Oracle® Banking Corporate Lending

Development Workbench – Notifications

Release 14.8.2.0.0
G53378-01
April 2026

Oracle Banking Corporate Lending Development Workbench – Notifications, Release 14.8.2.0.0

G53378-01

Contents

Preface

This topic contains the following sub-topics:

Purpose

This manual is designed to help acquaint you with the steps to develop the notification XML and notification trigger using Oracle FLEXCUBE Development Workbench for Universal Banking.

Acronyms and Abbreviations

Table 1 Acronyms and Abbreviations

Acronyms	Abbreviations
FCUBS	Oracle FLEXCUBE Universal Banking Solution
OBCL	Oracle Banking Corporate Lending
ODT	Oracle Development Tool

Audience

This document is intended for Oracle FLEXCUBE Universal Banking Application developers/ users that use Development Workbench to develop various Oracle FLEXCUBE Universal Banking components. To use this manual, the user needs a conceptual and working knowledge of the below:

Table 2 Proficiency Details

Proficiency	Resources
Oracle FLEXCUBE Universal Banking Technical Architecture	Training programs from Oracle Financial Software Services.
Working knowledge of Web based applications	Self Acquired
Working knowledge of Oracle Database	Oracle Documentations

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

Conventions

The following text conventions are used in this document:

Table 3 Conventions

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Resources

For more information on any related features, refer to the following documents:

- Development Workbench - Administration
- Development Workbench - Screen Development I
- Development Workbench - Screen Development II

Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes.

1

Notification – Getting started

This topic provides an overview of the Notification framework in FLEXCUBE UBS.

Notification

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

Notification Trigger

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

This topic contains the following sub-topics:

- Notification Development
  This topic provides an overview of the Notification Development process.
- Generate Notification Trigger
  This topic provides systematic instructions to generate new or modify the existing Notification Trigger.
- Create Notifications
  This topic provides systematic instructions to generate new or modify the existing Notification.
- Deploy Notification
  This topic provides systematic instructions to deploy Notification.
- Trigger and Test Notification
  This topic provides systematic instructions to test the notification.

1.1 Notification Development

This topic provides an overview of the Notification Development process.

This topic contains the following sub-topics:

- Prerequisite for Notification development and testing
  This topic provides the detailed information on prerequisites for the Notification Development process.
- Notification specification
  This topic provides a detailed information on Notification Specification.
- Notification XML development
  This topic provides information on Notification XML Development.
- Notification Process
  This topic provides information on Notification Process.
- Development process in Development Workbench
  This topic provides an overview of the Notification Development process.

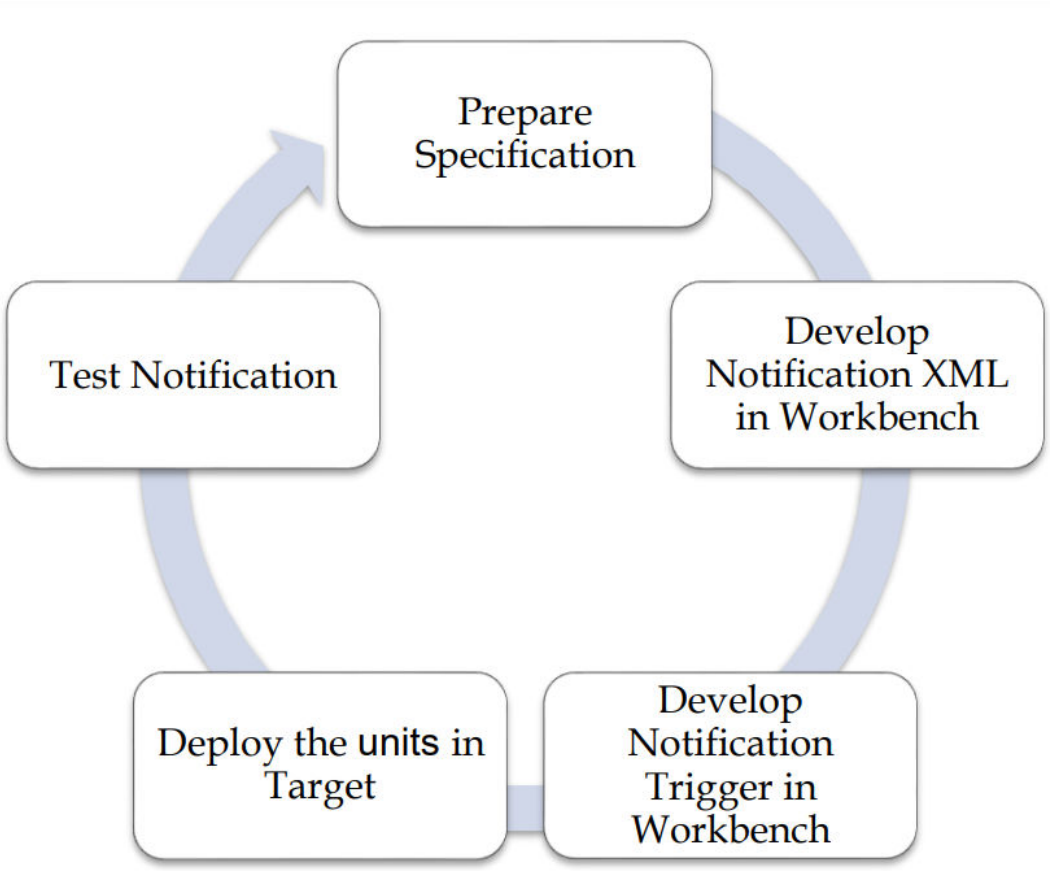
1.1.1 Prerequisite for Notification development and testing

This topic provides the detailed information on prerequisites for the Notification Development process.

