Java Cloud Service Demo Guide: Application Continuity

Description

For Use By	Oracle Sales / Pre-Sales, Consulting, Product Managers, Solution Managers, etc
Internal/External Consumption	Internal

Target Audience	• Customer / Partner
	  ○ Architects,
	  ○ Development Leads

Demo Duration (Approximate)	Approximately 15 minutes.
Background Knowledge Required/Suggested	• Maven • WLS architecture • WLST • Git

Related Materials	• Oracle Public Cloud Services
	  ○ Java Cloud Services
	  ○ Database Cloud Services
	• Git or Downloaded Sources
	• Java Development Kit 1.8+
	• Apache Maven 3.2.5+

Feature Overview

Application Continuity is a feature that enables the replay, in a non-disruptive and rapid manner, of a request against the database after a recoverable error that makes the database session unavailable.

The request can contain transactional and non-transactional work. After a successful replay, the application can continue where that database session left off, instead of having users left in doubt not knowing what happened to their funds transfers, flight bookings, and so on, and avoiding the need to reboot mid-tier machines to recover from logon storms.

With Application Continuity, the end user experience is improved by masking many outages, planned and unplanned, without the application developer needing to attempt to recover the request.

Without Application Continuity, it can be almost impossible for an application to mask outages in a safe way.

Demo Overview

Two scenarios will be demonstrated:

In the first case, we would be using the old JDBC driver and thus when a transaction fails midway due to the abrupt killing of a database session – the transaction fails / remains incomplete and thus data is not written to the database tables.

In the second case, we would be using the new 12C JDBC drivers and thus when a transaction fails midway due to the abrupt killing of the database session, the transaction still goes through.

Requirements / Prerequisites

- Windows or Linux operating system
- Computer with 2 GB RAM
- Java Development Kit 1.8+
- Apache Maven 3.2.5
- Git
- A web browser is required for working with Oracle Cloud services. These are the minimum requirements:
  - Microsoft Internet Explorer 9 or 10 with Browser Mode and Document Mode set to IE9 or IE10
  - Mozilla Firefox 24 and later
  - Google Chrome 29 and later
  - Apple Safari 6 and later
- Roles and Credentials - You must have the following roles and credentials:
  - Roles and privileges described in Oracle Cloud User Roles and Privileges in Getting Started with Oracle Cloud
  - Log-in credentials for creating Oracle Java Cloud Service instances and for working with the associated database and storage services.
  - Java Administrator role (This role lets you create Oracle Java Cloud Service instances.)
  - For more information, see About Oracle Java Cloud Service Roles and User Accounts in Using Oracle Java Cloud Service.

Tips

- Use FireFox for the demos.

