MESSAGE PROPAGATION FROM ORACLE TO WEBSHPERE MQ
USING MESSAGING GATEWAY

Rajashekhar Gopalapuram
B.Tech, Jawaharlal Nehru Technological University, India, 2006

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2010
MESSAGE PROPAGATION FROM ORACLE TO WEBSHPERE MQ
USING MESSAGING GATEWAY

A Project

by

Rajashekhar Gopalapuram

Approved by:

______________________________________, Committee Chair
William J Mitchell, Ph.D.

______________________________________, Second Reader
Meiliu Lu, Ph.D.

_______________________________________
Date
Student: **Rajashekhar Gopalapuram**

I certify that this student has met the requirements for the format contained in the University format manual, and that this project is suitable for shelving in the library and credit is to be awarded for the Project.

_______________________________, Graduate Coordinator

Nikrouz Faroughi, Ph.D.                    Date

Department of Computer Science
Abstract

of

MESSAGE PROPAGATION FROM ORACLE TO WEBSHPERE MQ USING MESSAGING GATEWAY

by

Rajashekhar Gopalapuram

In this project, I will be propagating the messages between Oracle Server and WebSphere MQ through a Messaging Gateway.

When web based business applications communicate, the producer application enqueue messages and the consumer applications dequeue the messages. The enqueued messages are said to be propagated when they are reproduced in another queue in same database or in a remote database.

Oracle Streams AQ provides database-integrated message queuing functionality which is built on top of Oracle Streams and leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S).

Oracle Streams AQ 11g Release 1 supports queue-to-queue propagation. Messages are propagated from one queue to another queue; the destination may be same or remote database. Propagation enables you to fan out messages to a large number of recipients. Also, through propagation we can combine messages from different queues into a single
queue known as compositing or funneling messages. Oracle Stream AQ has propagation between two Oracle Streams AQ queues to enable e-business. Messaging Gateway extends this functionality to applications based on non-Oracle messaging systems such as applications based on Web Sphere MQ. Conversely, messages published by non-Oracle messaging system applications can be consumed by Oracle Streams AQ applications. Message gateway also supports message conversion between Oracle Streams AQ messages and non-Oracle messaging system messages using built in routines or customized message transformation.

In general, integrated applications involve the integration of various messaging systems. Oracle Streams AQ seamlessly integrates with existing non-Oracle Database messaging systems like IBM Web Sphere MQ through Messaging Gateway, thus allowing existing Web Sphere MQ-based applications to be integrated into an Oracle Streams AQ environment.

Most of the facts discussed in the Abstract are referenced from [1].

______________________________, Committee Chair
William J Mitchell, Ph.D.

______________________________
Date
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. IBM DB2 EXPRESS C FEATURES AND INSTALLATION</td>
<td>5</td>
</tr>
<tr>
<td>Installation and Configuration</td>
<td>7</td>
</tr>
<tr>
<td>Create a Sample Database</td>
<td>9</td>
</tr>
<tr>
<td>Test Connectivity of DB2</td>
<td>9</td>
</tr>
<tr>
<td>3. IBM WEBSPHERE MQ</td>
<td>11</td>
</tr>
<tr>
<td>Terminology</td>
<td>11</td>
</tr>
<tr>
<td>Using WebSphere MQ</td>
<td>12</td>
</tr>
<tr>
<td>Features of WebSphere MQ</td>
<td>12</td>
</tr>
<tr>
<td>Supported API’s</td>
<td>13</td>
</tr>
<tr>
<td>Installation and Configuration of WebSphere MQ</td>
<td>13</td>
</tr>
<tr>
<td>Installing the DB2 WebSphere MQ Functions</td>
<td>14</td>
</tr>
<tr>
<td>4. WEBSPHERE MQ QUEUE LOAD / UNLOAD UTILITY</td>
<td>20</td>
</tr>
<tr>
<td>MQ Load/Unload Utility Installation</td>
<td>20</td>
</tr>
<tr>
<td>Unload/Load Features</td>
<td>21</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Page

1. Figure 6.1 Integrated Application Environment using Oracle Streams AQ...........32

2. Figure 7.1 Messaging Gateway Architecture..............................................41
Chapter 1

INTRODUCTION

The profitability of the businesses increased drastically with advent of Information Technology. Businesses started working with both internal and external organization to integrate their applications, reducing overall turn around time and increasing the customer satisfaction. Data communication is the prime enabler. As data is being stored in the databases, this led to the evolution of database integrated solutions. Web based business applications increased the use of message queuing solutions for communicating data. When web based business applications communicate, the producer application enqueues (inserting messages into a designated queue of an application) messages and the consumer applications dequeues (accepts the messages from the designated queue) messages [1]. The enqueued messages are said to be propagated when they are reproduced in another queue in same database or in a remote database [1].

Oracle Streams AQ provides database-integrated message queuing functionality which is built on top of Oracle Streams and leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases. They are transmitted using Oracle Net Services and HTTP(S) [1]. Being stored in database they also have high availability, scalability and reliability features [1].
The ORACLE AQ messages support standard SQL querying, through which we can get access to message properties, message history and payload (data contained in the message) [1]. Oracle Streams AQ supports both system level access control and queue level access control [1].

System level access control: The DBA designates a queue administrator who can invoke the administrative and operational interfaces on the queues to manage the queues [1].

Queue level access control: During the enqueue and dequeue operations, this feature protects the queues created in one schema from the applications running in other schemas by giving the fewest necessary privileges to the applications running outside the schema [1].

Performance [1]:
The efficiency of the Oracle Streams AQ can be increased by separating the request for services and supply of services in the applications. In general, the performance of the Oracle Streams AQ can be measured using following

1. Number of messages enqueued and dequeued per second.
2. Time to evaluate a complex query on a message warehouse.
3. Time to recover and restart the messaging process after failure.

Oracle Stream AQ is scalable; it can accommodate more messages as the usage increases. It shows high performance as the increase in number of programs using the application increases, messages increase and when the size of warehouse increases [1].
ORACLE RAC Environment: An Oracle RAC database is a clustered database. A Cluster is a pool of independent servers that together form a single system. In this scenario the database software is installed on the independent servers to access single database [2].

Oracle real application clusters (RAC) are also supported by the Oracle Streams AQ [1]. The performance of the Streams AQ can further be improved by managing different queues in different instances of RAC [1].

The following sections of this chapter are drawn from [1].

Oracle Streams AQ 11g Release 1 (11.1) supports queue-to-queue propagation. Messages are propagated from one queue to another queue; the destination queue may be in the same or a remote database. Propagation enables fanning out messages to a large number of recipients. Also, through propagation we can combine messages from different queues into a single queue known as compositing or funneling messages. Oracle Stream AQ has propagation between two Oracle Streams AQ queues to enable e-business. Messaging Gateway extends this functionality to applications based on non-Oracle messaging systems such as applications based on Web Sphere MQ. Conversely, messages published by non-Oracle messaging system applications can be consumed by Oracle Streams AQ applications. Message gateway also supports message conversion between Oracle Streams AQ messages and non-Oracle messaging system messages using built in routines or by customized message transformation.

In general, integrated applications involve the integration of various messaging systems. Oracle Streams AQ seamlessly integrates with existing non-Oracle Database messaging
systems like IBM Web Sphere MQ through Messaging Gateway, thus allowing existing
Web Sphere MQ-based applications to be integrated into an Oracle Streams AQ
environment.

In order to propagate the messages from Oracle AQ streams to WebSphere MQ, the
Database servers, IBM Websphere MQ and required API’s needs to be installed. Then
the Oracle Messaging Gateway objects needs to be loaded and set up on Oracle Database
Server. On validating the Oracle Messaging Gateway setup, the Messaging Gateway
should be configured to send messages to registered non-Oracle messaging system (IBM
Web sphere MQ) queues through the Messaging System Links.
Chapter 2

IBM DB2 EXPRESS C FEATURES AND INSTALLATION

In order to enable DB2 WebSphere MQ functions, we need to install the DB2 data server software on the WebSphere MQ messaging system. We are using DB2 Express-C data server software which can manage both relational and XML data [3]. It is free software downloadable from the IBM website. This can be used by the variety of users such as [3]:

1. Application developers for building standalone, client-server and enterprise applications.
2. Architects for evaluating and prototyping.
3. Supports Web 2.0 and next generation applications.

DB2 Express C can be installed on both 32-bit and 64-bit software with Linux/windows/Solaris. It also supports the pureXML, the unique DB2 technology which stores and processes XML documents [3].

The following languages and standards can be used to build application using DB2 Express C software [3].