Following are the prerequisite 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.

Figure 1-1 Development of Notifications



1.1.2 Notification specification

This topic provides a detailed information on Notification Specification.

Identify the notification requirement as below:

- What is the Notification function ID name for RAD XML (the 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 the filter?
- What is the query Web service to be used?
  - What is the operation?
  - What are the tags required?

For 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

1.1.3 Notification XML development

This topic provides information on Notification XML Development.

Notification RAD XML development creates the following files:

- RAD XML
- SPC
- SQL
- Static Data

1.1.4 Notification Process

This topic provides information on 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.

1.1.5 Development process in Development Workbench

This topic provides an overview of the Notification Development process.

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 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 the 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**

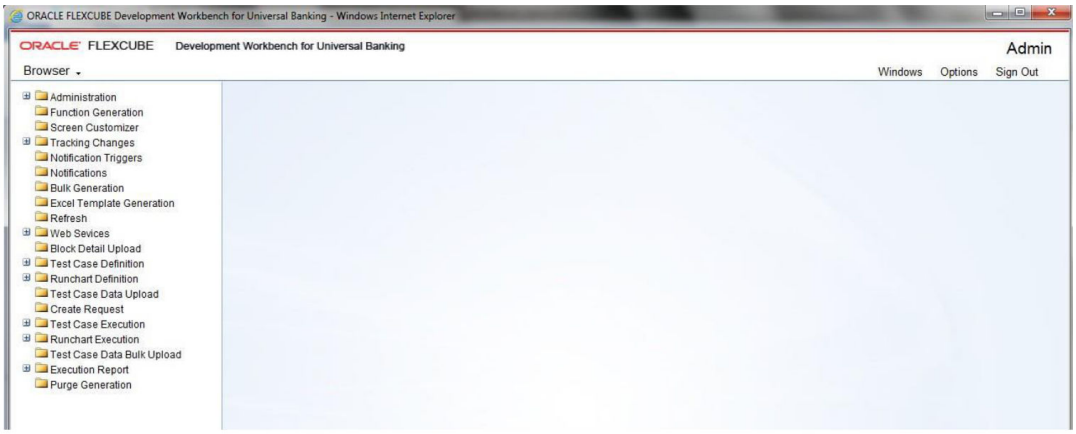
1.2 Generate Notification Trigger

This topic provides systematic instructions to generate new or modify the existing Notification Trigger.

1. Log in to the Development Workbench.

   Development Workbench For Universal Banking screen displays.

Figure 1-2 Development Workbench For Universal Banking



2. Click on the **Notification Triggers** option under the **Browser** menu.
3. To add a new Notification Trigger, follow the steps given below:
   - Specify the details in the **Notification Trigger- New** screen.

