

Oracle® Banking APIs

Event Management Guide



Release 25.2.0.0.0

G60584-01

August 2026

ORACLE®

Copyright © 2006, 2026, Oracle and/or its affiliates.

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.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

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," "commercial computer software documentation," or "limited rights data" 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 fail-safe, 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.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

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.

Contents

Preface

Purpose	i
Audience	i
Documentation Accessibility	i
Critical Patches	i
Diversity and Inclusion	ii
Conventions	ii
Related Resources	ii
Screenshot Disclaimer	ii
Acronyms and Abbreviations	ii

1 Introduction

2 Database Configurations

3 Code Configuration

4 Event Processing

5 Custom Fields For Push notifications

6 Multi-Entity Specific templates

7 WhatsApp Configurations

Index

List of Figures

7-1	<u>WhatsApp Quickstart</u>	<u>1</u>
7-2	<u>WhatsApp Quickstart- API Setup</u>	<u>1</u>
7-3	<u>Service Consumers</u>	<u>2</u>
7-4	<u>Service Consumers - Transformation</u>	<u>2</u>
7-5	<u>Map the service and verify the request transformation template</u>	<u>3</u>

Preface

- [Purpose](#)
- [Audience](#)
- [Documentation Accessibility](#)
- [Critical Patches](#)
- [Diversity and Inclusion](#)
- [Conventions](#)
- [Related Resources](#)
- [Screenshot Disclaimer](#)
- [Acronyms and Abbreviations](#)

Purpose

This guide is designed to help acquaint you with the Oracle Banking application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

Audience

This document is intended for the following audience:

- Customers
- Partners

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

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.

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](#).

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.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	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.

Related Resources

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

- Oracle Banking APIs Installation Manuals

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.

Acronyms and Abbreviations

The list of the acronyms and abbreviations used in this guide are as follows:

Table 1 Acronyms and Abbreviations

Abbreviation	Description
OBAPI	Oracle Banking APIs

1

Introduction

This document contains steps to configure alerts for any event in the OBDX application. An alert configuration is identified by following properties:

1. **Event Group:** It groups similar events of similar functionalities together. An Event Group may contain multiple events.
2. **Event:** An event could be any action taken by the user or system in OBDX application that triggers alert. Each event maps to a corresponding action or an activity executed in the business logic in OBDX. OBDX application may receive events from external system which are responsible for triggering alert.
3. **Message template:** This is a template of the message that needs to be sent as an alert to the receiver. An event can have multiple templated based on the channel on which it is getting delivered.
4. **Message Destination:** Destination is the channel on which alert/ notification will be delivered to the user. OBDX supports 5 such channels – Email, SMS, WhatsApp, Push Notification and On-screen notification.
5. **Message Attribute:** A message template contains the message to be delivered to the receiver. The message is relevant to the action being performed in the specific event and it contains dynamic data values from the business logic. A message attribute is an identifier, which is used in the message template to replace with the actual values dynamically.
6. **Message Action:** Some of the events in OBDX contains an actionable link in its content. This link is responsible for the navigation of user to desired location from the alert. Message action attribute defines the action to be executed during event processing.

This document also contains the business logic required to invoke an event for triggering alert within OBDX and other miscellaneous aspects.

2

Database Configurations

1. DIGX_EM_EVENT_GROUP

This table contains the available event group entries in OBDX. One event group may have multiple events. An event group can be created as per the requirements.

COLUMN NAME	DESCRIPTION
ID	A unique identifier for the event group.
NAME	Name of the event group.
DESCRIPTION	Description of the event group.
IS_DND_APPLICABLE	Identifies whether the DND setting is applicable for this event group or not. Possible values: 'Y' or 'N'.

2. DIGX_EM_EVENT

The events are added in the **DIGX_EM_EVENT** table.

COLUMN NAME	DESCRIPTION
ID	A unique identifier for the event occurred. It should be a logical name for the event.
NAME	Name of the event.
DESCRIPTION	Description of the event.
EVENT_TYPE	Identifies if the event is Mandatory or Subscribe-able for the user. Possible values are: 'M' or 'S'.
EVENT_GROUP_ID	Specifies the Group id of the event.
TASK_ID	This column is used for subscription-based alerts. If account access needs to be checked for an alert before sending it to receiver, this task id will be used to check account access.