- SQL, XQuery, XPath
- C/C++ (CLI, ODBC and embedded SQL)
- Java (JDBC and SQLJ)
- COBOL
- PHP
- Perl
- Python
- Ruby on Rails
- .NET languages
- OLE-DB
- ADO
- MS Office: Excel, Access, Word
- Web services

The free version of DB2 Express C has core DB2 capabilities, Administration tools, Development tools, autonomic features and pureXML features. The maximum processors used by this version are 2, with maximum memory utilization of 2GB [3]. DB2 Express C Subscription provides the DB2 fix packs and the features like High Availability & Disaster Recovery, Data Replication [3]. The DB2 HADR solution gives you following features with low impact on overall system performance [3]

- Fast failover capability
- Full transaction atomicity
- Upgrade current systems without interrupting the service.
- Remote System failover

A demo of HADR is available at

http://www.ibm.com/software/data/db2/express/demo.html
DB2 Express C does not include the features like Database partitioning, Connection concentrator, Geodetic Extender, Label based access control, workload manager and Deep compression [3].

Installation and Configuration

Prerequisites:
IBM DB2 express C is available for download on platforms like Linux®, Sun Solaris (x64), and Microsoft Windows® 2003, XP, and Vista. In this project it is installed on Windows XP 32 bit OS [3].

Hardware Requirements: DB2 Express C can be installed on systems with any number of processors; however it does not utilize more than 2 cores and 2GB memory. In this project it is installed on dual core system with 3 GB RAM [3].

The OS user installing the DB2 must belong to Administrators group in Windows. Download the software from the IBM website and uncompress image [3]. Launch the wizard by executing setup.exe at EXP/image/ directory.

Installation procedure:
The step by step installation and configuration is detailed in Appendix A.
Validation of Installation:

You can validate the installation files, instance and database functionality by running the validation tool db2val.exe located at C:\Program Files\IBM\SQLLIB\bin\db2val.exe [3]

C:\Program Files\IBM\SQLLIB\BIN>db2val.exe

DB1279I The db2val command is running. This can take several minutes.

DB13331 Installation file validation for the DB2 copy DB2COPY1 was successful.

DB1399I The instance validation for the instance DB2 was successful.

DB13431 The db2val command completed successfully. For details, see the log file C:\DOCUME~1\My Docu~1\DB2LOG\db2val-Sat Mar 13 16_48_36 2010.log

C:\Program Files\IBM\SQLLIB\BIN>

The DB2 extended windows security is enabled, thus the db2 user that runs the applications or tools must be added to the DB2 administrators group. The DB2 instance has standard connection port number 50000 [3]. However the port number to which DB2 instance connects can be modified if the default port is already in use.

You can check the license information of the product installed using the command db2licm -l

C:\Program Files\IBM\SQLLIB\BIN>db2licm -l
Product name: "DB2 Express-C"
License type: "Unwarranted"
Expiry date: "Permanent"
Product identifier: "db2expC"
Version information: "9.7"
Max number of CPUs: "2"
Max amount of memory (GB): "2"
Create a Sample Database

![Image of DB2 First Steps interface]

Create a sample database with SQL data only.

Test Connectivity of DB2 [3]

Once the SAMPLE database is created, run db2cwadmin.bat in command prompt. It opens a command prompt window, enter DB2 this will start the command line client (CLI). Connect to database using following command.

CONNECT TO sample USER <userid> USING <password>
Connection is successful.

Additional DB2 Resources


DB2 Express-C download page:

DB2 Express-C developer handbook:
http://www.redbooks.ibm.com/abstracts/sg247301.html
IBM WebSphere MQ is dynamic integration software through which programs can talk to each other over the network with different components (processors, OS, communication protocol) using a reliable application program interface [4]. This is a messaging and queuing middleware and includes publish/subscribe features. It allows the systems to operate independently, but guarantees the delivery of messages. Thus WebSphere MQ is used in the mission critical applications in industry for its robustness and reliability [4].

Terminology [4]

Messaging: Programs communicate with each other by sending data in the form of messages rather than calling each other directly.

Queuing: The messages are stored in the queues during the transit, waiting to be dequeued.

Topic: The Topic determines what a publication is about. Topic is a character string that describes the subject of the information that is published in publish/subscribe message.

Publish/Subscribe: Allows the programs to publish the data at a single destination and the WebSphere MQ distributes the data to various subscriber programs. In this scenario the publisher defines topics and the subscriber specifies the topics it wants to receive.
Using WebSphere MQ [4]

Using WebSphere MQ the programmers need not write communication programs and can send and receive data between applications over various networks. The Message delivery is guaranteed and is decoupled from the application. The initial configuration of the MQ can be done using the MQ Explorer workbench and its associated tools. The MQI is used by the application to connect to the queue manager, open the queues or topics, and put/get messages.

Features of WebSphere MQ [4]

1. A queue or topic is owned and run by a queue manager.
2. The queues and queue managers are stored in WebSphere MQ storage rather than DB2 tables.
3. The queue managers can be started manually or during the system boot time (for example the queue manager can run under MQ windows service IBM MQseries)
4. If application wants to transfer data, it puts the data in to a message and then puts the message on to the queue or publishes it with a topic.
5. The Queue manager can also be configured to send the messages to a remote queue on a queue manager on another computer.
6. Once the message is stored in the queue, the application can get the messages from the queue at any time as needed.
7. The subscriber application is sent the messages by a queue manager.
8. You can have many queues and topics on the same queue manager and more than one queue managers on the same system.

9. Also there can be a client installation with no queue managers, but uses the queue managers on the server installation on another computer for messaging.

Supported API’s [4]

Message Queuing Interface (Used for program to program communication)

Application Messaging Interface (Provides support for point to point messaging, publishing and subscribing)

Java Messaging Services (allows application to create, send, receive and read messages asynchronously)

Installation and Configuration of WebSphere MQ

1. Download the trial software version from IBM Website

2. Install the prerequisite software WebSphere Eclipse platform version 3.3 from C:\Documents and Settings\Raj\My Documents\Masters project\Project Downloads\WebSphere MQ v 7.0.1 Trial\Prereqs\IES [4]

3. The step by step screen shots of installation of IBM WebSphere MQ is shown in Appendix B.

4. Install the Application Messaging Interface as shown in Appendix C.
Installing the DB2 WebSphere MQ Functions

Prerequisites [4]:

Make sure the DB2 Installation added the db2qgmd.dll and db2mqsw.dll in directory sqllib\bin for windows installation.

Install the IBM WebSphere MQ Application Messaging Interface, Version 1.2.5

Add the AMT_DATA_PATH environment variable to the list that is used by DB2 to ensure that the message queuing user-defined functions (MQ UDFs) execute correctly. In order to do this run the following command in the command prompt.

db2set DB2ENVLIST="AMT_DATA_PATH" and restart the database instance.
By default the environment variable is set to the instance.

Database Manager Configuration:

In order to use the transactional MQ UDFs, make sure to configure the database for federated operations using

update dbm cfg using federated yes

```
C:\Documents and Settings\Raj>db2cmd
```

```
C:\Documents and Settings\Raj>db2 update dbm cfg using federated yes
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command completed successfully.
```

```
C:\Documents and Settings\Raj>db2 get dbm cfg
```

```
Database Manager Configuration

Node type = Database Server with local and remote clients

Database manager configuration release level  = 0x0d00

Maximum total of files open  <MAXTOTFILEO>  = 16000
CPU speed <millisec/instruction>  <CPU_SPEED>  = 3.30641E-007

Max number of concurrently active databases  <NUMDB>  = 8
Federated Database System Support  <FEDERATED>  = YES
Transaction processor monitor name  <TP_MON_NAME>  =
```
Restrictions [4]:

The DB2 UDB MQ transactional functions existing under db2mq1c does not support CLOB messages. MQ user-defined functions allow only 40 AMI Policies to exist in an AMI repository file for the queue manager specified with the -q option.

The transactional MQ user-defined functions support only one Queue Manager within a single transaction

If you use publish/subscribe functions, you need to create certain WebSphere MQ objects which can be using MQSC commands as shown below (We are not using this in the project)

a. Make sure that you have the amtsamp.tst and amtsdfts.tst files
b. Update the *.tst files for your queue manager, if necessary
c. Start the queue manager that is used by your AMI service
d. Issue a command similar to the following command:

runmqsc QMName <amtsamp.tst

Procedure [4]:

1. Run the amtsetup.sql script. If this is the first time you are enabling the database, ignore the errors from the drop table statements.
2. Change your current directory to sqllib/cfg/mq.
3. From the MQ directory type db2 -tvf amtsetup.sql.
4. Configure and enable the database with MQ Function
The enable_MQFunctions checks the WebSphere MQ environment and installs the default configuration for the WebSphere MQ functions. Assuming the user is connected to the database

```
enable_MQFunctions -n sample -u user1 -p password1
```

The user should have required privileges to enable the function; by default db2 admin doesn’t have required authorizations.

Test the MQ function by running following command in the command prompt in mq directory.

```
C:\Program Files\IBM\SQLLIB\cfg\mq>db2 values DB2MQ.MQSEND('TEST')
1
```

The MQ functions are enabled successfully.

