Oracle® Banking APIs
Third Party Simulation Guide

Innovation Release 25.1.1.0.0

G46853-01

October 2025

ORACLE®

Oracle Banking APIs Third Party Simulation Guide, Innovation Release 25.1.1.0.0

G46853-01

Contents

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.
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.

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

Objective and Scope

1.1 Background

This document is intended to outline the OBAPI-External system simulation process.

OBAPI is the banking solution platform which needs to be integrated with one of core banking systems to retrieve the customer banking details and to execute the set of operations on the core banking system as per the service requirement. External system simulation will simulate host system and will return response whenever it receives request.

1.2 Scope

The scope of this document is to explain the set up of external system simulation environment considering external system has exposed JMS as API to OBAPI. Details covered are as follows:

- Basic Definitions
- Sequence diagram of External system simulator.
- Creation of queue to send the message (producer queue) and to receive message (consumer queue) on application server.
- Deployment of Simulator MDB.
- Response files guidelines.

2

Basic Definitions

- [Java Message Service (JMS)](#)
- [Message Driven Bean (MDB)](#)

2.1 Java Message Service (JMS)

Java Message Service (JMS) API, a Java API that allows applications to create, send, receive, and read messages using reliable, asynchronous, loosely coupled communication.

Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility: A messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages. It enables distributed communication that is loosely coupled. A component sends a message to a destination, and the recipient can retrieve the message from the destination.

2.2 Message Driven Bean (MDB)

A message-driven bean is an enterprise bean that allows Java EE applications to process messages asynchronously. This type of bean normally acts as a JMS message listener, which is similar to an event listener but receives JMS messages instead of events. The messages can be sent by any Java EE component (an application client, another enterprise bean, or a web component) or by a JMS application or system that does not use Java EE technology. Message-driven beans can process JMS messages or other kinds of messages

Client components do not locate message-driven beans and invoke methods directly on them. Instead, a client accesses a message-driven bean through, for example, JMS by sending messages to the message destination for which the message-driven bean class is the MessageListener. When a message arrives, the container calls the message-driven bean's onMessage method to process the message.

Enterprise beans may be packaged within a EAR module as Java programming language class files or within a JAR file that is bundled within the EAR module, or it can be packaged within WAR module.

Jar contains ejb-jar.xml deployment descriptor. it must be located in the JAR module's WEB-INF directory.

Figure 2-1   Fig. 1.0 [ejb-jar.xml]

```
<display-name>com.ofss.digx.extxface.mdb</display-name>
<enterprise-beans>
  <message-driven>
    <display-name>ExtxfaceSimulatorMDB</display-name>
    <ejb-name>ExtxfaceSimulatorMDB</ejb-name>
    <ejb-class>com.ofss.digx.extxface.mdb.ExtxfaceSimulatorMDB</ejb-class>
    <transaction-type>Bean</transaction-type>
    <message-destination-type>javax.jms.Queue</message-destination-type>
  </message-driven>
</enterprise-beans>
</ejb-jar>
```

The WebLogic Server weblogic-ejb-jar.xml deployment descriptor file describes the elements that are unique to WebLogic Server. Configuration of queue details, which mdb listener listen are mentioned here.

Figure 2-2   Fig. 2.0 [weblogic-ejb-jar.xml]

```
<weblogic-ejb-jar xmlns="http://xmlns.oracle.com/weblogic/weblogic-ejb-jar" xmlns:xsi="h
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-ejb-jar/1.1/weblogic-ejb-j
    <weblogic-enterprise-bean>
        <ejb-name>ExtxfaceSimulatorMDB</ejb-name>
        <dispatch-policy>FileProcessWorkManager</dispatch-policy>
        <message-driven-descriptor>
            <pool>
                <initial-beans-in-free-pool>10</initial-beans-in-free-pool>
                <max-beans-in-free-pool>100</max-beans-in-free-pool>
            </pool>
            <destination-jndi-name>EndPointPREPROCESS</destination-jndi-name>
            <connection-factory-jndi-name>EndPointQCF</connection-factory-jndi-name>
            <jms-polling-interval-seconds>1</jms-polling-interval-seconds>
        </message-driven-descriptor>
```

3

External System Simulation

This section describes how simulation of external system will happen with OBAPI considering external system exposing JMS.