3. DIGX_EM_MESSAGE_ATTRIBUTE

Message attributes are added in the table **DIGX_EM_MESSAGE_ATTRIBUTE** table.

COLUMN NAME	DESCRIPTION
NAME	Name of the attribute. This needs to be used in the message template where the dynamic value needs to be replaced.
DESCRIPTION	Description of the attribute.
EVENT_ID	ID of the Event. It should match ID column of DIGX_EM_EVENT
DATATYPE	It determines the type of data. Example – String, Date, Currency, Complex and Number.
PATH	It specifies the path of message template.

4. DIGX_EM_MESSAGE_ACTION

Message Actions are added in **DIGX_EM_MESSAGE_ACTION** table. This is only for those events that are actionable and contains URL.

COLUMN NAME	DESCRIPTION
NAME	Name of the action. This needs to be used in the template where the link needs to be replaced with
DESCRIPTION	Description of the action.
EVENT_ID	ID of the Event. It should match ID column of DIGX_EM_EVENT
URL_TEMPLATE	URL of the event. This is the actual URL/ link where the user will be redirected to.
URL_TEXT	This is the text that will be displayed in the alert received by the user.
LOGIN_REQUIRED	Identifies if login is required or not. If the redirection page is restricted, it should ask for login page. Possible values: 'Y' or 'N'.

5. DIGX_EM_MESSAGE_TEMPLATE

Message templates are added to the table **DIGX_EM_MESSAGE_TEMPLATE** table.

COLUMN NAME	DESCRIPTION
NAME	Name of the message template.
TITLE	Title of the message template. This is the subject on the alert in case of email and on-screen message.
CONTENT	It contains the format for the message body. It is stored as CLOB in the table.
LOCALE	The locale column stores language and regional preferences, typically represented by language codes like "en" for English.
DETERMINANT_VALUE	It determines the entity code for the template.
DELETE_STATUS	Identifies the delete status of message template. Possible values are: 'Y' or 'N'.
EVENT_ID	ID of the Event. It should match ID column of DIGX_EM_EVENT .
DESCRIPTION	Description of the message template.

Note

While defining the content of the message template, the attribute name and the message action name needs to have # as prefix and suffix.

Example: If the attribute name is payeeName, the message content would be, "You have added #payeeName# as a beneficiary for payment."

6. DIGX_EM_MESSAGE_TEMPLATE_DESTINATION_REL

A message template needs to be mapped to the desired destinations to which alert needs to be delivered.

COLUMN NAME	DESCRIPTION
TEMPLATE_NAME	Name of the message template. It should match the column NAME of DIGX_EM_MESSAGE_TEMPLATE .
LOCALE	The locale column stores language and regional preferences, typically represented by language codes like "en" for English. It should match the column LOCALE of DIGX_EM_MESSAGE_TEMPLATE .
DETERMINANT_VALUE	It determines the entity code for the template. It should match the column DETERMINANT_VALUE of DIGX_EM_MESSAGE_TEMPLATE .
DESTINATION_ID	Determines the channel on which alert is to be sent. Possible values are SMS , WA , SMB , EM and PN .

7. DIGX_EM_DESTINATION

Destinations are added in **DIGX_EM_DESTINATION** table. In the application five such destinations are already present which are Email(EM), SMS(SMS), Push Notification(PN), Secure Mailbox(SMB) and WhatsApp(WA).

COLUMN NAME	DESCRIPTION
ID	Primary key of the table. An identifier for the destination.
NAME	Name of the destination.
DESCRIPTION	Description of the destination.

Note

Entries for most of the event groups, events, message attributes, message action, message templates and message template destination relation are already added. Please check for the entries in the table to avoid repetition.