Errors occurred while enabling of MQ Functions:

1. While enabling the MQ functions you may come across privileges issues as shown below
Resolution:

Connect to database sample as db2admin

Run the following command in db2cmd prompt

C:\Documents and Settings\Raj>enable_MQFunctions -n sample -u db2admin -p project3

*** Please wait: creating queue manager (DB2MQ_DEFAULT_MQM) ....
*** Please wait: starting the queue manager (DB2MQ_DEFAULT_MQM) ....
--- Create MQ Functions ...
SQL error or warning occurred. Processing can continue...
Inspect the SQL error or warning conditions that occurred.

UNEXPECTED FAILURE <SQLSTATE=42502> <SQLCODE=-552>
<errorMsg: "[IBM][CLI Driver][DB2/NT] SQL0552M "DB2ADMIN" does not have the privilege to perform operation "CREATE FUNCTION". LINE NUMBER-1. SQLSTATE=42502"

*** Error -- while creating functions
*** enable_MQFunction finished with error

C:\Documents and Settings\Raj>
2. When you try to Create DB2MQ1C functions under schema DB2MQ1C you might come across following issue.

```
$ Documents and Settings\Raj>enable_MQFunctions -n sample -u db2admin -p project
$ Create MQ Functions
$ SQLExecDirect - $SELECT COUNT(*) FROM DB2MQ.MQPOLICY

--- (-1)

SQL error or warning occurred. Processing can continue...
Inspect the SQL error or warning conditions that occurred.

UNEXPECTED FAILURE (SQLSTATE=42882) (SQLCODE=-204)
<errorMsg: [IBM]CLI Driver]IDB2/NT] SQL204N "DB2MQ:MQPOLICY" is an undefined name. SQLSTATE=42704

An error occurred during a configuration check.
The MQPolicy table does not exist, or the table is empty.
Create the configuration database objects using the script that is provided before you issue the command again.
Validate MQ Functions...
SQL error or warning occurred. Processing can continue...
Inspect the SQL error or warning conditions that occurred.

UNEXPECTED FAILURE (SQLSTATE=42884) (SQLCODE=-448)
<errorMsg: [IBM]CLI Driver]IDB2/NT] SQL2448N No authorized routine named "MQ END" of type "FUNCTION" having compatible arguments was found. SQLSTATE=42884

*** enable_MQFunction finished with error
$ Documents and Settings\Raj>enable_MQFunctions -n sample -u db2admin -p project
$ Create MQ Functions
$ SQLExecDirect - $SELECT COUNT(*) FROM DB2MQ.MQPOLICY

--- (-1)

SQL error or warning occurred. Processing can continue...
Inspect the SQL error or warning conditions that occurred.

UNEXPECTED FAILURE (SQLSTATE=42882) (SQLCODE=-204)
<errorMsg: [IBM]CLI Driver]IDB2/NT] SQL204N "DB2MQ:MQPOLICY" is an undefined name. SQLSTATE=42704

An error occurred during a configuration check.
The MQPolicy table does not exist, or the table is empty.
Create the configuration database objects using the script that is provided before you issue the command again.
Validate MQ Functions...
SQL error or warning occurred. Processing can continue...
Inspect the SQL error or warning conditions that occurred.

UNEXPECTED FAILURE (SQLSTATE=42884) (SQLCODE=-448)
<errorMsg: [IBM]CLI Driver]IDB2/NT] SQL2448N No authorized routine named "MQ END" of type "FUNCTION" having compatible arguments was found. SQLSTATE=42884

*** enable_MQFunction finished with error
$ Documents and Settings\Raj>db2end
```

Resolution: Disconnected as db2admin and connected as RAJ (SYSADMIN) to sample database
Chapter 4

WEB SPHERE MQ QUEUE LOAD / UNLOAD UTILITY

In this chapter we will discuss about WebSphere MQ Load/Unload utility to validate the WebSphere MQ Configuration. The WebSphere MQ Queue Load /Unload Utility is used to copy or move the messages from the message queue to a file. This can be done using a selection criteria based on current position in the queue or search string or age of the message [5].

MQ Load/Unload Utility Installation [5]

Supports following platforms [5]:

- Windows
- AIX
- Linux Intel
- Linux Intel 64
- Linux Power
- HP
- HP Itanium
- Solaris
- z/OS

In this project, the tool is installed on Windows 32 bit machine by executing qload.exe.
Unload/Load Features [5]

Unload a queue to file:

In order to archive the messages or for later reload we unload the queue messages to a file using following command [5].

qload –m QM1 –i Q1 –f c:\myfile

Start the sample database and enable MQ functions.

Start the Default queue in the WebSphere MQ.
Output file:

C:\unload1.txt

To unload a queue and write each message to a different file use the following command

\[5\]

```
qload -m QM1 -i Q1 -f c:\myfile%n
```

Where QM1 is queue manager and Q1 is queue.
The message is stored in the format shown below in a file [5].

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>The text shown on this line is ASCII string text</td>
</tr>
<tr>
<td>X</td>
<td>The text shown on this line is hex</td>
</tr>
<tr>
<td>A</td>
<td>The text shown on this line is an attributed in the Message Descriptor (MQMD)</td>
</tr>
<tr>
<td>*</td>
<td>The text on this line is a comment and will be ignored</td>
</tr>
</tbody>
</table>

Similarly we can do following operation on the queues using this utility [5]

- Load a queue from a file.
  
  ```
  qload –m QM1 –o Q1 –f c:\upload1
  ```

- Load queue from a series of files
  
  ```
  qload –m QM1 –o Q1 –f c:\myfile%n
  ```

- Copy messages from one queue to another queue.
  
  ```
  qload –m QM1 –i Q1 –o Q2
  ```

- Move messages older than one day from one queue to another.
  
  ```
  qload –m QM1 –i Q1 –o Q2 –T1440
  ```

- Change the format of the unloaded file.
  
  ```
  qload –f c:\oldfile –f c:\newfile –dA
  ```
New formatted file is

C:\unload_new.txt

- Display ages of messages currently in queue.

qload –m QM1 –i Q1 –f stdout -dT

I have used the WebSphere MQ Load /unload utility to test sending messages and check all desired functionalities are configured correctly in WebSphere MQ.
Chapter 5

WEBSHHERE MQ CLASSES FOR JAVA

WebSphere MQ classes for Java are also referred to as WebSphere MQ base Java [6]. Using these classes we can connect directly to the WebSphere MQ Server or through a WebSphere MQ Client [6].

This includes the Message Queue Interface (MQI) through which programs can access the queue manager, structures to send/receive data from the queue manager and native WebSphere MQ Application Programming Interface.

Installation [6]

The WebSphere MQ classes for Java are installed with the installation of the WebSphere MQ and is contained in the JAR file com.ibm.mq.jar

WebSphere MQ JMS applications need the dhbcore.jar file provided by the WebSphere MQ Java.

Following libraries are provided by the WebSphere MQ Java

- connector.jar (Version 1.0)
- fscontext.jar (Version 1.2)
- jms.jar (Version 1.1)
- jndi.jar (Version 1.2.1)
- jta.jar (Version 1.0.1)
- ldap.jar (Version 1.2.2)
• providerutil.jar (Version 1.2)

Environment Variables for WebSphere MQ Base Java [6]

In order to run the WebSphere MQ base Java applications set CLASSPATH environment variable as mentioned below

CLASSPATH=install_dir\Java\lib\com.ibm.mq.jar;
install_dir\Tools\Java\base;

Here the install_dir is the directory in which you installed WebSphere MQ. For Windows, this directory is normally C:\Program Files\IBM\WebSphere MQ.

To run the JMS application add the following to the environment variable CLASSPATH setting as mentioned below

CLASSPATH=install_dir\Java\lib\com.ibm.mqjms.jar;
install_dir\Tools\Java\jms;

Other environment variables relevant to WebSphere MQ Java are

MQ_JAVA_DATA_PATH : specified directory for log and trace output.
MQ_JAVA_INSTALL_PATH : Installation directory of WebSphere MQ Java.
MQ_JAVA_LIB_PATH : Specifies the directories where the WebSphere MQ Java libraries are stored.

In Windows, the above variables are set automatically by the installation.

To specify the JNI libraries location, execute the application with following command

Java -Djava.library.path=library_path application name
For windows, the library_path is `install_dir\Java\lib` (32-bit libraries)
\Java\lib64 (64-bit libraries)

Verify WebSphere MQ Base Java Installation [6]

This can be done by running the sample application

Configure queue manager for client connections:

(Needed if client is on different system)

1. Configure the queue manager to accept connection requests from the TCP/IP clients.
   - Start the queue manager using `STRMQM`.