Demo Steps

Setup & Dependencies

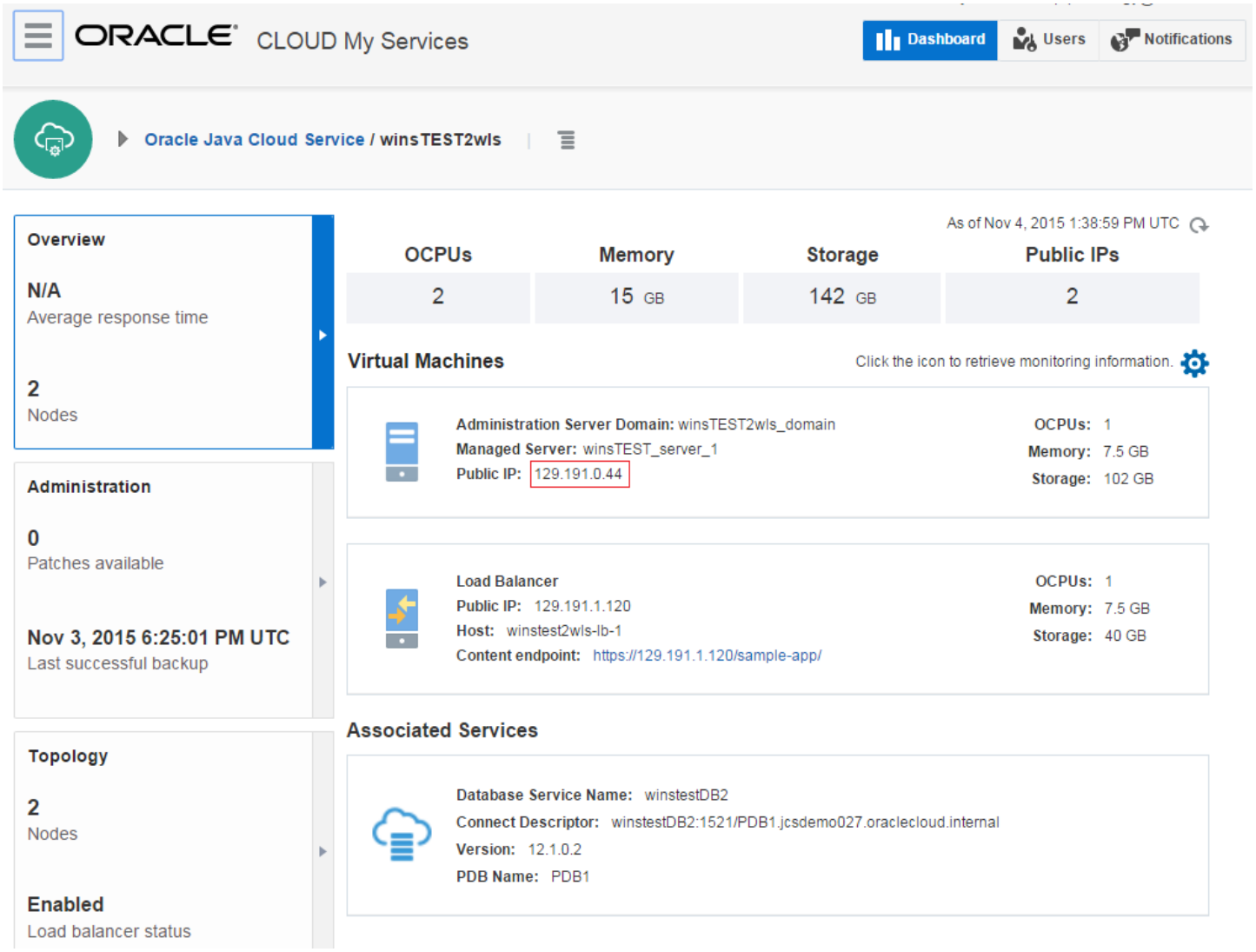
1. Install Java Development Kit and Apache Maven
2. Follow the steps outlined in 'WInS - Demo Guides - Setup – Clone Git repository' to get the necessary sources ready.
3. Create your SSH Keys for use with Oracle Cloud Services
4. Environment variables and directories
   a. This document will refer to the local clone of WInS remote repository as **WINS_SOURCE_REPOSITORY**. Every steps where it is used replace to your location.
   b. Set JAVA_HOME to point your jdk location. Example:
   Windows: `set JAVA_HOME=c:\jdk1.8.0_31`
   Linux: `export JAVA_HOME=/usr/java/jdk1.8.0_31`
   c. Add Apache Maven to your PATH variable. In this case it is easier to execute maven. Example:
   Windows: `set PATH=%PATH%;c:\apache-maven-3.2.5\bin`
   Linux: `export PATH=$PATH:/usr/apache-maven-3.2.5/bin`

Executing the Side by Side Deployment demo

The demo environment preparation does not require manual access, configuration to the Java and Database Cloud Service. All changes will be done by Maven installation. It is important to use clean JCS environment otherwise error can happen during the execution.

1) First get the IP address of your Java Cloud Service instance's Administration server and Database Cloud Server instance.
   To get JCS information run the following maven command in `WINS_SOURCE_REPOSITORY\cloud.demos` directory:
   `mvn install -DexecuteCloudUtil -Dgoal=jcs-get-instance-details`
   The following result will contain the necessary address:

Or open the Java Cloud Service Console. Click on the desired JCS instance and you can see the public IP address.



