Files S	lystem Packages	Hook Packages		Meta Dala		Others	^
2 RadXML 3 Screen Xml 3 System JS	🗌 Upload I		Kemel Package Spec Kemel Package Body Custer Package Spec Custer Package Spec Custer Package Spec Custom Package Spec Custom Package Body	Menu Delails Datasource Delails LoV Details Block Delails Screen Details Arriendable Details Call form Delails Summary Details	Label Details Block PK Colur Function Call Fr Caleway Details Notification Det Function Param Purge Details	orms 5 ails	Xsds Xsd With Annotations Careen Himi Upload Tables Definition Archive Table Definition	
SLNo			File Name		File Type		Status	*
	floks_Ithconon_main.sp	iC			SPC		Senerated +	
	fipks_itnconon_kernel.s	pc			SPC		Senorated +	
	fipks_itnconon_main.sc	l Informati	DN		×	3	Senerated 💌	
	fipks_ithconon_kernel.s	qi					enerated *	
	GWTM_NOTIFICATIONS	S_MAS1	Error Description		Error Code	[	Senerated +	
	GWTM_NOTIFICATION_	TAG_N !	Request successfully Processed		RD-SAVE-007	[	Senerated 👻	
	FTNCONON_RAD xml				Ŧ		senerated 👻	Ţ
					Ok		Generate	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
Noblication Code Y	Module		<b></b>
Upload File Browse	RECORD	AD.xml         12/23/2012 948.0           xml         12/23/2012 948.0           AD.xml         12/23/2012 948.0           AD.xml         12/23/2012 948.0           AD.xml         12/23/2012 948.0           AD.xml         12/23/2012 948.0           D.xml         12/23/2012 948.0           D.xml         12/23/2012 948.0           D.xml         12/23/2012 948.0           AD.xml         12/23/2012 948.0           AD.xml         12/23/2012 948.0	Type A AM XMLF AM XMLF AM XMLF AM XMLF AM XMLF AM XMLF AM XMLF AM XMLF AM XMLF
Neb Service Tags	le Field Data Type M	dexmum Length	

Fig 4.7.4: Notification Screen Loading

														×	1
			Notification Function FT	NCONON		Actio	n Load 👻			Save Xml F	ath FTNCON	N_RAD.xml	BROWSE		
	Notification	on Code *	NOTIF_FT_CONTRACT					Module	FT						
			This is the notification in		2		Modu		Funds Transfer						
	Notificatio								CSTB_CONTRA	ACT 🔼					
	Firin	ng Time	Insert 👻						CONTRACT_RE						
	Filte	ter Type	PIsql Block 🔹					PK Types *	VARCHAR2						
	Gateway S		FCUBSFTService	×=					Full Screen	Reply					
G	Gateway Ope		QueryContract	×=					HO Only						
Gaf	ateway IO Re	Request	Contract-Details-IO												
		<b>SNOTIFY T</b>	o Y/N & Refer Current R RECORD.module_code			iotify := 'N';	END IF; RETUR	RN TRUE;	*						
Filter L	Logic(Set \$	\$NOTIFY T	_RECORD.module_code			10TIFY := 'N';	END IF; RETUR	RN TRUE;	*						
Filter L	Logic( Set \$	\$NOTIFY T	_RECORD.module_code			IOTIFY := 'N';	END IF; RETUR	RN TRUE;	•		<b>.</b>				
V	Ucgic(Set S IF \$CL Web Servi Order	SNOTIFY T CURRENT	RECORD.module_code	e = 'FT' THEN \$NOTIF	Table Field		Da	ta Type	^ ~ Maxim	ium Length	+-				
V	Logic(Set S IF \$CL Web Servi Order 1	SNOTIFY T CURRENT	_RECORD.module_code		Table Field	24		ta Type	Maxim	num Length					
Filter L	Logic(Set S IF \$CL Web Servi Order 1	SNOTIFY T CURRENT	RECORD.module_code	e = 'FT' THEN \$NOTIF	Table Field EF_NO		Da	ta Type	Maxim	num Length					

Fig 4.7.5: Notification Screen Loaded

								178	×	3 6	4
										/ w	
		Notification Function FT	NCONON	Actic	n Load 🔻	Save Xml Pat	h FTNCONON_RAD.xml	BROWSE	]		
N	Notification Code *	NOTIF_FT_CONTRACT			Module	FT	×1				
	Description	This is the notification in			Module Description	Funds Transfer					
10	Notification Xsd				Base Table	* CSTB_CONTRACT					
	Firing Time	Insert 👻			PK Cols	* CONTRACT_REF_NO					
	Filter Type	Pisqi Block 🔹			PK Types	* VARCHAR2					
C	Gateway Service	FCUBSFTService	*=			Full Screen Reply					
Gat	ateway Operation	QueryContract	*=			HO Only					
Gate	teway IO Request	Contract-Details-IO									
Т	Type XSD Name										
·ilter Lo			lecord as \$CURRENT_RECOP	RD)							
	i soonaen	I_RECORD.module_codi	e = 'FT' THEN \$NOTIFY := Y', E	:LSE \$NOTIFY := 'N';	END IF; RETURN TRUE;	*					
			e = 'FT' THEN \$NOTIFY := 'Y'; E	:LSE \$NOTIFY := 'N';	END IF; RETURN TRUE;	*					
W	Web Service Tag		e = FT THEN \$NOTIFY := Y; E	LSE \$NOTIFY := 'N';	END IF; RETURN TRUE;	*	<u>+ – </u>				
	Web Service Tag Order	15 Xsd Field	Table Fi	ield	Data Type	Maximum Length	+				
	Veb Service Tag Order 1 S	15					+				
	Veb Service Tag Order 1 S	15 Xsd Field	Table Fi	ield	Data Type	Maximum Length	<b>+</b> ]]				

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.

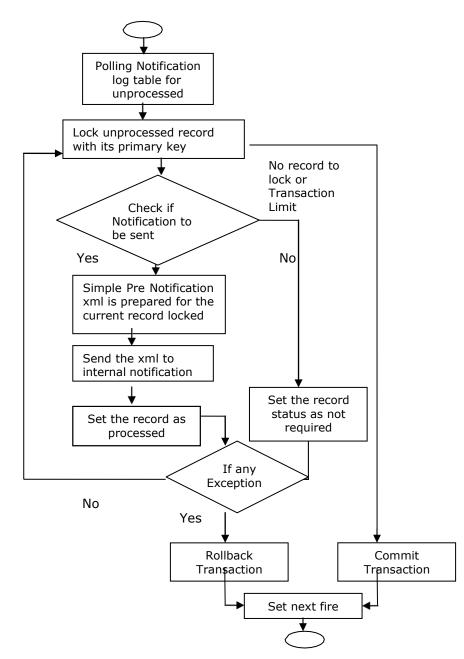
The record is verified against the notification maintenance and checked whether notification is to be sent or not.

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

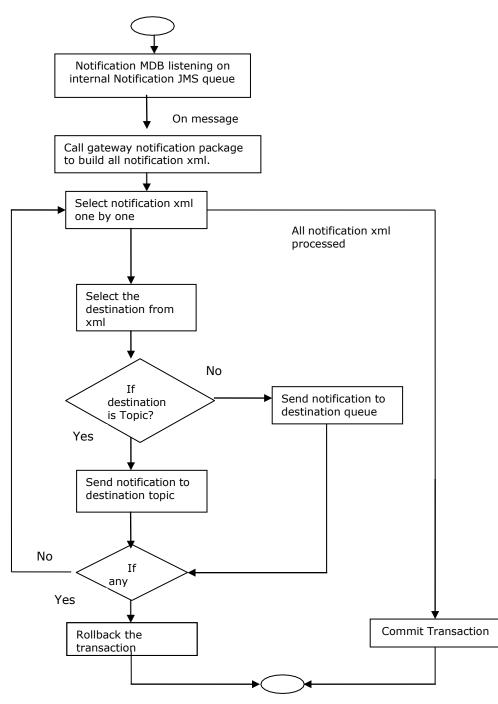


### 6.3 MDB based notification flow

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 [August] [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 © 2021, 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.