   ```
   C:\Documents and Settings\raj>strmqm DB2MQ_DEFAULT_MQM
   There are 25 days left in the trial period for this copy of WebSphere MQ.
   WebSphere MQ queue manager 'DB2MQ_DEFAULT_MQM' starting.
   5 log records accessed on queue manager 'DB2MQ_DEFAULT_MQM' during the log replay phase.
   Log replay for queue manager 'DB2MQ_DEFAULT_MQM' complete.
   Transaction manager state recovered for queue manager 'DB2MQ_DEFAULT_MQM'.
   WebSphere MQ queue manager 'DB2MQ_DEFAULT_MQM' started.
   ```
   - Start runmqsc program using following command

     ```
     runmqsc [QMNAME]
     ```
   - Define a channel called `JAVA.CHANNEL` using following command

     ```
     DEF CHL('JAVA.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +DESCR('Sample channel for WebSphere MQ classes for Java')
     ```
   - Start a listener program with following command

     ```
     runmqlsr -t tcp [-m QMNAME] -p 1414
     ```

Use the MQIVP program to verify the installation and configuration for different connection modes of WebSphere MQ base Java.
Prerequisites to run MQIVP:

The user id running the application must have authorizations to connect to queue manager and inquire on the attributes of the queue manager object.

CLASSPATH environment variable must be updated as mentioned in above section.

Execute the program as mentioned below

C:\Program Files\IBM\WebSphere MQ\tools\wmqjava\samples> java MQIVP

Output:

```
Please enter the IP address of the MQ server: 
Please enter the queue manager name: 
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager
Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
```

Programming with WebSphere MQ Base Java [6]

The WebSphere MQ Application programming contains following verbs:

MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQPUT1, MQSET
The queue manager is represented as an object of class MQQueueManager and is used as mentioned below

//declare the object of type queue manager
MQQueueManager qman = new MQQueueManager();

//code

//disconnect from the queue manager
qman.disconnect();

The programming using the WebSphere MQ classes for Java is dependent on the connection modes used.

Client connections:

It is similar to WebSphere MQ C client, but does not support TCP/IP, MQBEGIN and does not read the MQ environment variables on startup.

Binding Mode:

Binding mode differs from client mode in following ways

- Most parameters provided by MQEnvironment class are ignored.
- Support the MQBEGIN call and fast path binding.
Sample Application Program to Queue Multiple Messages

The sample program provided by IBM is customized to demonstrate application that uses binding mode. Using the program connection is made to the queue manager to enqueue 10 messages in to DB2MQ_DEFAULT_Q and dequeue a message.

Find the code snippet in Appendix D.

The output of the program is
Oracle Streams AQ [1]

Oracle Streams AQ gives a database integrated message queuing functionality. It is built on top of Oracle Streams and leverages functions of Oracle Database so that the messages can be stored persistently, propagated between the queues on different systems and databases.

It has all operational benefits of high availability, scalability and reliability as it is implemented in the database tables. Also, standard database features such as recovery, restart, and security are supported by Oracle Streams AQ. Oracle Enterprise Manager can be used for monitoring the queues. Like any other database tables, queue tables can be imported and exported.

Oracle Streams AQ supports queue level access control for enqueue and dequeue operations. This protects the queues created in one schema from the applications running in a different schema.
Oracle Streams AQ in Integrated Application Environments [1]

Oracle Streams AQ can be used for message management and communication needed for application integration. In the integrated environments the messages are communicated between the applications, database servers and the users as shown below.

![Integrated Application Environment using Oracle Streams AQ](image)

Figure 6.1 Integrated Application Environment using Oracle Streams AQ [1]

Application integration involves the integration of heterogeneous messaging systems. Oracle streams AQ integrates with existing non-Oracle Database messaging systems like IBM WebSphere MQ through messaging gateway. Thus, allowing the existing WebSphere applications to integrate with Oracle Streams AQ environment easily.
Interfaces to Oracle AQ [1]

The Oracle Streams AQ functionality can be accessed through the following interfaces

1. PL/SQL using DBMS_AQ, DBMS_AQADM and DBMS_AQELM
2. Visual basic using Oracle objects for OLE
3. Java Messaging Service using oracle.jms java package
4. Internet access through HTTP(S)

Message Queuing in to Oracle AQ [1]

Create AQ queue Table:

The following script needs to be run to create an AQ Table. The queues are created under the queue table and the queue data is stored in this table. The AQ table creation is done by the AQ Administrator.

-- Create a queueTable for processing type: SYS.ANYDATA msgs

set echo ON

spool create_queue_table_mitchell_qtab_any.out

BEGIN

DBMS_AQADM.CREATE_QUEUE_TABLE(
queue_table => 'mitchell_qtab_any',
queue_payload_type => 'SYS.ANYDATA',
multiple_consumers => TRUE,
comment => 'Multiconsumer queueTable for type SYS.ANYDATA messages');
Create ANYDATA Queue:

The following script needs to be run by AQ_ADMIN user to create an ANYDATA queue. The queue is owned by the user, however the queue needs to be created by AQ_ADMIN.

```sql
BEGIN
DBMS_AQADM.CREATE_QUEUE(
queue_name => 'raj_queue_any',
queue_table => 'mitchell_qtab_any');
END;
/
```

Grant Privileges to AQ user: The queue privileges to enqueue messages should be given to the AQ user raj by AQ_ADMIN user. This can be done by running following script as AQ_ADMIN user.
--Grant privileges script

BEGIN

DBMS_AQADM.GRANT_QUEUE_PRIVILEGE ( 

    privilege => 'ALL',
    queue_name => 'aq_admin.raj_queue_any',
    grantee => 'RAJ',
    grant_option => FALSE);

END;
/

Start the Queue: The queue needs to be started before enqueuing the messages by running
the following script as AQ_ADMIN user.

--start queue script

BEGIN

DBMS_AQADM.START_QUEUE( 

    queue_name => 'raj_queue_any',
    enqueue => TRUE,
    dequeue => TRUE);

END;
/


Enqueue message in to the AQ:

The following script needs to be run to create a message in the queue rajgopal_queue_any. The user should have following privileges to execute the script

```
GRANT EXECUTE ON dbms_aq TO user
queue operation priv on queue aq_admin.rajgopal_queue_any

-- enqmessage.plsql
DECLARE
recipients DBMS_AQ.aq$_recipient_list_t;
enqueue_options DBMS_AQ.enqueue_options_t;
message_properties DBMS_AQ.message_properties_t;
message_handle RAW(16);
message ANYDATA;
BEGIN
recipients(1):= sys.aq$_agent('rajgopal',NULL,NULL);
message_properties.recipient_list := recipients;
message_properties.SENDER_ID :=
SYS.AQ$_AGENT('rajgopal',NULL,NULL);
message := ANYDATA.ConvertNumber('16');
DBMS_AQ.ENQUEUE(
queue_name => 'aq_admin.rajgopal_queue_any',
enqueue_options => enqueue_options,
message_properties => message_properties,
payload => message,
```
Validating the enqueue procedure:

The enqueue script can be verified by running the following SQL command as a privileged user.

```
SQL> select count (*), queue
    2   from aq$mitchell_qtab_any
    3   group by queue;

COUNT(*)  QUEUE
----------  ----------------------
    1   RAJ_QUEUE_ANY
```

Here the AQ Table mitchell_qtab_any houses the queue RAJ_QUEUE_ANY. The number of messages in the raj_queue_any is 1. The queue data can be viewed using the AQ$Queue_Table_Name view.

**AQ Security Aspects during the Enqueue Process**

1. When the enqmessage.plsql is executed the following error occurred
This is resolved by granting the user execute privileges on `dbms_aq` as mentioned below

```sql
GRANT EXECUTE ON dbms_aq TO <UserName>;
```

ii. After execute privileges are granted on `dbms_aq`, the execution of the `enqmessage.plsql` gave following error

```
DECLARE
ERROR at line 1:
ORA-24010: QUEUE RAJ.RAJ_QUEUE_ANY does not exist
ORA-06512: at "SYS.DBMS_AQ", line 169
ORA-06512: at line 9
```

This is resolved by providing privileges to the user to operate on the queue `aq_admin.raj_queue_any` by the `AQ_ADMIN` User.
The Oracle Streams AQ is used to propagation of messages between AQ queues, but there are situations where the messages need to be communicated to a non-oracle messaging system. This can be done using the messaging gateway [1]. Oracle messaging gateway enable communication between applications based on non-oracle messaging systems and oracle streams AQ [1].

Oracle messaging gateway is integrated in to the oracle streams AQ and oracle database, thus provides a reliable message delivery (guaranteed messaged delivery once and only once between streams AQ and non oracle messaging system) [1]. The messaging gateway in both 10g and 11g releases support integration with applications based on WebSphere MQ 6.0 and TIB/Rendezvous 7.2 [1].

Features [1]

Extends Oracle Streams AQ message propagation:

Messaging gateway propagates messages from oracle streams AQ to non oracle messaging systems applications and vice versa.

Support Java Messaging systems:

Messaging gateway propagates the messages between the Oracle Java messaging services (Oracle JMS) and WebSphere MQ Java messaging service (WebSphere MQ JMS).
Native message format support:

It supports the native message formats of the messaging systems. Oracle Stream AQ messages can be RAW or any ORACLE object type data. WebSphere MQ messages can be text or byte messages.