Sample Scripts

- insert into DIGX_EM_EVENT_GROUP (ID,NAME,DESCRIPTION,IS_DND_APPLICABLE)
values ('SMS', ' User Management', 'Event group for user management', 'N');
- insert into DIGX_EM_EVENT
(ID,NAME,DESCRIPTION,EVENT_TYPE,EVENT_GROUP_ID,TASK_ID)
values ('USER_LOGIN_SUCCESS', 'Login success Alert', 'Login success
Alert', 'M', 'SMS', null);
- insert into DIGX_EM_MESSAGE_ATTRIBUTE
(NAME,DESCRIPTION,EVENT_ID,DATATYPE,PATH)
values ('BankName', 'Bank Name For Login Success Alert',
'USER_LOGIN_SUCCESS', 'String', 'bankName')

- ```
insert into DIGX_EM_MESSAGE_ACTION
(NAME,DESCRIPTION,EVENT_ID,URL_TEMPLATE,URL_TEXT,LOGIN_REQUIRED)
values ('act1', 'Url Template for Approval of Non-Financial Transaction',
'com.ofss.digx.app.approval.service.transaction.Transaction.checkApprovals.
nonfinancial_TRANSACTION_INITIATED_APPROVER',
'home.html?homeModule=approvals&homeComponent=transaction-
detail¶ms={"apiType":"#ApiType#","transactionId":"#TxnReferenceNo#"}',
'click here', 'Y');
```
- ```
insert into DIGX_EM_MESSAGE_TEMPLATE
(NAME,TITLE,CONTENT,LOCALE,DETERMINANT_VALUE,DELETE_STATUS,EVENT_ID,LAST_UP
DATED_BY,
LAST_UPDATED_DATE,DESCRIPTION) values ('USER_LOGIN_SUCCESS_SHORT', 'Login
Success Alert.', 'You have successfully logged in
to your internet banking on #loginSuccessDateAndTime#. If you do not
recognize this login attempt, immediately contact customer
care/branch.', 'en', '*', 'N', 'USER_LOGIN_SUCCESS', 'OBXUser',sysdate,
'Login success Alert Short Template');
```
- ```
insert into DIGX_EM_MESSAGE_TEMPLATE_DESTINATION_REL (TEMPLATE_NAME,
LOCALE,DETERMINANT_VALUE,DESTINATION_ID)
values ('USER_LOGIN_SUCCESS_SHORT', 'en', '*', 'SMS');
```
- ```
insert into DIGX_EM_MESSAGE_TEMPLATE_DESTINATION_REL
(TEMPLATE_NAME,LOCALE,DETERMINANT_VALUE,DESTINATION_ID)
values ('USER_LOGIN_SUCCESS_SHORT', 'en', '*', 'PN');
```
- ```
insert into DIGX_EM_DESTINATION (ID, NAME, DESCRIPTION)
values ('SMS', 'SMS', 'Destination for sending messages via SMS');
```

# 3

## Code Configuration

### API for Raising an Event:

For raising an event, generateEvent API has been provided in the **AbstractApplication** class. A developer needs to call this API to generate an alert that is required for the respective business logic

It takes three parameters:

1. Session Context
2. ActivityLog: This object contains the dynamic data that needs to be replace in place of attributes in the message template content.
3. EventId

Typically, an event is triggered from service after the business logic has been performed. While triggering alert API event id and the parameters that needs to be passed should be determined and defined to the database tables mentioned above.

For an account-based alert of type mandatory or subscription, accountId and accountType attribute should be populated in ActivityLog. Similarly for a Party-based alert, customerId attribute should be populated in the ActivityLog.

In the application, Activity log contains some basic fields that can be used as attributes for the alert processing. In case, additional fields are required in the message content a sub class of ActivityLog should be created and used and passed as an argument to "generateEvent" API.

Following is a sample code that can be used in the business logic to generate and event and trigger alert.

```
ActivityLog activityLog = new ActivityLog();

activityLog.setCustomerId(sessionContext.getTransactingPartyCode());

 activityLog.setAccountId("<<AccountNumber>>");
 activityLog.setAccountType("<<AccountType>>");
 //If required, set other attributes in activityLog
 super.generateEvent(sessionContext, activityLog, <<EventId>>);
```

### Alert generated using schedulers, listener classes

An alert can be generated from a non-standard REST API of OBDX application. It can be invoked in a scheduler class, a listener class or it can also be invoked from a runnable thread invoked within a service. In such cases, ensure that following code is executed at the end of the business logic. Possibly within a finally block of the code, through which generate event has been triggered.

```
EmHandler.getInstance().putMessage();
```

# 4

## Event Processing

Event processing in OBDX application initiates from the business logic. A developer needs to determine the service class from which alert needs to be triggered. Following steps/instructions can be followed to use event processing mechanism