This simulator will consume request and produce response. It will pick up request from producer queue and post the response on consumer queue. For same, **ExtxfaceSimulatorMDB.ear** should be deployed, as Enterprise Application, on application server. MDB listener in ExtxfaceSimulatorMDB.ear will listen producer queue, retrieves request whenever it comes, process it and sends response to the consumer queue. Below sequence diagram explain the same.

- [Sequence Diagram](#)
- [Producer and Consumer queue configuration](#)
- [Deployment of ExtxfaceSimulatorMDB.ear](#)
- [Response files guidelines](#)

3.1 Sequence Diagram

Figure 3-1   Fig. 3.0 [Sequence diagram of External System Simulator]
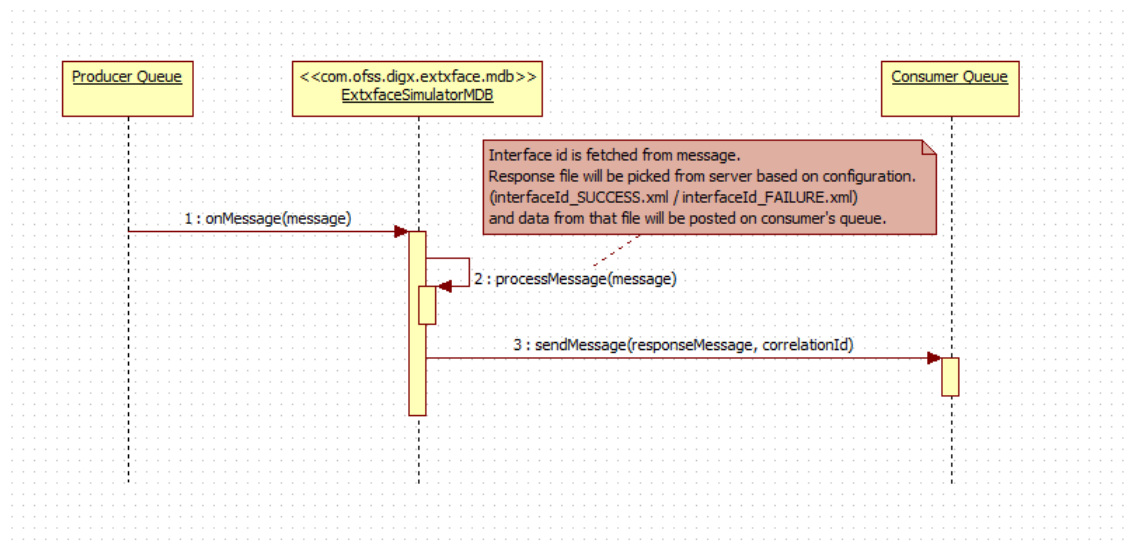
3.2 Producer and Consumer queue configuration

ExtxfaceSimulatorMDB.ear will contain details of producer queue, consumer queue details will be configured in a property file. Create producer and consumer queue with JNDI names as given below on application server.

1. Producer Connection Factory JNDI name: **ExtSystemSenderQCF**

2. Producer Queue JNDI name: ExtSystemSenderQueue

3. Consumer Connection Factory JNDI name: ExtSystemReceiverQCF

4. Consumer Queue JNDI name: **ExtSystemReceiverQueue**

3.3 Deployment of ExtxfaceSimulatorMDB.ear

Deployment of ear can be manual or automated.

ExtxfaceSimulatorMDB.ear should be deployed on application server as an enterprise application.

3.4 Response files guidelines

This section describes response file location on server, its naming convention and file type.

MDB Listener responds to OBAPI system by identifying interface id. Using interface id and usertype provided to MDB, response file will be picked from **applications/ com.ofss.digx.extxface.mdb/xml/{usertype}** folder in the mdb.

Every transaction (interface id) will have two related files with below file names.

• {interface_id}_SUCCESS.xml

• {interface_id}_FAILURE.xml

Based on configuration done for transaction result in table `DIGX_FW_CONFIG_ALL_B` for `category_id =' ExtSystemsConfig'` MDB Listener will pick up success or failure response xml to post it on consumer queue.

Index

B

D

E

J

M

P

R

S