Message conversion:

Messaging gateway provides message conversions between AQ and the non oracle messaging systems either through the automatic routines provided by the messaging gateway or using custom message transformation functions.

Integration with oracle database:

Messaging gateway can be managed through the PL/SQL and its configuration is stored in the Oracle Database tables. Message propagation is carried out by the external process of the database server.

Guaranteed message delivery:

If the propagation source and propagation destination both support transactions, the messaging gateway guarantees that persistent messages are propagated exactly once. If they do not support the transactions or if the messages are not persistent then at-most-one propagation is guaranteed.

Security support:

It supports client authentication of oracle database and non oracle messaging system. It supports SSL for WebSphere MQ connection and WebSphere MQ JMS connections.
Multiple agent support:

Messaging gateway supports multiple agents for a given database. Thus, users can partition the propagation jobs based on functionality, organization or workload, and assign them to different messaging gateway agents.

Oracle Messaging Gateway Architecture [1]

Messaging gateway has two main components

- Administration package DBMS_MGWADM
- Messaging gateway Agent

![Figure 7.1 Messaging Gateway Architecture [1]](image-url)
Administrative package DBMS_MGWADM:

This component of messaging gateway provides an interface for creating named Messaging gateway agents, managing agents, creating messaging system links, registering non-Oracle queues, and setting up propagation jobs.

Users calls the procedures to configure the messaging agents dynamically (while the agents are running the procedures send the notifications which alters the configurations, although for some changes the agent needs to be shut down and restarted).

Oracle Messaging Gateway Agent:

The messaging gateway agent runs as an external process of the Oracle Database server and process propagation jobs. It is started and shutdown using STARTUP and SHUTDOWN procedures in DBMS_MGWADM package.

It has a multithreaded propagation engine and set of drivers for messaging systems. The agent schedules the propagation jobs and processes the propagation jobs concurrently. The polling thread in the agent periodically polls the source queues of enabled propagation jobs and assigns the worker threads to process the propagation jobs if messages are available. The drivers of the messaging systems act as the clients for the messaging operations.

Oracle Database

Messaging gateway is a feature of Oracle Database, which is managed through the PL/SQL administration package DBMS_MGWADM. All configuration and execution state information is stored in Oracle Database and can be access through database views.
Messaging gateway runs as an external procedure of the Oracle Database Server. Therefore it runs only when its associated database server is running.

Non-Oracle Messaging System

The messaging gateway agent connects to non-Oracle messaging system through messaging system links. Messaging system links are the communication channels to the non oracle messaging systems. The users can configure multiple system links to the same or different non-Oracle messaging systems using DBMS_MGWADM package. Queues in the non-oracle messaging systems, such as WebSphere MQ serve as the source or the destination for the messaging gateway. These queues are referred to as foreign queues. All foreign queues involved in the message propagation as source queues, destination queues or exception queues need to be registered at messaging gateway using the administrative package. The registration of the foreign queues include the information about the queue such as the messaging system links to access the queues, native name and its domain (queue or topic). However, corresponding physical queues must be present in their respective messaging systems.
Propagation Process

The propagation jobs are defined in the order the messages needs to be propagated. A propagation job defines the source queue, destination queue and other attributes which affect the processing of propagation job.

When the messaging gateway processes the propagation job, it dequeues the message from the source queue and enqueues the messages to the destination queue. During the propagation, the message is converted from its native format in the source messaging system to the native format in the destination messaging system.

If the agent fails to convert the message from the source format to the destination format, the agent moves the message from the source queue to the exception queue and continues with the propagation job.

To guarantee the reliable message delivery, messaging gateway requires logging queues which support transactions and persistent messages. The agent uses the logging queues to store the processing states of the propagation jobs so that it can restore from the failures.
Chapter 8

CONFIGURING ORACLE MESSAGING GATEWAY

In this chapter we discuss how to load and setup Oracle Messaging Gateway.

Prerequisites [1]

Messaging Gateway uses one Oracle scheduler job for each messaging gateway agent. If the value of JOB_QUEUE_PROCESSES database initialization parameter is a non-zero, it must be large enough to accommodate scheduler job for the agent.

Loading Database Objects in to Database [1]

Using SQL*Plus, run ORACLE_HOME/mgw/admin/catmgw.sql as user SYS as SYSDBA. The script loads all the database objects related to messaging gateway, including roles, tables, views, object types and PL/SQL packages.

All these objects are owned by SYS. The two important roles it creates are MGW_ADMINISTRATOR_ROLE and MGW_AGENT_ROLE.

Modifying listener.ora for External Procedure [1]

The listener.ora needs to be modified so that the messaging gateway PL/SQL packages can call the external procedures
Check whether the IPC (Inter process communication protocol) protocol address is set for the external procedure. Also, the static service information is added by setting the SID_DESC for the listener.

Example 8.1 Adding the static service and IPC protocol address to the listener [1]

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      # ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
      # ADDRESS = (PROTOCOL = TCP)(HOST = delphi.ecs.csus.edu)(PORT = 1521))
      # ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC)
      (ADDRESS = (PROTOCOL = TCP)(HOST = delphi.ecs.csus.edu)(PORT = 1521))
    )
  # Add a SID_DESC for external mgw agent
SID_LIST_LISTENER=
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = MGWEXTPROC)
      (PROGRAM = extproc)
      (ENVS="LD_LIBRARY_PATH=/opt/Oradb11g/app/oracle/product/11.1.0/db_1/jdk/jre/lib/i386:/opt/Oradb11g/app/oracle/product/11.1.0/db_1/jdk/jre/lib/i386/server:/opt/Oradb11g/app/oracle/product/11.1.0/db_1")
      (ORACLE_HOME=/opt/Oradb11g/app/oracle/product/11.1.0/db_1)
    )
  )
```

Modifying tnsnames.ora for External Procedure [1]

A net service name is an alias to connection descriptor used to access a net service such as a database across the network and it is a fixed value [8].

Add a net service name MGW_AGENT in tnsnames.ora whose connector descriptor matches the information configured in listener.ora as given in the example 8.2 below.
Example 8.2 Add MGW_AGENT net service in tnsnames.ora

```
MGW_AGENT =
  (DESCRIPTION=
    (ADDRESS_LIST= (ADDRESS= (PROTOCOL=IPC) (KEY=EXTPROC) )
    (CONNECT_DATA= (SID=MGW_EXTPROC) (PRESENTATION=RO)
    )
  )
```

Check in sqlnet.ora if the names.default_domain is used to set the default domain. If set, append the domain to the MGW_AGENT net service name in tnsnames.ora.

Setting up mgw.ora initialization file [1]

The messaging gateway default initialization file located at ORACLE_HOME/mgw/admin/mgw.ora is a text file. The external procedure uses the initialization file parameters to start the messaging gateway agent. Copy sample_mgw.ora present in ORACLE_HOME/mgw/admin to mgw.ora and modify it specific to your Oracle AQ environment.

Set the CLASSPATH environment variable to include the following:

- Java Runtime class: JRE_HOME/lib/rt.jar
- Oracle JDBC classes: ORACLE_HOME/jdbc/lib/ojdbc5.jar
- Oracle internationalization classes: ORACLE_HOME/jlib/orail8n.jar
- SQLJ runtime: ORACLE_HOME/sqlj/lib/runtime12.jar
- Java messaging service (JMS) interface: ORACLE_HOME/rdbms/jlib/jmscommon.jar
- Oracle JMS implementation classes: ORACLE_HOME/rdbms/jlib/aqapi.jar
- JAVA Transaction API: ORACLE_HOME/jlib/jta.jar
Also, include the classes needed by messaging gateway to access the Non-Oracle messaging systems. The required classes are discussed in Setting up for WebSphere MQ Base Java or JMS Section.