1. Determine and create an event ID to be used and the group it belongs to. If any existing group cannot be used define a new group.
2. Determine all the parameters that can be used as attributes or actions in the alert's message content. These parameters will be used in the activity log. Keep in mind the end message content while determining the parameters in the activity log.
3. Find the location in the business logic to generate the event and use the API information provided above.
4. Make all the necessary Day 0 entries in the database tables discussed in the previous section.
5. The event processing in OBDX happens in 2 steps, Generation of event and Processing of that event. In a regular REST based service scenario, developer needs to write the logic to generate the event, however the processing of that event is done by OBDX framework upon successful completion of transaction.
6. This 2-step process is based on queue notification framework which is based on either JMS or KAFKA implementation. Ensure that proper configurations are in place based on the implementation preferences. Queue setup information is given in the following sections.
7. Upon successfully completion of the transaction, event is processed by event framework and notifications are sent to the receiver over the configured and desired channels.
8. Populating Notification Details in the activity log – In general, if a user who has logged in into the application, performed a specific transaction and the same user needs to receive the alert, event framework considers its contact details and dispatches the message correctly. However, if the message needs to be delivered to a receiver who has not logged in, developer needs to populate its contact details in notificationDetails field of the ActivityLog object. If the receiver is an OBDX application user, its user id can be set. In other cases, (or alternately for OBDX users case as well) direct contact details like email id or mobile number can also be set in the notification details.
9. DND settings – Event Processing framework allows DND options for the user to stop receiving few alerts. Developer can decide the event groups that are applicable for DND settings. There are 2 ways to set DND for the receiver.
  - a. An admin user can map DND groups to other users using User management transaction
  - b. A user can do the DND mapping by itself using preferences transaction
10. Delivery Mode preferences – A receiver may choose its preferred delivery mode using preferences transaction. By default, all the destinations are set as preferred delivery modes. A user must have at least one delivery mode enabled.
11. Language Preference – A receiver may choose preferred language to receive the alerts. If preferred language is not set Bank's default language will be used to deliver the message. Developer needs to ensure that message templates are properly configured for all the

supported languages defined in the implementations. If the language specific template is not available, by default, template in the English language will be taken up for processing and delivered.

**12. Events in OBDX are categorized in 2 types**

- a. Mandatory Alert – this alert is always sent to the receiver whenever the event occurs
- b. Subscription Alert – this alert is sent to the receiver only if the receiver has subscribed for it. A receiver can subscribe for events using 'Alert Subscription' transaction under preferences.

**13. Message Template maintenance** – as explained above, the message templates for the events can be maintained using Day-0 scripts. However, Administrator user can create or edit these message templates using message Template Maintenance transaction. A template can be deleted as well using this transaction.

(More information on all the above transactions is given in the user manuals.)

**Important Tables in Event Processing**

1. DIGX\_EM\_ALERT\_DISPATCH\_LOG – Provide the final status of the alert
2. DIGX\_EM\_ALERT\_STATE\_LOG – Provides the in-detail logging of the alert processed and the various states it passed through
3. DIGX\_EM\_SUBSCRIPTION – Lists all the users who have subscribed for the event
4. DIGX\_EM\_SUBSCRIBED\_EVENTS – contains the mapping of receivers and the events they have subscribed to
5. DIGX\_EM\_DND\_PREF – Contains the mappings of receiver and the event groups that the respective user has marked for DND.
6. DIGX\_EM\_PREFERRED\_DESTINATION\_PREF – provides the listing of destination preferences maintained by the users

**Event Processing Dispatchers**

OBDX application uses dispatcher classes to provide business logic for sending the notifications to the receiver on desired destination or channel. Each destination must have a dispatcher class associated with it. The application provides a default dispatcher for all five pre-defined destination. An implementor may use custom dispatcher classes for these destinations.

Default Email dispatcher class uses standard JAVA mail APIS to send emails using SMTP server. SMTP configurations need to be maintained in the configuration related tables. Also, SMTP configurations maintenance and testing can be done using 'System Configurations' transaction. Details on this transaction are given in the OBDX core user manual.

Default SMS dispatcher is not pre-integrated with any SMS provider. An implementor is advised to use a custom SMS dispatcher as per the requirements.

Custom dispatcher class must extend following class and implement the necessary methods - '*com.ofss.digx.app.em.alert.service.process.dispatch.dispatcher.AbstractDispatcher*'. A custom message class can also be used to use specific recipient details. This message class must implement '*com.ofss.digx.app.em.alert.service.process.message IMessage*'.