Figure 1-3 Notification Trigger - New

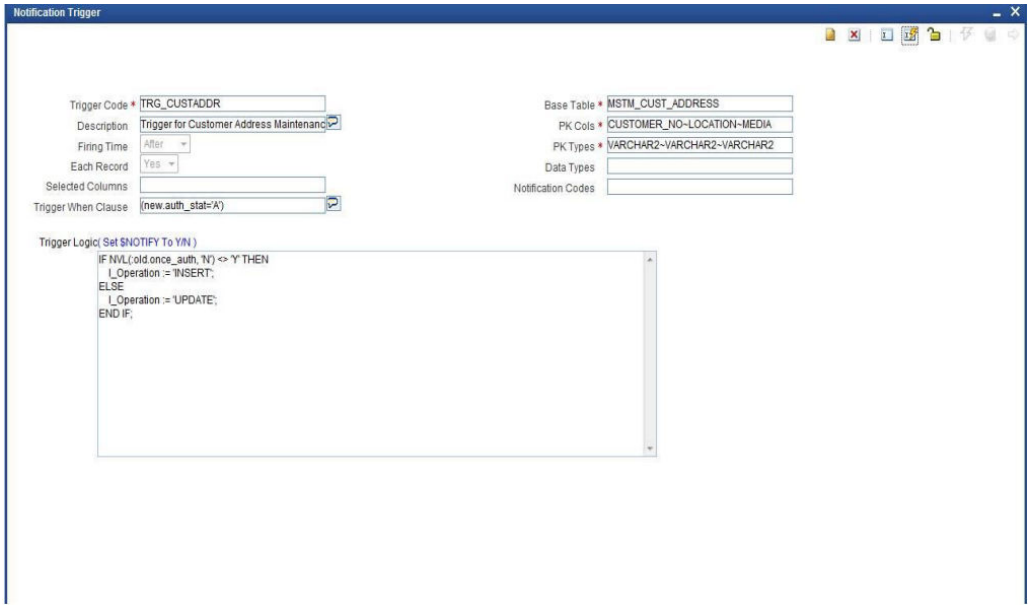


Table 1-1 Notification Trigger- New

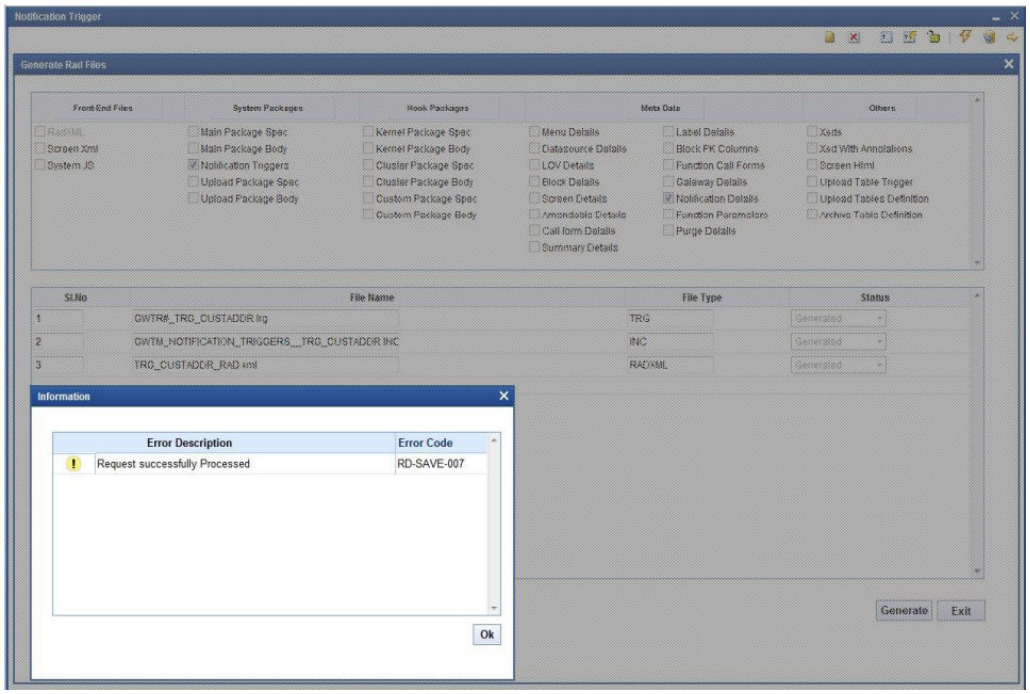
Field	Description
Trigger Code	A unique value for a notification trigger. The naming conversion should start with **TRG_XXXX**. This is a mandatory field. This attribute signifies the trigger code created as part of the trigger creation step in OTD. Each notification will be linked to a trigger code.
Description	This is an information field. Specify a meaningful description of Trigger.
Firing Time	Specify when the trigger needs to be fired. The user 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 a 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 a mandatory field.
Pk Types	Enter the Primary key type of the corresponding primary key field. This is a 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 for Oracle to fire the trigger. This condition cannot contain sub queries. Without the WHEN clause, the trigger is fired for each row.

Table 1-1 (Cont.) Notification Trigger- New

Field	Description
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, the field can be left empty.

An **Information** window displays with confirming the message of successful generation of new Notification Trigger.

Figure 1-4 Information



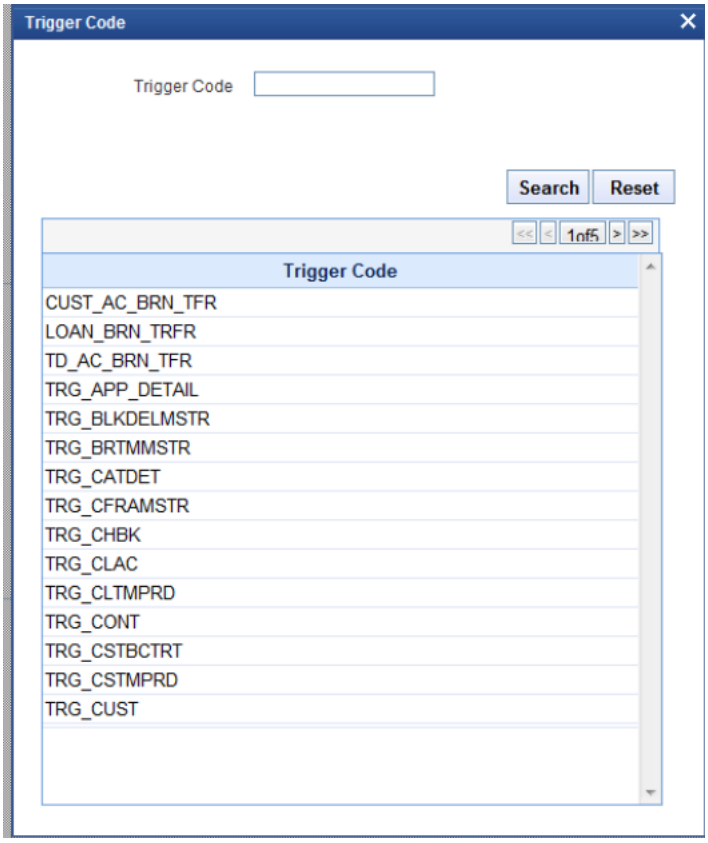
On successful save, Notification Trigger generates two files (**gwtr#_<trg-code>.trg** and **GWTM_NOTIFICATION_TRIGGERS__<trg-code>.INC**) that user needs to compile them in FLEXCUBE schema.

4. To modify the existing Notification Trigger, follow the steps given below:

   a. In **Notification Trigger** screen, click on the browse icon at **Trigger Code** option.

   Trigger Code window displays.

Figure 1-5 Trigger Code

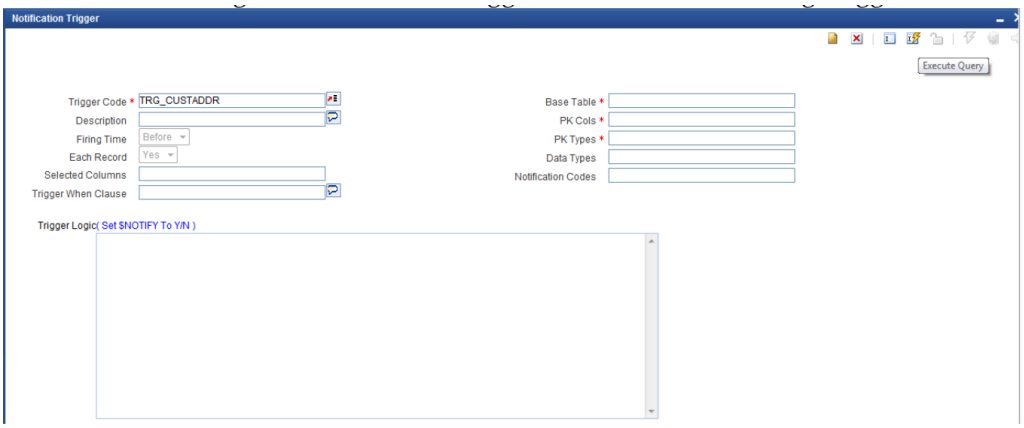


b. Enter the trigger to be modified in the **Trigger code** field and click on **Search** button.

c. Select the **Trigger Code** from the list.

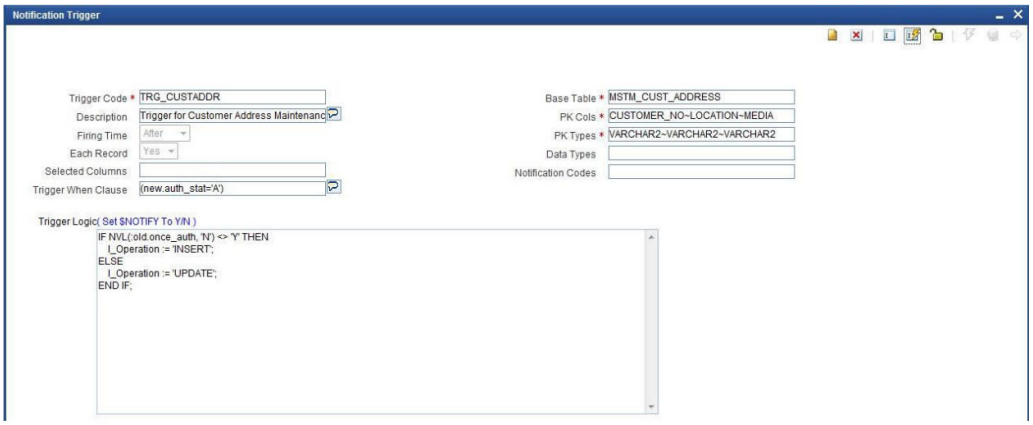
Notification Trigger screen displays with the selected **Trigger code**.

Figure 1-6 Notification Trigger - Modify values



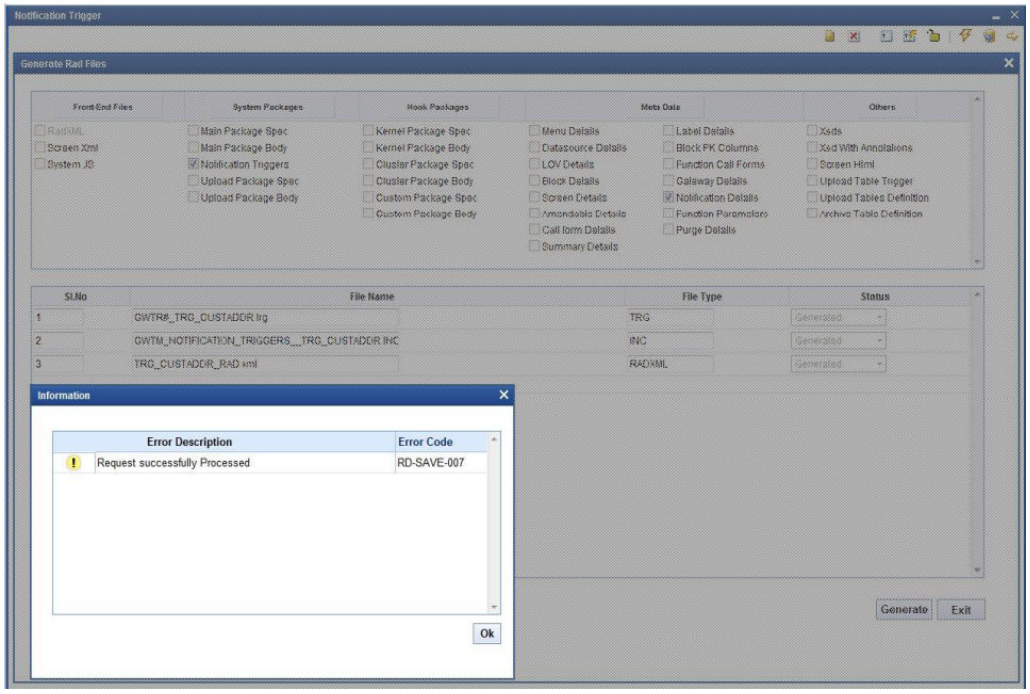
d. Specify the new details to modify the existing one and save the Notification Trigger.

Figure 1-7 Notification Trigger- Modify



An **Information** window displays with confirming the message of successful modification of the Notification Trigger.

Figure 1-8 Information



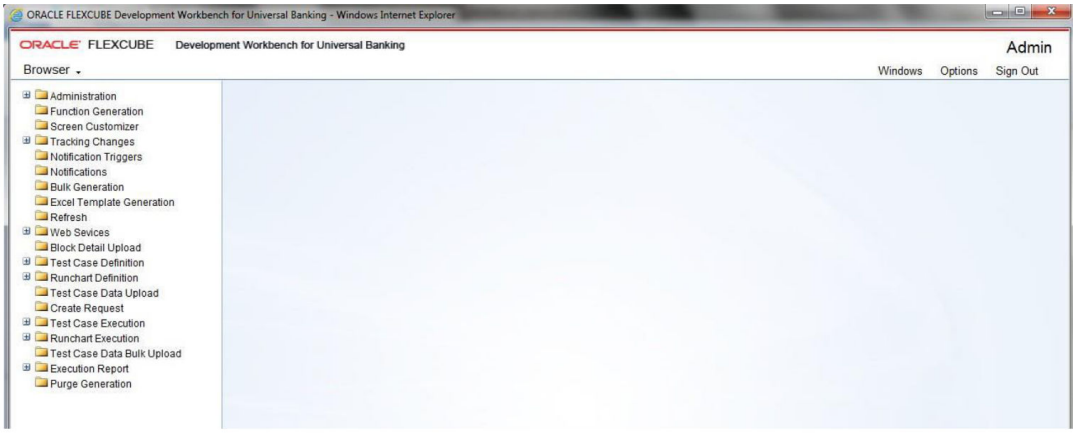
1.3 Create Notifications

This topic provides systematic instructions to generate new or modify the existing Notification.

1. Log in to the Development Workbench.

   Development Workbench For Universal Banking screen displays.

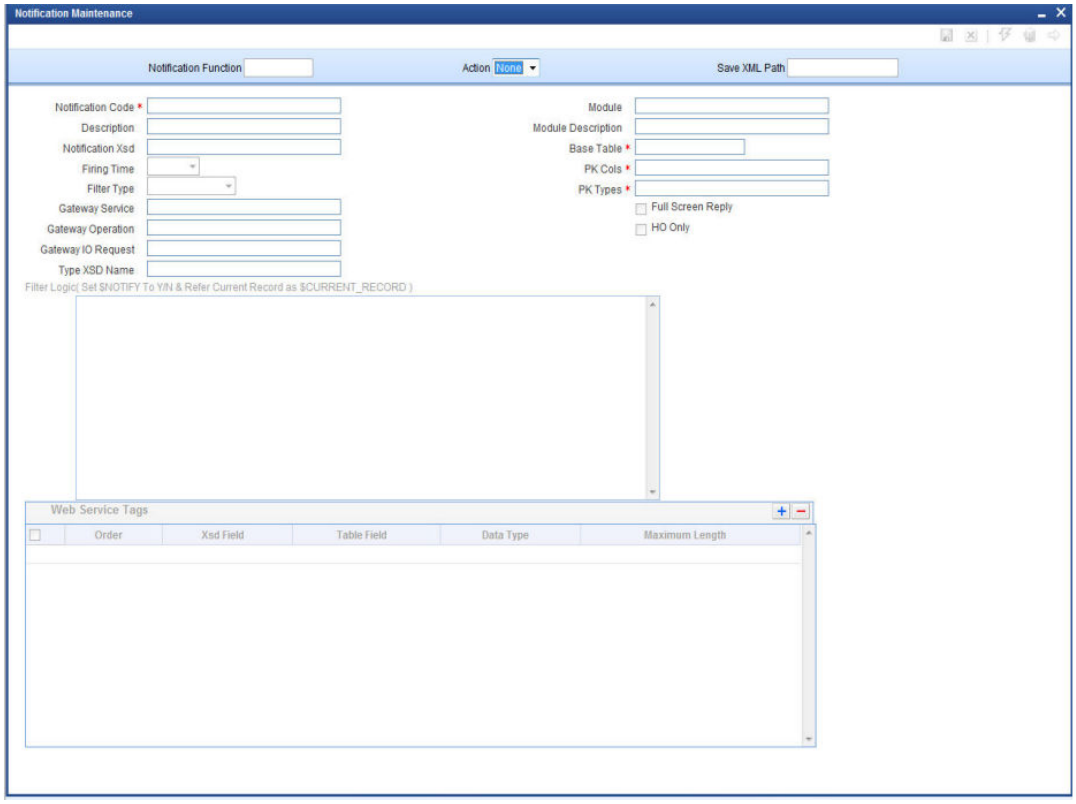
Figure 1-9 Development Workbench For Universal Banking



2. Click on the **Notifications** option under the **Browser** menu.

   The **Notification Maintenance** screen displays.

Figure 1-10 Notification Maintenance



Notification Maintenance screen is used to create a new notification or modify an existing notification, here notification information gets captured for notification codes and the user needs to save notification details into XML.

3. To create a new Notification, follow the steps given below:

   a. Specify the details in the **Notification Maintenance** screen and save it.