```
# Example 8.4

```

Create Oracle Messaging Gateway Administrator User [1]

This user is intended for administration of the messaging gateway. Create a user with **MGW_ADMINISTRATOR_ROLE** privileges as mentioned in example 8.3

Example 8.3 Creating Messaging gateway Administrator user [1]

```
CREATE USER mgw_admin IDENTIFIED BY admin_password;
GRANT CREATE SESSION to mgw_admin;
GRANT MGW_ADMINISTRATOR_ROLE to mgw_admin;
```

Create Oracle Messaging Gateway Agent User [1]

In order to establish a messaging gateway agent connection back to the database, a database user with **MGW_AGENT_ROLE** privileges must be created as shown in the example 8.4
Example 8.4 Creating Messaging gateway Agent user [1]

CREATE USER agent_user IDENTIFIED BY agent_password;
GRANT CREATE SESSION to agent_user;
GRANT MGW_AGENT_ROLE to agent_user;

Configure Oracle Messaging Gateway Connection Information

The administrator uses the DBMS_MGWADM.ALTER_AGENT to configure Messaging Gateway with agent’s username, password and database connect string used by the messaging gateway agent to connect to the database, as given in the example 8.5. Always specify a not NULL value for the database connect string when calling DBMS_MGWADM.ALTER_AGENT.

Example 8.5 Configure Messaging Gateway Configuration Information

-- Run this as MGW Admin user
BEGIN
DBMS_MGWADM.ALTER_AGENT(
agent_name => 'default_agent',
username => 'agent_user',
password => '******',
database => 'DELPHI');
END;
/
Setting up for WebSphere MQ Base Java [1]

The WebSphere MQ Client and WebSphere MQ Classes for Java need to be installed on the system where the messaging gateway agent runs.

Installation of Websphere MQ Client on Oracle server [9]:

Downloaded the WebSphere MQ 7.0 Client software from the IBM website (http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24019253) for Linux 32-bit Operating System. In this project the Oracle database server and messaging gateway is deployed on Linux version 2.6.27-7-server.

Create the /opt/mqm and /var/mqm directories in the file system. The Oracle Messaging Gateway configuration does not create these directories. The /opt/mqm directory is used for installation of WebSphere MQ product and the working data is stored in /var/mqm.

Create the userid and groupid for WebSphere MQ and set the users home directory to /var/mqm. In order to run the administrative commands such as create the queue manager or start queue manager the user id must be a member of mqm group.

Procedure for installing WebSphere MQ Client:

1. Login as root.

2. To view the license run ./mqlicense.sh –text and to accept the license run ./mqlicense.sh -accept . The license should be accepted before proceeding with the installation.

3. Run the following command to install the Runtime, Client and Java components
   
   rpm -ivh MQSeriesRuntime-6.0.0-0.i386.rpm MQSeriesClient-6.0.0-0.i386.rpm
   rpm –ivh MQSeriesJava-7.0.1-3.i386.rpm
4. Installation is complete.

Install WebSphere MQ Classes for Java:

Download the WebSphere MQ Classes for Java from the IBM Website (http://www-01.ibm.com/support/docview.wss?uid=swg24000668). Unpack the bundle and save the com.ibm.mq.pcf-6.1.jar in required directory. Update the CLASSPATH environment variable to reference the com.ibm.mq.pcf-6.1.jar. These classes are required for running the WebSphere MQ Base Java Applications.

In this report mq_home refers to location of the installed client, which is opt/mqm on Linux os.

Modify mgw.ora:

For WebSphere MQ Base Java interface, set the CLASSPATH to include the following

mq_home/java/lib/com.ibm.mq.jar
mq_home/java/lib/connector.jar

Verifying the Messaging Gateway Setup [1]

The messaging gateway configuration can be verified using following procedure through which we startup and shutdown the Messaging gateway agent:

1. Start database listeners for the external procedure and the listeners for the regular database connection.

2. Test the database connect string for the messaging gateway agent user by running
sqlplus mgw_agent/agent_password@agent_database, if successful the agent is able to connect to the database. In this project the agent_database is DELPHI.
3. Test the net service entry used to call the external procedure.
   Run sqlplus mgw_agent/agent_password@MGW_AGENT
   This should fail with ORA-28547: connection to server failed, probable Oracle
   Net admin error”. Any other error indicates that tnsnames.ora, listener.ora or both
   are not correct.

4. Connect as mgw_admin user and call DBMS_MGWADM.STARTUP to start the
   Messaging gateway agent.

   SQL> exec DBMS_MGWADM.STARTUP;

5. Using the MGW_GATEWAY view, check if the agent_status is changed to
   RUNNING and agent_ping is changed to REACHABLE as shown below

   SQL> select AGENT_STATUS,AGENT_PING,AGENT_JOB,AGENT_USER,
     2       AGENT_START_TIME from mgw_gateway;

   +-----------------+-----------------+--------+-------------------+-------------------------+
   | AGENT_STATUS   | AGENT_PING      | AGENT_JOB | AGENT_USER        | AGENT_START_TIME        |
   +-----------------+-----------------+--------+-------------------+-------------------------+
   | RUNNING         | REACHABLE       | AGENT_USER |                  | 28-NOV-10 03:40:59     |

6. Connect as mgw_admin and call DBMS_MGWADM SHUTDOWN to shutdown
   the agent.

   SQL> BEGIN
     2   DBMS_MGWADM.SHUTDOWN;
     3   END;
     4  /

7. Using MGW_GATEWAY view, check if the agent_status is changed to
   NOT_STARTED.

   SQL> select AGENT_STATUS,AGENT_PING,AGENT_JOB,AGENT_USER,
     2       AGENT_START_TIME from mgw_gateway;
Once the Oracle Messaging gateway is loaded and set up, it can be configured as needed. DBMS_MGWADM.ALTER_AGENT command can be used to set the username, password, Database specifier and the connection type the messaging gateway agent uses for creating a database connection.

Configuring Messaging Gateway Agent [1]:
The messages are propagated between Oracle Streams AQ and non-Oracle messaging system by the messaging gateway agent. Oracle messaging gateway supports multiple agents.

The messaging gateway default agent, DEFAULT_AGENT, is created when the messaging gateway is installed. The messaging gateway agent configuration can be modified using DBMS_MGWADM.ALTER_AGENT. Before the agent can be started, the configuration of the agent should be done by database user who is granted MGW_AGENT_ROLE.

The agent can be created using the following SQL command

SQL> exec DBMS_MGWADM.CREATE_AGENT(
    agent_name => 'agent1',
    username => 'mgwagent',
    password => 'mgwagent_password',
    database => 'database');
The resource (memory, propagation threads) limits can be set using 
DBMS_MGWADM.ALTER_AGENT. However, these resources cannot be modified 
while the agent is running.

Starting the Oracle Messaging Gateway Agent:
The following commands can be used to start the agent after it has been configured

To start default agent

SQL> exec DBMS_MGWADM.STARTUP;

To start any other agent, such as agent1

SQL> exec DBMS_MGWADM.STARTUP ('agent1');

Shutting down the Oracle Messaging Gateway Agent:
The following commands can be used to shutdown the agent

To shut down default agent

SQL> exec DBMS_MGWADM.SHUTDOWN;

To shut down agent1

SQL> exec DBMS_MGWADM.SHUTDOWN ('agent1');

Configuring Messaging System Links [1]

The messaging gateway agent can run as a client for non–Oracle messaging system
and communicate with non-Oracle messaging system using messaging system links.
To configure the messaging system links the agent should be provided with non-
Oracle messaging system configuration information. An agent is associated with each
messaging system link and is responsible for the processing of propagation jobs to registered foreign queues.

Creating WebSphere MQ Base Java link:

The system messaging link to WebSphere MQ Base Java can be created by calling DBMS_MGWADM.CREATE_MSGSYSTEM_LINK with following information

1. Interface type: DBMS_MGWADM.MQSERIES_BASE_JAVA_INTERFACE

2. WebSphere MQ connection information:

   - Host name and port number of the WebSphere MQ Server
   - Queue manager name
   - Channel name
   - Username and password used to authenticate to MQ Server

The communication between the WebSphere MQ Client and the Server is done through the Message Queuing Interface Channels. In WebSphere MQ Server, create a Server Connection Channel (similar to SYSTEM.DEF.SVRCONN object) for communicating with the Queue manager using the following command or using the MQ Explorer (GUI Based interface to WebSphere MQ Server).

```
define channel('S_Xtreme') chltype(SVRCONN)
```

The Server Connection Channel created is used as the channel name.

3. Maximum number of message connections allowed.

4. Log queue names for inbound and outbound propagation. In this project we are doing an outbound propagation. Create a local queue in WebSphere MQ Server under the queue manager as outbound log queue. Similarly for inbound
propagation to Oracle AQ from WebSphere MQ create a local log queue in WebSphere MQ.

5. Optional information like send, receive, and security exits, Character sets

When configuring the messaging system links for the non-Oracle messaging systems which support transactions and persistent messages like WebSphere MQ, the native name of the log queues for inbound and outbound propagation must be specified to guarantee exactly once message delivery. These log queues are only used by the messaging gateway agent.

Example 8.6 configures a WebSphere MQ Base Java link named mq_link_2. This link is configured to use the WebSphere MQ queue manager named QM_Xtreme on host myhost and port 1414, using the WebSphere MQ channel S_Xtreme.

The messaging gateway default agent named DEFAULT_AGENT is responsible for the link and all propagation jobs defined on the link.

Example 8.6 Configuring a WebSphere MQ Base Java Link

```
set echo ON
set serveroutput ON
DECLARE
  v_options sys.mgw_properties;
  v_prop sys.mgw_mqseries_properties;
BEGIN
  v_prop := sys.mgw_mqseries_properties.construct();
```
v_prop.interface_type := dbms_mgwadm.MQSERIES_BASE_JAVA_INTERFACE;
v_prop.max_connections := 1;
v_prop.username := null;
v_prop.password := null;
v_prop.hostname := 'myhost';
v_prop.port := 1414;
v_prop.channel := 'S_Xtreme';
v_prop.queue_manager := 'QM_Xtreme';
v_prop.outbound_log_queue := 'OUTBOUND_LOG_QUEUE';
dbms_mgwadm.create_msgsystem_link(
linkname => 'mqlink_2', agent_name=>'default_agent', properties => v_prop,
options => v_options );
END;
/
Views for Messaging System Links [1]:
The links created and their type can be viewed using MGW_LINKS view as mentioned below

```
SQL> select link_name, link_type from MGW_LINKS;

<table>
<thead>
<tr>
<th>LINK_NAME</th>
<th>LINK_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQLINK_2</td>
<td>MQSERIES</td>
</tr>
</tbody>
</table>
```
The MGW_MQSERIES_LINKS view can be used to check the WebSphere MQ messaging system type specific configuration information as shown below
Removing Message System Links [1]:

All registered queues associated with system link must be unregistered before removing the link. The messaging gateway link to non-Oracle messaging system can be removed using `DBMS_MGWADM.REMOVE_MSGSYSTEM_LINK` as shown below.

