GoldenGate for Oracle to Java Messaging System (JMS) on ActiveMQ

Objective

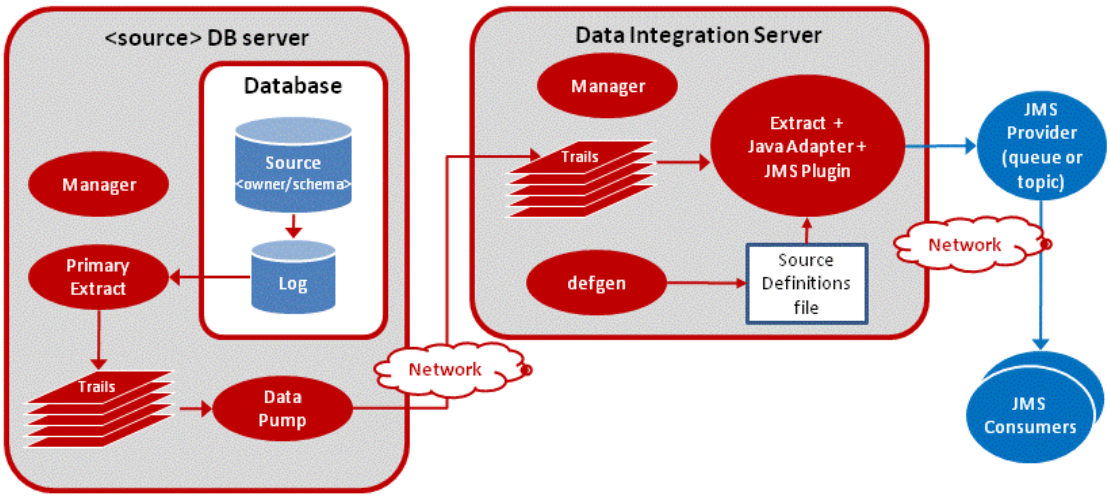
Upon completion of this lesson, you will be able to take DML transactions from an Oracle database and send them to an Apache ActiveMQ JMS queue using GoldenGate.

During this lesson, you will learn how to:

- Prepare your user environment.
- Configure and start the change capture process of database operations.
- Configure and start the change delivery process of database operations.

Oracle to JMS configuration

The following diagram illustrates GoldenGate a configuration with Oracle source data being replicated to a JMS target on an ActiveMQ server.



Overview of Tasks

Prepare the Environment

In order to execute this lesson, the GoldenGate application must be installed on both the source and Data Integration Server systems, and a JMS provider needs to be installed as well. The JMS server can be on the same machine as the Data Integration Server, or can be on a different machine. The installation includes a sample database and scripts to generate initial data as well as subsequent update operations. The source tables are created and loaded with initial data. The GoldenGate Manager processes are also started so that other processes may be configured and started.

The JMS provider used here is Apache ActiveMQ version 5.5.0, which can be downloaded from http://activemq.apache.org/ . Different versions of ActiveMQ handle logging differently.

Configure Change Capture

For log-based Oracle capture, the capture process is configured to capture change data directly from the Oracle online redo logs or archive logs and store the changes in queues known as GoldenGate trails.

Configure Change Delivery

Once the tables have been initially loaded with data and the table definitions file has been copied to the Data Integration Server, the delivery process is configured to deliver the captured change data to the target JMS queues or topics as XML messages.

Exercise 1.
Prepare the Environment

Objective

The goals of this exercise are to:

- Configure and start the Manager processes
- Create and load practice data to Oracle DB tables
- Add supplemental logging
- Create table column definitions file

Prepare your Oracle source environment

1. Configure the Manager process on the source

On the **<source>** system, create the Manager parameter (`.prm`) file and specify the port it should use. Keywords in `.prm` files are not case-sensitive, but directories are.

- Create the Manager parameter file.

```
Shell> cd <ogg_install>
Shell> ggsci
GGSCI> EDIT PARAMS MGR
```

- Use the editor to assign a port.

```
--GoldenGate Manager parameter file
Port <port>
PurgeOldExtracts dirdat/*, UseCheckPoints
```

- Start the Manager.

```
GGSCI> START MGR
```

- Verify that the Manager has started.

```
GGSCI> INFO MGR
GGSCI> exit
```

2. Create the source tables and load the initial data.

Using SQL*Plus, create and populate the `TCUSTMER` and `TCUSTORD` tables by running the **demo_ora_create.sql** and **demo_ora_insert.sql** files found in the install directory.

Execute the following commands on the **<source>** system.

```
Shell> cd <ogg_install>
Shell> sqlplus <login>/<password>
SQL> @demo_ora_create
SQL> @demo_ora_insert
```

Verify the results:

```
SQL> select * from tcustmer;
SQL> select * from tcustord;
SQL> exit
```

3. Add supplemental logging

The following steps are required only when GoldenGate will be capturing data from the server. Therefore it is only on your **<source>** environment.

- Turn on supplemental logging at the database level.

  This command requires the ALTER DATABASE permissions.

  ```
  Shell> sqlplus <login>/<password>

  SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
  ```

- Switch to the next redo log file.

  ```
  SQL> ALTER SYSTEM SWITCH LOGFILE;
  SQL> EXIT
  ```

- Using GGSCI, log in to the database on the **<source>** and turn on supplemental logging for the TCUSTMER and TCUSTORD tables.

  ```
  Shell> ggsci
  GGSCI> DBLOGIN USERID <login>, PASSWORD <password>
  GGSCI> ADD TRANDATA <schema>.TCUSTMER
  GGSCI> ADD TRANDATA <schema>.TCUSTORD
  ```

  Verify that supplemental logging has been turned on for these tables.

  ```
  GGSCI> INFO TRANDATA <schema>.TCUST*
  ```

4. Create the table definitions (source-defs) file

- Specify the properties file that the column definition generation process should use. Note the semi-colon on the last line.

  ```
  GGSCI> EDIT PARAMS defgen
  ```

```
-- GoldenGate defgen Parameter file
USERID <login>, PASSWORD <password>
defsfile dirdef/twest.def
TABLE <schema>.TCUST*;
```

- Run `defgen` on the database **<source>** system to create the table column definitions file. Note the report file name is expected to be UPPER case.

```
GGSCI> exit

Shell> defgen paramfile dirprm/defgen.prm reportfile
dirrpt/DEFGEN.rpt
```

- Copy the files from the **<source>** system to the **<target>** host. If you are using a NAS, or the source and target "hosts" are all on one physical host, then use copy. If they are on separate hosts, use `ftp` or some physical media to transfer the files.

```
Shell> cp /<source>/dirdef/twest.def
/<target>/dirdef/twest.def

Shell> cp /<source>/dirprm/defgen.prm
/<target>/dirprm/defgen.prm
```

Prepare your ActiveMQ target environment

1. Install Oracle GoldenGate for Java

There are two parts to Oracle GoldenGate Application Adapters for JMS Media Pack, (specifically Oracle GoldenGate for Java version 11.1.1.0.0) that supports the JMS functionality. When you unzip it, you get two files:

- `ggs_JavaAdapter_xxxxxxxxxxx.tar`
- `ggs_xxxxx_Generic_xxxxxxxxx.zip`

a.) If Oracle GoldenGate is already installed for a particular database, skip to the next step. If not, then install "Generic" Oracle GoldenGate in the `<ogg_install>` directory.

b.) Install the Oracle GoldenGate Java Adapter. Put it in the same directory as the rest of GoldenGate. Untar and unzip the file into the `<ogg_install>` directory. The directory structure (with new files in bold) will look similar to:

```
<ogg_install>
   |---extract
   |---ggsci
   |---mgr
   |---libggjava_ue.so     (Windows ggjava_ue.dll)
   |---libggjava_vam.so    (Windows ggjava_vam.dll)
   |---ggjava/
         |-----ggjava.jar
         |-----resources/...
```

2. Install ActiveMQ

You can install ActiveMQ anywhere by simply unzipping and untarring it. The installation directory will be referred to as `<amq_inst>`.

3. Create the javaue.properties file

Using your editor of choice, create the `javaue.properties` file. Everything should start in column one of the file. If you see something indented, not in column one, then it is a continuation of the previous line.

```
### javaue.properties (GoldenGate Java Extract properties file)
gg.handlerlist=myjms1

java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInit
   ialContextFactory
java.naming.provider.url=tcp://localhost:61616

gg.handler.myjms1.type=jms
gg.handler.myjms1.destination=dynamicQueues/testQueue123
gg.handler.myjms1.connectionFactory=ConnectionFactory
gg.handler.myjms1.format=xml

### native library config ###
goldengate.userexit.nochkpt=TRUE
goldengate.userexit.timestamp=utc

goldengate.log.logname=cuserexit
goldengate.log.level=INFO
goldengate.log.tofile=true

goldengate.userexit.writers=javawriter
#javawriter.stats.time=3600
#javawriter.stats.numrecs=10000
javawriter.stats.full=TRUE
javawriter.stats.display=TRUE
javawriter.bootoptions=-Xmx32m -Xms32m
   -Djava.class.path=ggjava/ggjava.jar:dirprm:<amq_inst>/activem
   q-all-5.5.0.jar:<amq_inst>/lib/optional/log4j-1.2.14.jar:<amq
   _inst>/lib/optional/slf4j-log4j12-1.5.11.jar -Dlog4j.configur
   ation=log4j.properties
```

Each property should be on a single line. There is a space after `–Xms32m`, but no returns, and no breaks for the rest of the `bootoptions`. Make sure that the

`java.class.path` separators are colons "**:**" for UNIX (as shown above) and semi-colons "**;**" for Windows.

The syntax for naming the JMS parts is:

```
gg.handlerlist=<somename_1>,<somename_2>,…,<somename_n>
```

and then:

```
gg.handler.<somename_1>.<property_a>=<value_w>
gg.handler.<somename_1>.<property_b>=<value_x>
gg.handler.<somename_2>.<property_a>=<value_y>
gg.handler.<somename_2>.<property_b>=<value_z>
```

Note that `somename` is arbitrary (in this example, `myjms1`) and need not match anything in ActiveMQ. Property names are case-sensitive. These conventions allow for multiple handlers to be defined in a single properties file. Handlers not included in the `gg.handlerlist` are ignored. See the Java application log file for warning and additional details when there are configuration errors.

4. Verify the PATHs and CLASSPATHs

Execute the following tasks on the **<target>** system.

- If UNIX, you may have to add `.` (dot to pick up the current directory) to the `$LD_LIBRARY_PATH` to pick up the `libggjava*.so` files.
- Add `<amq_inst>/activemq-all-5.5.0.jar` to the `CLASSPATH`.
- Make sure that `<ogg_install>` is in the `PATH`.
- Remember that Windows separates `PATH1;PATH2;PATH3` with semi-colons, and UNIX separates `PATH1:PATH2:PATH3` with colons.
- The library search path must be set before "mgr" is started. The "mgr" process starts the "extract" process, and passes its environment, including the library search path, to "extract". On Windows, if running "mgr" as a service, you may have to delete and re-add the service (or reboot) so the updated setting for PATH is used.

5. Configure the Manager process on the Data Integration Server

Execute the following commands on the Data Integration Server.

- Start the command interface

```
Shell> cd <ogg_install>
Shell> ggsci
```

- Specify the port that the target Manager should use.

```
GGSCI> EDIT PARAMS MGR
```

```
-- GoldenGate Manager Parameter file
PORT <target_port>
PURGEOLDEXTRACTS dirdat/*, USECHECKPOINTS
```

- Start Manager

  ```
  GGSCI> START MANAGER
  ```

  Verify the results:

  ```
  GGSCI> INFO MANAGER
  ```

6. Create the javaue parameter file

Specify the properties file that the extract process should use.

```
GGSCI> EDIT PARAMS javaue
```

```
-- GoldenGate Java Extract Parameter file
EXTRACT javaue
setEnv ( GGS_USEREXIT_CONF = "dirprm/javaue.properties" )
GetEnv (JAVA_HOME)
GetEnv (PATH)
GetEnv (LD_LIBRARY_PATH)
GetEnv (LIBPATH)
CUserExit libggjava_ue.so CUSEREXIT PASSTHRU INCLUDEUPDATEBEFORES
sourceDefs ./dirdef/twest.def
getUpdateBefores
TABLE <schema>.TCUST*;
```

Be mindful of the last semi-colon. It must be only on the last line. None of the keywords are case-sensitive, so `GetEnv` is the same as `GETENV` and `getenv`. You can `GetEnv` all kinds of other system and application environment variables for debug and audit purposes, but you must *NOT* have any whitespace inside the parentheses. On the other hand, the `SetEnv` *MUST* have the whitespace as shown.

Exercise 2.
Configure Change Capture

Objective

The goals of this exercise are to:

- Configure and add the Extract process that will capture changes.
- Add the trail that will store the changes.
- Start the Extract process.

Configure change capture

1. Create the Extract parameter file

```
GGSCI> edit param extwest
```

On the **<source>** system, add the following commands to the parameter file.

```
EXTRACT extwest
USERID <login>, PASSWORD <password>
EXTTRAIL dirdat/ew
RecoveryOptions OverwriteMode
TABLE <schema>.TCUST*;
```

Note: The RecoveryOptions you set in the primary extract must match the `RecoveryOptions` in the `pump.prm` file that generates the target trail.

2. Add the Extract group

Execute the following command on the **<source>** system to add an Extract group named EORA<unique id>.

```
GGSCI> ADD EXTRACT extwest, TRANLOG, BEGIN NOW, THREADS <instances>

GGSCI> ADD exttrail dirdat/ew, EXTRACT extwest, megabytes 50
```

Note: In a non-RAC environment, the THREADS parameter can be omitted or the **THREADS** <instances> can be set to **1**. On RAC, it is the number of nodes in the cluster.