Table 1-2 Notification Maintenance- Field Description

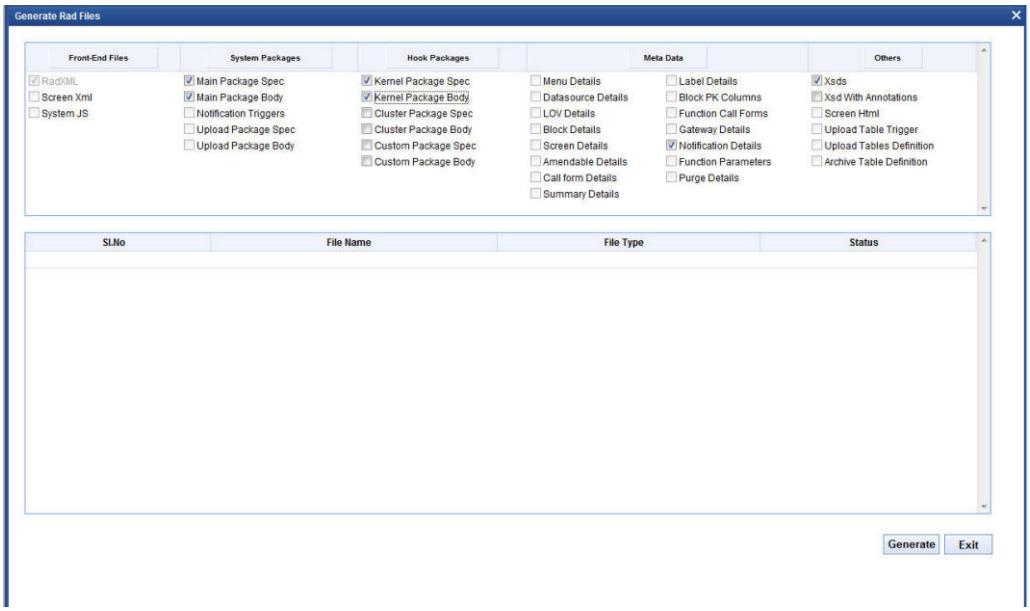
Field	Description
Action	Select either **New** or **Load** action. The **New** option is to create a new notification and the **Load** option is used to modify the existing one.
Save XML Path	Specify the path to save notification XML. This would be considered only if the Save Mode is Client and Work Directory specified as **$CURRENT_DIRECTORY**.
Notification Function	Specify the notification function-id name. **Note**: Naming Conventions- Notification Function must have maximum 8 characters and the 3rd letter must be **N**. For Example: **FTNCONON**
Notification Code	Enter the notification code to which we need to capture values. This is a mandatory field. **Note**: Recommended Convention for Notification Codes- NOTIF_<Module Code>_<Description> For Example: **NOTIF_LD_CONTRACT**, This is the notification indicating that an LD contract has been created/modified.
Description	Information field. A 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 that would be defaulted from Module LOV.
Notification XSD	Notification XSD name will have to be provided in the corresponding Field. **Note**The naming convention to be followed while naming Notification XSD is as follows- [Module Name] – [Notification Description] – Notif.xsd. For 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 have to be sent. Options available are **Insert**, **Update**, or both.
Filter Type	This attribute can take the following values. • **Where Clause** • **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 the query for the mentioned Service.
Gateway Service	The gateway service is to be used to get the full-screen response.
Gateway IO Request	The gateway IO request node to be used in querying operations.
Type XSD Name	This field has to be entered if Notification is mapped to a Service and Request. The name of the Master Type XSD for the service and operation has to be provided here. This can be found in the included portion of the Request Msg XSD of particular Service-Operation. For Example: **LC-Contract-Types.xsd**

Table 1-2 (Cont.) Notification Maintenance- Field Description

Field	Description
Full screen Reply	This attribute decides whether a full screen or primary key notification response is to be sent. This is applicable only if gateway Service details are provided.
HO only	This attribute is used to send notifications only from the head office.
Filter Logic	The filter logic decides whether the notification needs to be sent or not. This can be a simple where-clause on the base table or a complex pl/sql block.
Web service Tags	The columns selected from the base table as part of web service tags will be used to send the full-screen notification response. These tags define the elements of Notification XML when no Query service is mapped to it.

The **Generate Rad Files** screen displays.

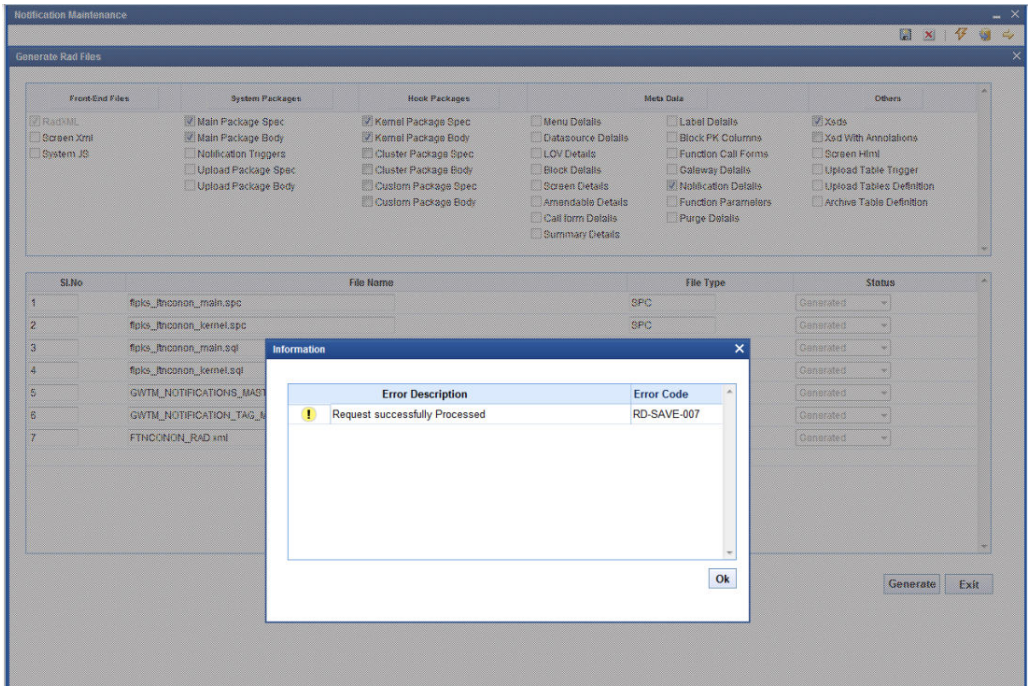
Figure 1-11 Generate Rad Files



b. Click on the **Generate** button.

An **Information** window displays confirming the successful generation of Notification.