```sql
BEGIN
    dbms_mgwadm.remove_msgsystem_link(linkname => 'mqlink_2');
END;
```

Configuring Non-Oracle Messaging System Queues [1]

The non-Oracle messaging system queues used during the propagation needs to be registered through the Messaging gateway administration interface. The registered queue can be a source queue, destination queue or exception queue (the undelivered messages are moved to exception queue). The Oracle AQ queues involved in the propagation should not be registered.

Registering a Non-Oracle Queue:

The registration of a queue provides information about the queue to the messaging gateway agent. Using this information the messaging gateway agent accesses the non-Oracle queue.
The non-Oracle queue can be registered using DBMS_MGWADM.REGISTER_FOREIGN_QUEUE. The registration of queue does not create a physical queue.

The following information is needed for registering a queue:

1. Name of messaging system link used to access the queue
2. Native name of the queue (its name in the non-Oracle messaging system WebSphere MQ)
3. Domain of the queue DBMS_MGWADM.DOMAIN_QUEUE for point to point messaging

Example 8.7 Registering a WebSphere MQ Base Java Queue [1]

```sql
BEGIN
DBMS_MGWADM.REGISTER_FOREIGN_QUEUE(
  name => 'destq',
  linkname => 'mclink_2',
  provider_queue => 'MQ_QUEUE',
  domain => dbms_mgwadm.DOMAIN_QUEUE);
END;
/
```

The domain can be DBMS_MGWADM.DOMAIN_QUEUE or NULL, because only point to point queues are supported by WebSphere MQ [1].

View for Registered Non-Oracle Queues:

The MGW_FOREIGN_QUEUES view can be used to check the registered non-Oracle queues and the links used by them by running following command
In order to propagate messages between Oracle AQ and non-Oracle messaging system a propagation job must be defined. Each propagation job has a unique propagation type, source, and destination triplet. Messaging gateway supports bidirectional message propagation. The outbound messages are propagated from Oracle AQ to non-Oracle messaging system and the inbound messages are propagated from non-Oracle messaging system to Oracle AQ.

In this project, the outbound propagation source is a queue (point-to-point), thus the messaging gateway agent moves all messages from the source queue to the destination queue. When the message is propagated, the messaging gateway agent converts message from the format in source messaging system to the format of the destination messaging system.

Creating an Oracle Messaging Gateway Propagation Job [1]:

The Messaging gateway propagation jobs are created by

DBMS_MGWADM.CREATE_JOB as mentioned in the example 8.8.
Example 8.8 Creating a Messaging Gateway Propagation Job

```
BEGIN
DBMS_MGWADM.CREATE_JOB(
  job_name => 'job_aq2mq',
  propagation_type => DBMS_MGWADM.OUTBOUND_PROPAGATION,
  source => 'aq_admin.mgw_queue',
  destination => 'destq@mqlink_2');
END;
/
```

The message propagation job is enabled by default when it is created using by
DBMS_MGWADM.CREATE_JOB. However, the propagation job can be enabled or
disabled using DBMS_MGWADM.ENABLE_JOB or
DBMS_MGWADM.DISABLE_JOB respectively as mentioned in the example 8.9 and
example 8.10.

Example 8.9 Enable a propagation job aq2mq_job

```
BEGIN
DBMS_MGWADM.ENABLE_JOB(job_name => 'job_aq2mq');
END;
/
```

Example 8.10 Disable a propagation job aq2mq_job

```
BEGIN
DBMS_MGWADM.DISABLE_JOB(job_name => 'job_aq2mq');
END;
/
```

The MGW_JOBS view can be used to display the scheduled jobs and their status as
mentioned below

```
SQL> select job_name, link_name, failures, last_error_time, last_error_msg
  2  from mgw_jobs;
```
After the message propagation job is scheduled, perform the following test procedure to propagate a message from AQ queue to WebSphere MQ queue through the Oracle Messaging Gateway.

Test Message Propagation from Oracle AQ to WebSphere MQ

1. Create a queue table with queue payload type as sys.mgw_basic_msg as shown below

```sql
BEGIN
DBMS_AQADM.CREATE_QUEUE_TABLE(
    queue_table => 'mgw_qtab',
    queue_payload_type => 'sys.mgw_basic_msg_t',
    multiple_consumers => TRUE,
    comment => 'Multiconsumer queueTable for basic data type messages');
END;
/
```

2. Create an AQ queue corresponding to mgw_qtab queue table.

```sql
BEGIN
DBMS_AQADM.CREATE_QUEUE(
    queue_name => 'mgw_queue',
    queue_table => 'mgw_qtab');
END;
/
SHOW_ERRORS
```
3. Grant queue privileges to enqueue and dequeue the messages for the user

```
BEGIN
    DBMS_AQADM.GRANT_QUEUE_PRIVILEGE(
        privilege => 'ALL',
        queue_name => 'aq_admin.raj_queue_any',
        grantee => 'RAJ',
        grant_option => FALSE);
END;
/
```

4. Start the queue using following script

```
BEGIN
    DBMS_AQADM.START_QUEUE(
        queue_name => 'mgw_queue',
        enqueue => TRUE,
        dequeue => TRUE);
END;
/
```

5. Enqueue a message containing single text string ‘9167’ in to the AQ queue using following script