Verify the results:

```
GGSCI> INFO EXTRACT extwest
```

3. Create the Extract Datapump parameter file

Execute the following command on the **<source>** system to create the Extract datapump parameter file.

```
GGSCI> EDIT PARAM pwest
```

Add the following commands to the parameter file.

```
-- Data Pump
EXTRACT pwest
RMTHOST <target>, MGRPORT <target_port>, COMPRESS
RMTTRAIL dirdat/pe
RecoveryOptions OverwriteMode
PASSTHRU
TABLE <schema>.TCUST*;
```

Note that the**<target>** is the Data Integration Server, not the JMS server.

4. Define the GoldenGate trail

Execute the following commands on the **<source>** to add the trail that will store the changes on the **<target>**.

```
GGSCI> ADD extract pwest, exttrailsource dirdat/ew
GGSCI> ADD rmttrail dirdat/pe, extract pwest, megabytes 50
```

Verify the results:

```
GGSCI> INFO ALL
```

5. Start the capture process

On the **<source>** system, issue the following commands:

```
GGSCI> START *
```

Verify the results:

```
GGSCI> INFO ALL
GGSCI> INFO ER *
```

6. Start the JMS Delivery process

On the **<target>** system, issue the following commands:

```
GGSCI> ADD extract javaue, extTrailSource dirdat/pe
GGSCI> INFO all
GGSCI> START *
```

Verify the results:

```
GGSCI> INFO ALL
GGSCI> INFO ER *
```

Discussion points

1. Configuration locations

Where is the Java application classpath set?

_____

_____

A: in the property file for the Java user exit, in the `bootoptions` property.

2. Sizing the GoldenGate trail

Where do you set how large a GoldenGate trail file may get before it rolls to the next file? What option do you use?

_____

_____

A: `ADD exttrail dirdat/ew, EXTRACT extwest, megabytes 50`

3. Extracts

How does the “extract” process running the Java adapter find the locally installed JVM?

_____

_____

A: On Windows, the `PATH` must include the directory containing `jvm.dll`. On UNIX, the library path (for example, `LD_LIBRARY_PATH`) must include the directory containing `libjvm.so` and any other required libraries to run the JVM. The “mgr” process must have this set, since it starts “extract”, which in turn loads the Java user-exit, which loads the JVM.

Exercise 3.
Generate Activity and Verify Results

Objective

The goals of this exercise are to:

- Execute miscellaneous update, insert, and delete operations on the source system.
- Verify the delivery of the changes to the target

Generate database operations

1. Execute miscellaneous update, insert, and delete operations

Execute the following commands on the **<source>** system.

```
Shell> cd <ogg_install>
Shell> sqlplus <login>/<password>
SQL> @demo_ora_misc
```

Note: SQL DDL operations (`CREATE, DROP, ALTER`) are not captured, nor are table `TRUNCATE` operations.

Verify change capture and delivery

2. Verify results on the source system

Execute the following commands on the **<source>** system.

```
SQL> select * from tcustmer;
SQL> select * from tcustord;
SQL> exit

Shell> ggsci
GGSCI> SEND EXTRACT extwest, REPORT
GGSCI> VIEW REPORT extwest
```

3. Verify your results on the target system

Execute the following commands on the **<target>** JMS system to verify the target data.

a. Open a browser to URL `http://<jmshost>:8161/admin/`

b. Click Queues > testQueue123 > Message ID where the Message ID is of the form: `ID:host_name-nnnnn-nnnnnnnnnnnn-n:n:n:n:n`, for example: `ID:EDRSR31P1-48978-1312804229935-0:1:1:1:4.`
You should see XML with a timestamp, table schema. name, column names and values you inserted, updated, or deleted. For example, the XML might look like this:

```
<operation table='WEST.TCUSTMER' type='UPDATE_FIELDCOMP'
ts='2011-08-08 11:56:51.000262' pos='00000000010000001511'
numCols='2'>
 <col name='BRANCH_NUMBER' index='0'>
   <before missing='true'/>
   <after><![CDATA[40]]></after>
 </col>
 <col name='BRANCH_ZIP' index='1'>
   <before missing='true'/>
   <after><![CDATA[87237]]></after>
 </col>
</operation>
```

c. Under Message Actions, you could Delete the message; or you could write a JMS consumer application that read from the Queue.

Execute the following commands on the Data Integration Server to verify the target data.

```
Shell> ggsci
GGSCI> VIEW REPORT javaue
```