# 5

## Custom Fields For Push notifications

Following Keys can be used to customize Push Notifications.

| KEY NAME                   | VALUE                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------------|
| <b>SOUND_IOS</b>           | File name of custom sound file added to OBDX IOS App                                                |
| <b>SOUND_ANDROID</b>       | File name of custom sound file added to OBDX Android App                                            |
| <b>LARGE_ICON_ANDROID</b>  | URL of icon image to be displayed as large icon in Big Style Push Notification of OBDX Android App. |
| <b>LARGE_IMAGE_ANDROID</b> | URL of image to be displayed in Big Style Push Notification of OBDX Android App.                    |

These custom keys are to be added to the value of “CONTENT” column of **DIGX\_EM\_MESSAGE\_TEMPLATE** table.

If alerts are being created through front end UI, add following keys to “Notification Message” section.

Syntax for adding custom keys to Push Notification alert messages

```
[customfield1Name~customfield1Value|customfield2Name~customfield2Value]
```

### Example 1:

You have requested for #NoOfChequeBook# cheque book with #ChequeBookOption# leaves on Account #AccountNo#.

```
[SOUND_ANDROID~isntit|LARGE_IMAGE_ANDROID~http://static1.squarespace.com/static/54ac6f9ae4b0cf1d82a4b59e/t/587f9e52cd0f68e84c5548fd/1484758653422/?format=300w|SOUND_IOS~chime.m4a]
```

### Example 2:

You have requested for #NoOfChequeBook# cheque book with #ChequeBookOption# leaves on Account #AccountNo#.

```
[SOUND_ANDROID~isntit|LARGE_ICON_ANDROID~http://static1.squarespace.com/static/54ac6f9ae4b0cf1d82a4b59e/t/587f9e52cd0f68e84c5548fd/1484758653422/?format=300w|SOUND_IOS~chime.m4a]
```

# 6

## Multi-Entity Specific templates

Entity specific templates can be created by following ways :

- **Creation of a new alert and template before the entity creation.**

If a new alert has to be maintained before the creation of any new entity, the data for the same has to be inserted in the following tables twice.

One for DETERMINANT\_VALUE '\*' and the other for DETERMINANT\_VALUE 'OBDX\_BU', which is the default entity.

Tables:

DIGX\_EM\_MESSAGE\_TEMPLATE  
DIGX\_EM\_MESSAGE\_TEMPLATE\_DESTINATION\_REL

- **Creation of a new alert and template after the entity creation.**

If a new alert has to be maintained after the creation of entity/entities, the same can be replicated for the different entities using the below queries.

First insert the templates for DETERMINANT\_VALUE '\*' and DETERMINANT\_VALUE 'OBDX\_BU' and then execute the below queries for the respective entities.

```
insert into DIGX_EM_MESSAGE_TEMPLATE(NAME, DESCRIPTION, TITLE, CONTENT,
LOCALE,
 DETERMINANT_VALUE, DELETE_STATUS, EVENT_ID, LAST_UPDATED_BY,
LAST_UPDATED_DATE)
 (SELECT NAME, DESCRIPTION, TITLE, CONTENT, LOCALE,
#determinantValue, DELETE_STATUS,
 EVENT_ID, LAST_UPDATED_BY, sysdate FROM DIGX_EM_MESSAGE_TEMPLATE
WHERE
 DETERMINANT_VALUE = '*')
```

```
insert into DIGX_EM_MESSAGE_TEMPLATE_DESTINATION_REL(TEMPLATE_NAME, LOCALE,
 DETERMINANT_VALUE, DESTINATION_ID) (SELECT TEMPLATE_NAME, LOCALE,
#determinantValue,
 DESTINATION_ID FROM DIGX_EM_MESSAGE_TEMPLATE_DESTINATION_REL WHERE
DETERMINANT_VALUE =
 '*')
```

# 7

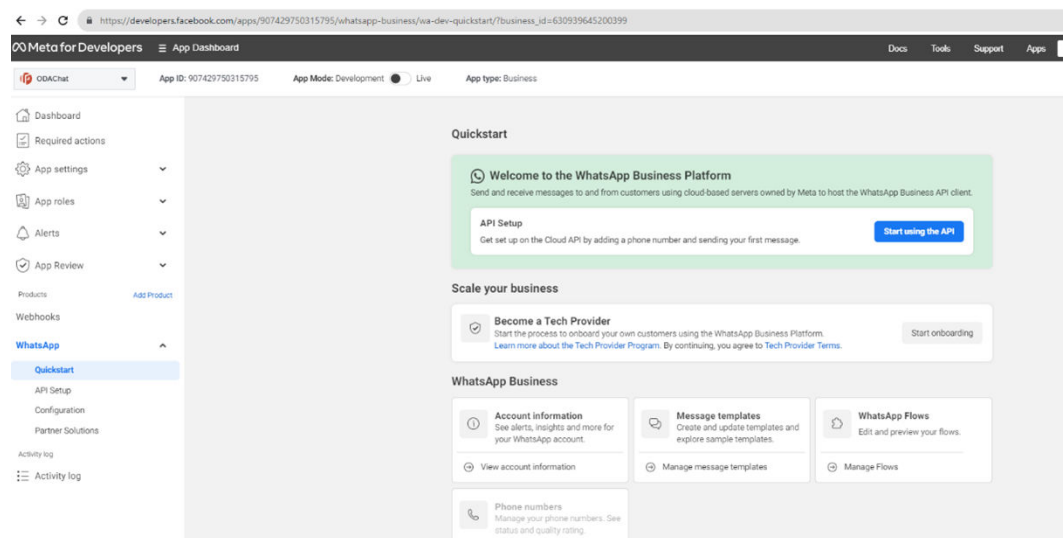
## WhatsApp Configurations

WhatsApp is defined as a destination in Alert framework. WhatsApp messages are delivered to WhatsApp server using OBRH.

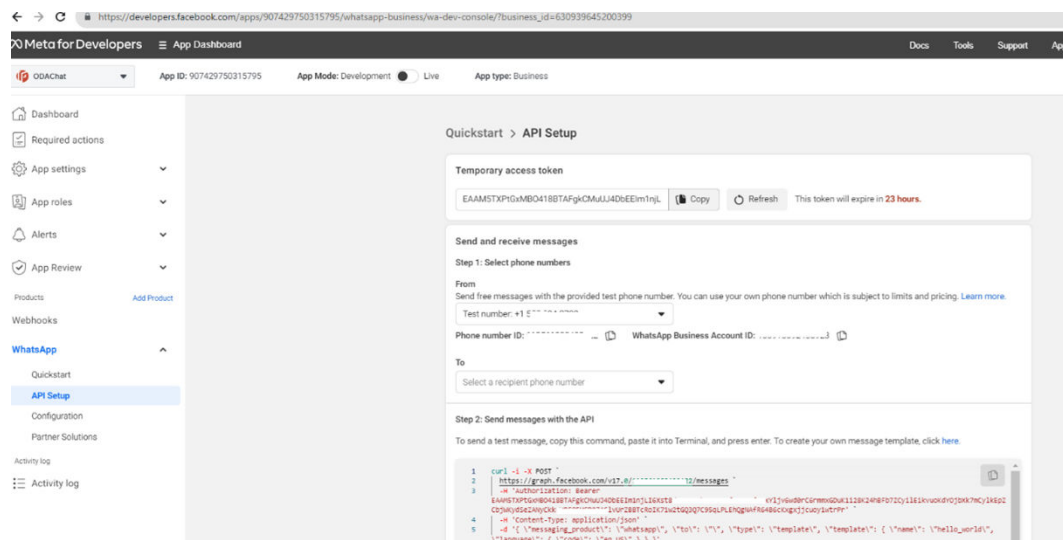
Banks must have a business account registered with WhatsApp.

Navigate to <https://developers.facebook.com/> and setup the WhatsApp capability.