```
DECLARE
    -- recipients DBMS_AQ.mq_recipient_list_t;
    -- enqueue_options DBMS_AQ.enqueue_options_t;
    -- message_properties DBMS_AQ.message_properties_t;
    -- message_handle RAQ_Handle;
    -- header sys.mgw_message_header_t;
    -- text_body sys.mgw_text_value_t;
    purge>VARCHAR2(1000);
BEGIN
    -- recipients Probably not needed in Gateway enqueue (?)
    -- recipients(1) := SYSA.Q_AGENT('raj',NULL,NULL);
    -- message_properties.recipient_list := recipients;
    -- message_properties.MESSAGE_ID := SYSA.AQG_AGENT('raj',NULL,NULL);
    msg := '9167';
    -- Create the text body.
    text_body := sys.mgw_text_value_t(msg, NULL);
    -- Construct the message from its constituent parts
    message_payload := sys.mgw_basic_msg_t(NULL, text_body, NULL);
    DBMS_AQ.ENQUEUE(
        queue_name => 'aq_admin.mgw_queue',
        enqueue_options => enqueue_options,
        message_properties => message_properties,
        payload => message_payload,
        msgid => message_handle);
    COMMIT;
END;
/
```
Unlike Oracle AQ to Oracle AQ propagation, the recipients need not be specified explicitly in this scenario.

6. Create and enable the propagation Job as mentioned in the previous section. The propagation jobs starts polling (checking) for any new messages in the source AQ queue (mgw_queue), after a message is enqueued to mgw_queue. The message will be automatically propagated to the WebSphere MQ local queue through the Messaging System Link.

7. Browse messages in the WebSphere MQ destination queue MQ_QUEUE where the message gets enqueued as shown below

<table>
<thead>
<tr>
<th>Position</th>
<th>Put date/time</th>
<th>User Identifier</th>
<th>Put application name</th>
<th>Format</th>
<th>Data length</th>
<th>Message data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nov 27, 2016 7:19 PM</td>
<td>MUSR_MQADMIN</td>
<td>WebSphere MQ Client for Java</td>
<td>MQSTR</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Nov 27, 2016 7:20 PM</td>
<td>MUSR_MQADMIN</td>
<td>WebSphere MQ Client for Java</td>
<td>MQSTR</td>
<td>4</td>
<td>9167</td>
</tr>
</tbody>
</table>

8. Therefore we see that the message is successfully propagated from Oracle AQ queue to WebSphere MQ queue through the Oracle Messaging Gateway.
Chapter 9

CONCLUSION

This project discusses the message propagation between the heterogeneous systems Oracle AQ and WebSphere MQ messaging system by using the Oracle Messaging Gateway functionality.

The Oracle messaging Gateway supports persistent messages and transactions, thus it can guarantee one time delivery of the messages. The project led to an in depth understanding of Message queuing configuration and operation features both in Oracle AQ and WebSphere MQ. Also, dealt with security aspects of message queuing and Oracle messaging gateway architecture. In the process of establishing a message propagation between Oracle AQ and WebSphere MQ, the author came across additional tools which can be used to validate the configuration like WebSphere MQ load/Unload tool with which we can load multiple messages from text file to the queue and vice versa. Deployed WebSphere MQ Base Java classes on WebSphere MQ server and customized a sample program to enqueue messages into the WebSphere MQ queue.

With information exchange being the primary need of the businesses, the message propagation functionality provided by Oracle Messaging Gateway has a greater scope supporting heterogeneous messaging systems. Oracle Messaging Gateway gives a reliable database integrated messaging solution to integrate businesses using different messaging and storage systems. In addition, both the enqueue and dequeue messages are
stored as table rows in Oracle queue table. This approach allows the SQL capabilities to be utilized by message processing.

Future Work

The messaging functionality discussed in this project can be further extended by exploring the message conversion features provided by the Oracle Messaging Gateway, publish/subscribe messaging and possibly other scenarios. This project is dealt with point-to-point communication between the queues.

In publish/subscribe communication, when a message is published on a topic, it is propagated to multiple queues subscribed to the topic.

Another generalization would be implementing gateway messaging between Oracle AQ and WebSphere MQ where the two sides are not operating on same local network, and are widely separated geographically.
APPENDIX A

Installation of DB2 Express C and Configuration Steps

Choose Install a product.
Following are the components included in the typical install:

Choose to install the product and create the response file.
This user will be used to setup and run the instance and services and should be part of the local administrator group.
The DB2 Instance Configuration

The following instances will be created during installation. You can customize the configurations by clicking on the Configure button.

DB2 Instances:

Configure...
Summary of the installation

DB2 Setup - DB2 Express-C - DB2COPY1

Start copying files and create response file

The DB2 Setup wizard has enough information to create the response file and start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, type the response file name and click Finish to begin copying files.

Current settings:

- Product to install: DB2 Express-C - DB2COPY1
- Installation type: Typical
- DB2 copy name: DB2COPY1
- Set as default DB2 copy: Yes
- Set as default IBM database client interface copy: Yes

Selected features:
- DB2 Update Service
- Base application development tools
- Base client support
- Configuration Assistant
- Control Center

DB2 Setup - DB2 Express-C - DB2COPY1

Start copying files and create response file

The DB2 Setup wizard has enough information to create the response file and start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, type the response file name and click Finish to begin copying files.

Current settings:

- Control Center
- IBM Data Server Provider for .NET
- First Steps
- JDBC Support
- DB2 LDAP support
- MIGR 2.8
- ODBC Support
- OLE DB Support
- Sample database source
- SQLJ Support
- DB2 WMI Provider

Languages:
- English
Setup is complete:

DB2 Setup wizard has finished copying files to your computer and has completed all the required system configuration tasks. Shut down all software programs running on the system now. The programs can then be reinstalled and DB2 will be ready for use. The install log is located in C:\Documents and Settings\fadath\Documents\IBM\DB2\log\DB2Setup-1.0.0.1.log. Consult the log file to ensure that all tasks completed successfully.

If you have not already done so, it is recommended that you complete the post-install steps after installation:

Required steps:

- You have enabled DB2 extended Windows security. You must add DB2 users that need to run DB2 local applications or tools to either the DB2 administrators group or DB2 users group.
- You can connect to the DB2 instance "DB2" using the port number "50000". Record it for future reference.

Click Finish to exit the DB2 Setup wizard.
APPENDIX B

IBM WEBSPHEREMQ Installation and Configuration

No action is needed, the WebSphere MQ administrator account is created automatically and the installer starts.
Set up the default configuration for WebSphere MQ.
Join default cluster
Identify whether the cluster repository is on this or another computer.

The first computer in the default cluster DEFAULT CLUSTER holds the repository for the cluster.
A repository is a queue manager in the cluster that holds information about all the other members.

Is this the first computer in the default cluster?
- Yes, make it the repository for the cluster.
- No, another computer has already joined the cluster as the repository.
- Don't know.

If this computer uses DHCP (dynamic allocation of IP address), then you should not put a repository on it. If the IP address changes, either queue managers, even on this computer, will no longer be able to find the repository. However, if this will be the only queue manager in the cluster, then the change of IP address will not have an effect on the repository.

Repository Location
Location of the repository computer.

The computer selected to hold the repository for the default cluster can be configured to either obtain an IP address automatically from a DHCP server or have a fixed IP address.

If this computer is configured to obtain an IP address automatically from a DHCP server, then is there another computer with a fixed IP address, that can be used to hold the repository?
- Yes
- No
- Not applicable

If you want to continue using this computer as the repository, click Next. Remember that other members of the cluster might have to be reconfigured when the IP address of this computer changes.
APPENDIX C

Installation of Application Messaging Interface
amtsamp.tst and amtsdfts.tst are run

WebSphere MQ AMI Objects

Would you like to run the scripts now (a running default Queue Manager is required)?

Yes  No

To run the samples, copy the sample repository file, amtsamp.xml, into {(installed location)}/amts. The default location is defined in AMT_DATA_PATH.
import com.ibm.mq.MQException;
import com.ibm.mq.MQGetMessageOptions;
import com.ibm.mq.MQMessage;
import com.ibm.mq.MQPutMessageOptions;
import com.ibm.mq.MQQueue;
import com.ibm.mq.MQQueueManager;
import com.ibm.mq.constants.MQConstants;
* Simple example program

*/

public class MQSamplenew {

// code identifier

static final String sccsid = "@(#) samples/wmqjava/MQSample.java, jmscc.samples, k000, k000-L090724 1.8 09/04/18 08:05:59";

// define the name of the QueueManager changed to default queueManager

// private static final String qManager = "my_queue_manager";

/**************** start of change by raj*******************/

private static final String qManager = "DB2MQ_DEFAULT_MQM";

/**************** End of change by raj*******************/

// and define the name of the Queue

//private static final String qName = "SYSTEM.DEFAULT.LOCAL.QUEUE";

private static final String qName = "DB2MQ_DEFAULT_Q";

/**
 * Main entry point
 *
 * @param args - command line arguments (ignored)
 *
 */

public static void main(String args[]) {

try {

    int i;

    // string str;
// Create a connection to the QueueManager

System.out.println("Connecting to queue manager: " + qManager);

MQQueueManager qMgr = new MQQueueManager(qManager);

// Set up the options on the queue we wish to open

int openOptions = MQConstants.MQOO_INPUT_AS_Q_DEF | MQConstants.MQOO_OUTPUT;

// Now specify the queue that we wish to open and the open options

System.out.println("Accessing queue: " + qName);

MQQueue queue = qMgr.accessQueue(qName, openOptions);

/**************** start of change by raj*************

for ( i=0; i<=9 ; i++)
{

// Define a simple WebSphere MQ Message ...

MQMessage msg = new MQMessage();

// ... and write some text in UTF8 format

msg.writeUTF("Hello, World!%d"+i);

// Specify the default put message options

MQPutMessageOptions pmo = new MQPutMessageOptions();

// Put the message to the queue

System.out.println("Sending a message...");

queue.put(msg, pmo);
// Now get the message back again. First define a WebSphere MQ
// message
// to receive the data
}

//**************** End of change by raj**********************

MQMessage rcvMessage = new MQMessage();

// Specify default get message options
MQGetMessageOptions gmo = new MQGetMessageOptions();

// Get the message off the queue.
System.out.println("...and getting the message back again");
queue.get(rcvMessage, gmo);

// And display the message text...
String msgText = rcvMessage.readUTF();

//System.out.println("...and getting the message back again");
queue.get(rcvMessage, gmo);

// Close the queue
System.out.println("Closing the queue");
queue.close();

// Disconnect from the QueueManager
System.out.println("Disconnecting from the Queue Manager");
qMgr.disconnect();

System.out.println("Done!");

}
catch (MQException ex) {

    System.out.println("A WebSphere MQ Error occurred : Completion Code " + 
    ex.completionCode
    + " Reason Code " + ex.reasonCode);

    ex.printStackTrace();

    for (Throwable t = ex.getCause(); t != null; t = t.getCause()) {
        System.out.println("... Caused by ");
        t.printStackTrace();
    }
}

} catch (java.io.IOException ex) {

    System.out.println("An IOException occurred whilst writing to the message buffer: " 
    + ex);
}

return;

}


[6] [Online]. Available:

[7] [Online]. Available:

[8] [Online]. Available:
http://www.adp-gmbh.ch/ora/network/net_service_name.html

[9] [Online]. Available: