Oracle Banking APIs

Third Party Simulation Guide
Release 19.1.0.0.0

Part No. F18559-01

May 2019

ORACLE®

Third Party Simulation Guide

May 2019

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/

Table of Contents

1. Preface

1.1 Intended Audience

This document is intended for the following audience*:*

- Customers
- Partners

1.2 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.

1.3 Access to OFSS Support

Oracle customers have access to electronic support through My Oracle Support. For information,

visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.4 Structure

This manual is organized into the following categories:

Preface gives information on the intended audience. It also describes the overall structure of the User Manual.

The subsequent chapters describes following details:

- Introduction
- Configuration / Installation.

1.5 Related Information Sources

For more information on Oracle Banking OBAPIs Release 18.3.0.0.0, refer to the following documents:

- Oracle Banking OBAPIs Licensing Guide

2. Objective and Scope

2.1 Background

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

OBAPIs 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.

2.2 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 OBAPIs. 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.

Home

3. Basic Definitions

3.1 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.

3.2 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.

```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>
```

Fig. 1.0 [ejb-jar.xml]

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.

```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>
```

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

Home

4. External System Simulation

This section describes how simulation of external system will happen with OBAPIs 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.
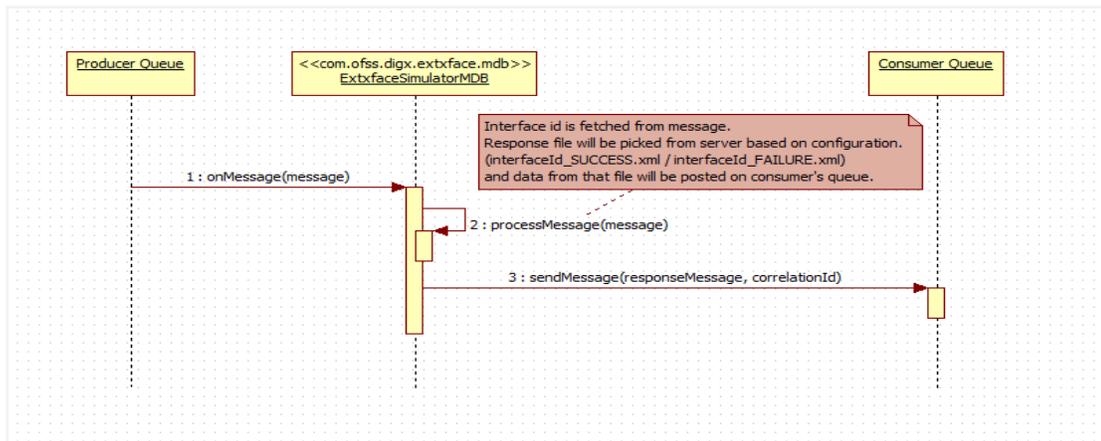
4.1 Sequence Diagram

Fig. 3.0 [Sequence diagram of External System Simulator]

4.2 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.

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

    b. Producer Queue JNDI name: ExtSystemSenderQueue

    c. Consumer Connection Factory JNDI name: ExtSystemReceiverQCF

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

4.3 Deployment of ExtxfaceSimulatorMDB.ear

Deployment of ear can be manual or automated.

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

4.4 Response files guidelines

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

MDB Listener responds to OBAPIs 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.

Home