**Figure 7-1 WhatsApp Quickstart**

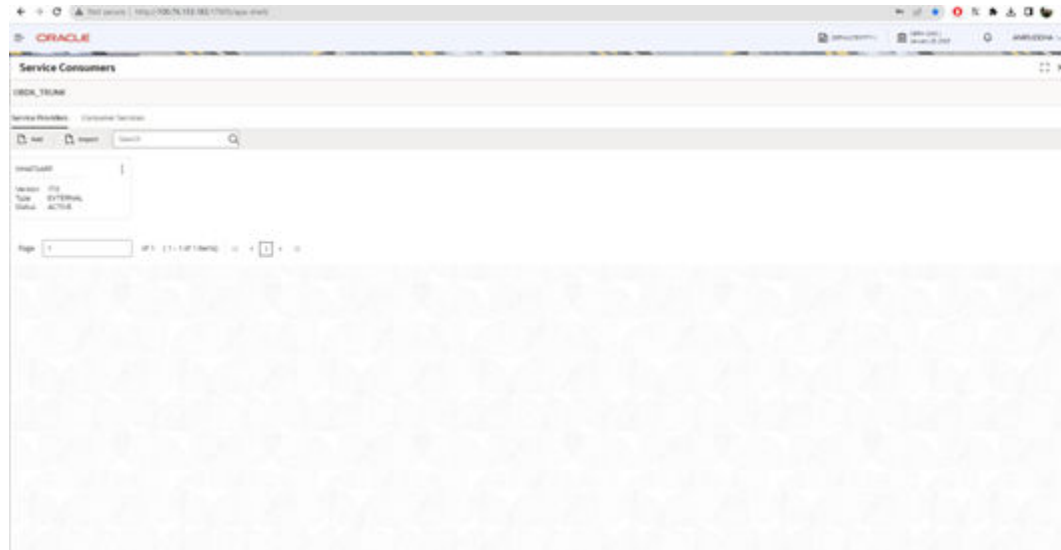


**Figure 7-2 WhatsApp Quickstart- API Setup**



This page gives temporary access token, long term access token can be obtained from <https://developers.facebook.com/tools/explorer/>. In production long term token will be required. This has to be setup in OBRH as shown below:

### Figure 7-3 Service Consumers



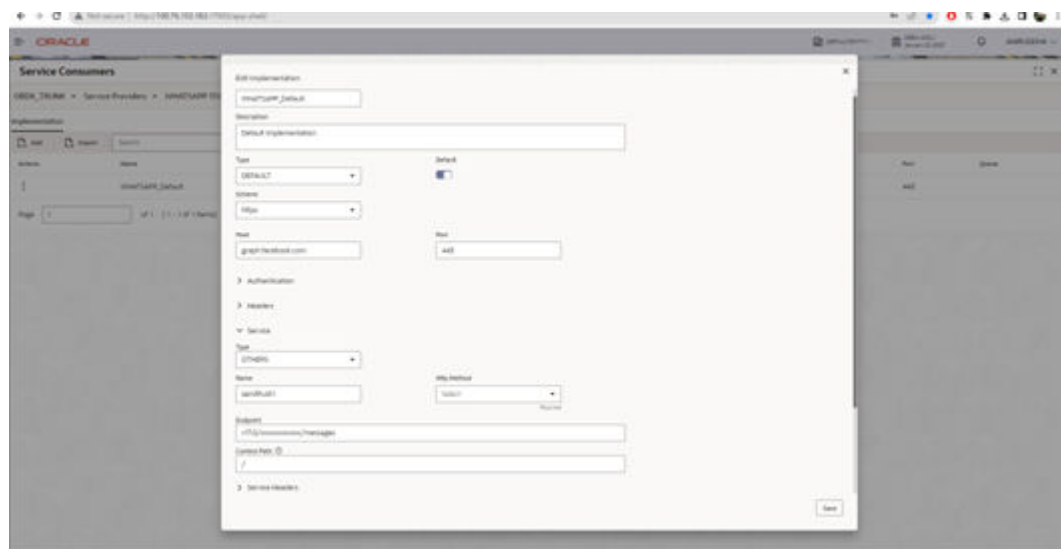
Setup the host, port and token as shown below:

Service needs to be added using **OTHERS** option.

**Note**

The url from the cURL URL shown in API Setup option of developer console.

### Figure 7-4 Service Consumers - Transformation



Map the service and verify the request transformation template (for country codes, if they need to be appended in case of mismatch in phone number format).

Ensure internet connectivity is enabled from OBRH server. Setup proxy in case required in weblogic managed server start args → -Dhttps.proxyHost=www-abc.in.oracle.com -Dhttps.proxyPort=80 -Dhttp.nonProxyHosts=\*.in.oracle.com.

**Figure 7-5 Map the service and verify the request transformation template**



# Index

## C

---

Code Configuration, [1](#)  
Custom Fields For Push notifications, [1](#)

## D

---

Database Configurations, [1](#)

## E

---

Event Processing, [1](#)

## M

---

Multi-Entity Specific templates, [1](#)

## W

---

WhatsApp Configurations, [1](#)