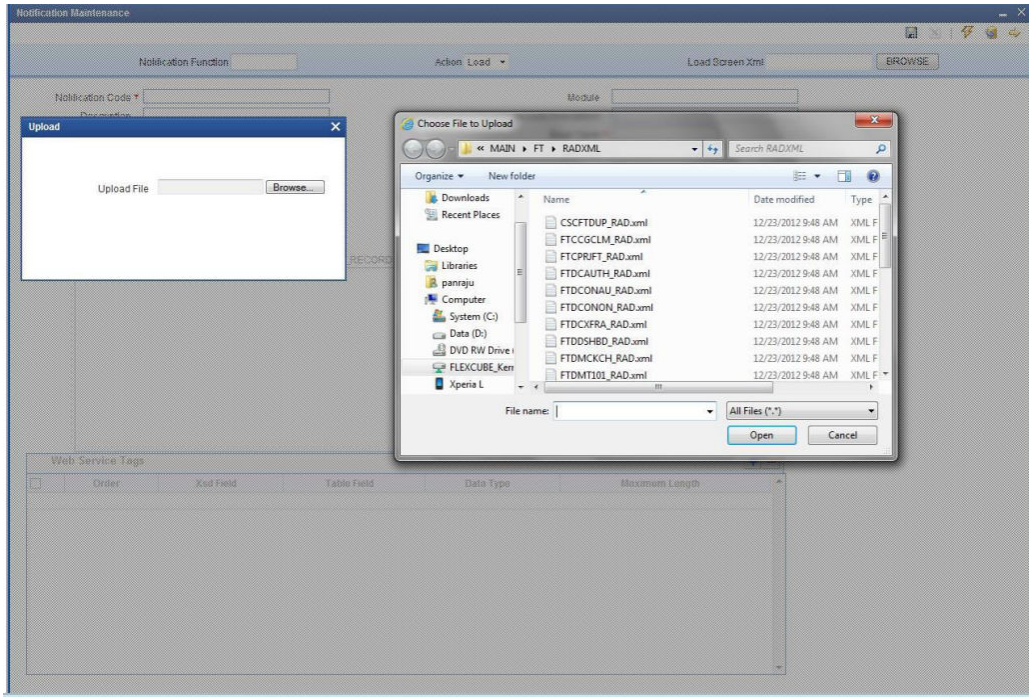
Figure 1-12 Information



4. To modify the existing Notification RADXML, follow the steps given below:
   a. In **Notification Maintenance** screen, select the **Action** field as **Load**.
   b. Click on the **BROWSE** button at **Load Screen XML** field.

      The **Choose File to Upload** folder window displays.

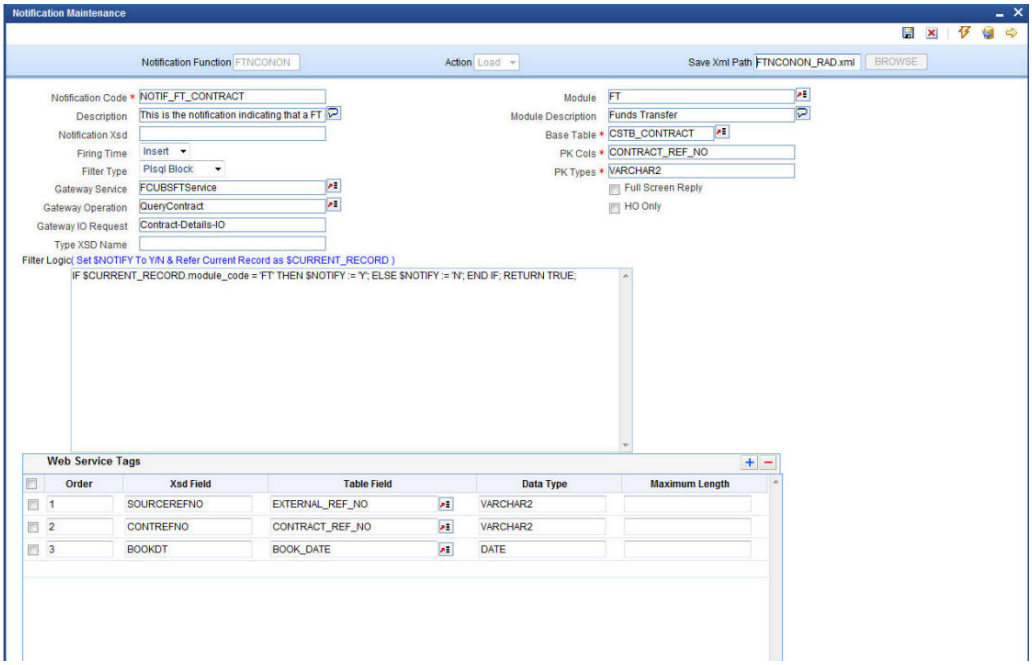
Figure 1-13 Choose File to Upload



c. Select the RADXML file to be modified and click on the **Open** button.

The **Notification Maintenance** screen displays with selected RADXML file details.

Figure 1-14 Notification Maintenance



d. Modify the details as required and click on the **Save** icon.

Selected RADXML file gets modified.

1.4 Deploy Notification

This topic provides systematic instructions to deploy Notification.

1. For 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
2. For Notification Trigger deployment, compile the following files in Target FLEXCUBE UBS Database schema:
   - GWTM_NOTIFICATION_TRIGGERS__TRIG_CONTRACT.INC
   - GWTR#_TRIG_CONTRACT.TRG

1.5 Trigger and Test Notification

This topic provides systematic instructions to test the notification.

1. Simulate a case where the base table undergoes data change.

2. Check record populated at **STTB_NOTIFICATION** table
3. Check Notification message:
   GWTBS_NOTIFICATIONS_LOG.NOTIFICATION_MESSAGE

This topic contains the following sub-topics:

- Notification Flow
  This topic describes the run time Notification flow.
- Scheduler Based Notification Flow
  This topic describes the Scheduler Based Notification Flow.
- MDB Based Notification Flow
  This topic describes about MDB Based Notification flow.

1.5.1 Notification Flow

This topic describes the run time Notification flow.

The notification process occurs in 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 prepares full web service response and send to external system queues.

Scheduler Based Notification

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

1. The trigger generated from Workbench will be inserting key details into a static notification log (STTB_NOTIFICATION).
2. Once Job is triggered, a request is sent to the EJB layer from the job execution class and the notification log table will be polled for unprocessed records.
3. Each unprocessed record is locked. The record is verified against the notification maintenance and checked whether notification is to be sent or not.
4. 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.
5. The job is then rescheduled to fire next time based on the previous execution.

Refer Gateway Installation documents on how to set up the Queues.

Figure 1-15 Architecture- Notification Flow in Scheduler

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 are processed and sent to the destination specified in the corresponding notification.

5. In case of an exception the transaction is rolled back.
6. If all notifications are successfully processed, the transaction is committed.

1.5.2 Scheduler Based Notification Flow

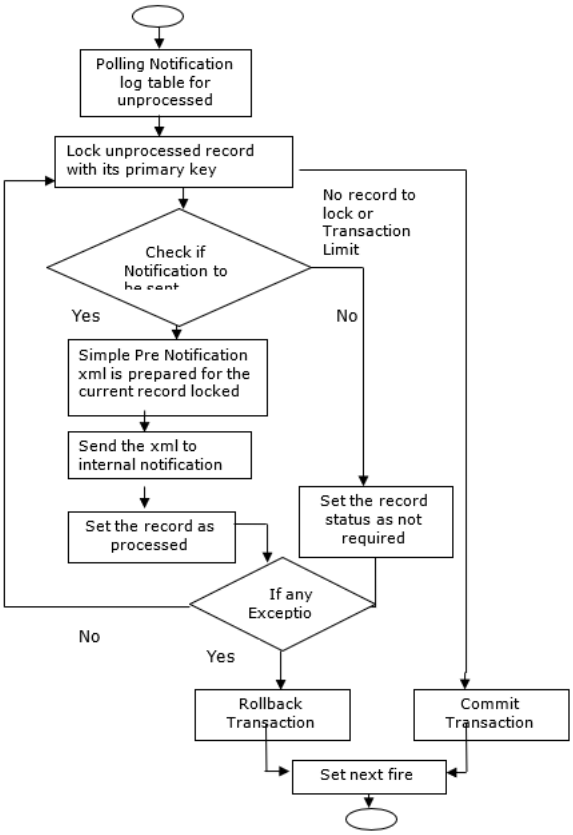
This topic describes the Scheduler Based Notification Flow.

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

1. The trigger generated from Workbench will be inserting key details into a static notification log (STTB_NOTIFICATION).
2. Once Job is triggered, a request is sent to the EJB layer from the job execution class and the notification log table will be polled for unprocessed records.
3. Each unprocessed record is locked. The record is verified against the notification maintenance and checked whether notification is to be sent or not.
4. 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.
5. The job is then rescheduled to fire next time based on the previous execution.

Refer Gateway Installation documents on how to set up the Queues.

Figure 1-16 Flow Chart for Notification Flow in Scheduler

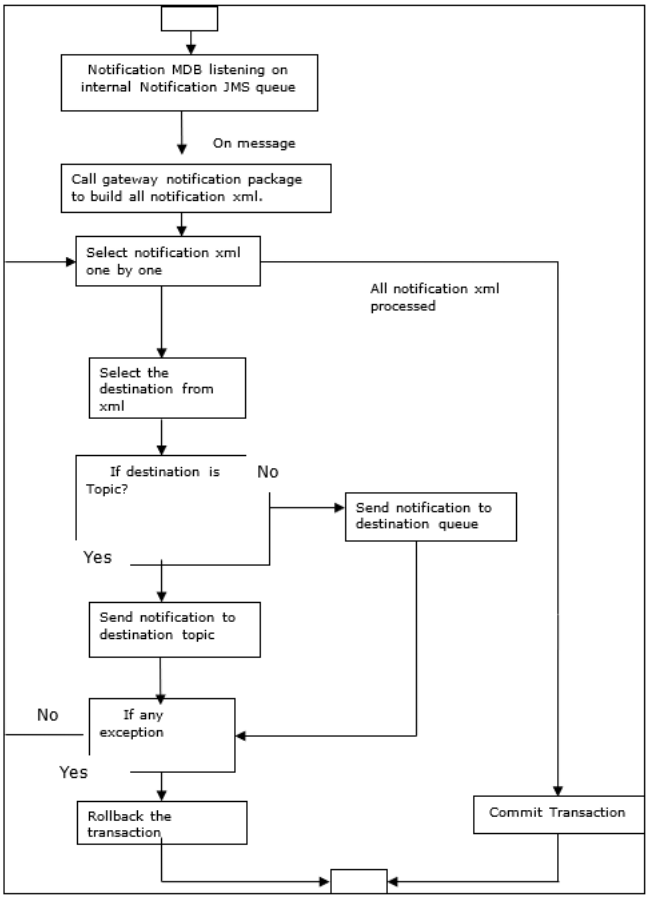


1.5.3 MDB Based Notification Flow

This topic describes about MDB Based Notification flow.

Notification processes in MDB are as follows:

Figure 1-17 MDB Based Notification Flow



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 are processed and sent to the destination specified in the corresponding notification.
5. In case of an exception the transaction is rolled back.
6. If all notifications are successfully processed, the transaction is committed.