Now get the Database Cloud Service instance's address. One way again to get this information run the following maven command now for DBCS in `WINS_SOURCE_REPOSITORY\cloud.demos` directory:

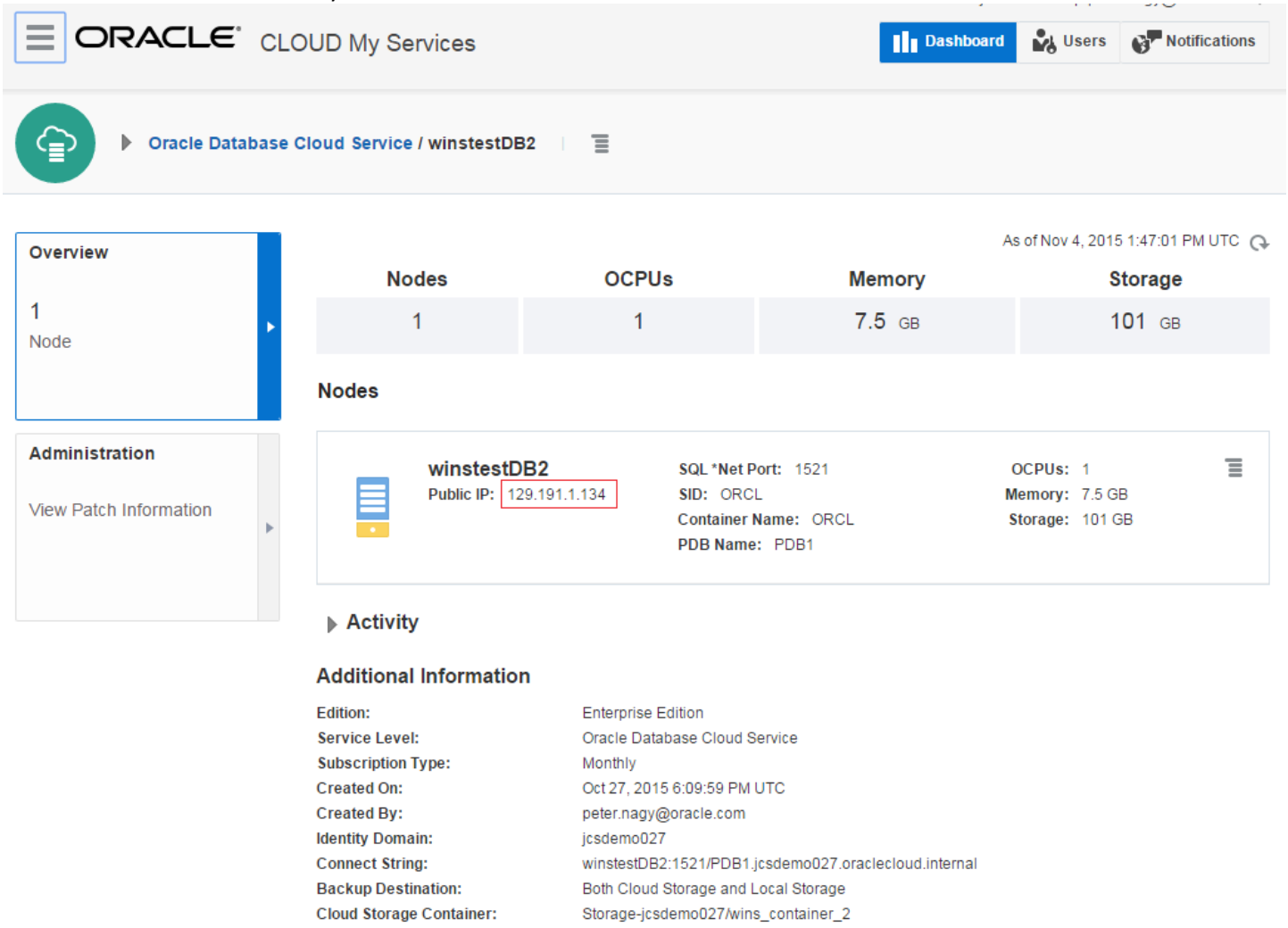
```
mvn install -DexecuteCloudUtil -Dgoal=dbcs-get-instance-details
```

The following result will contain the necessary address:

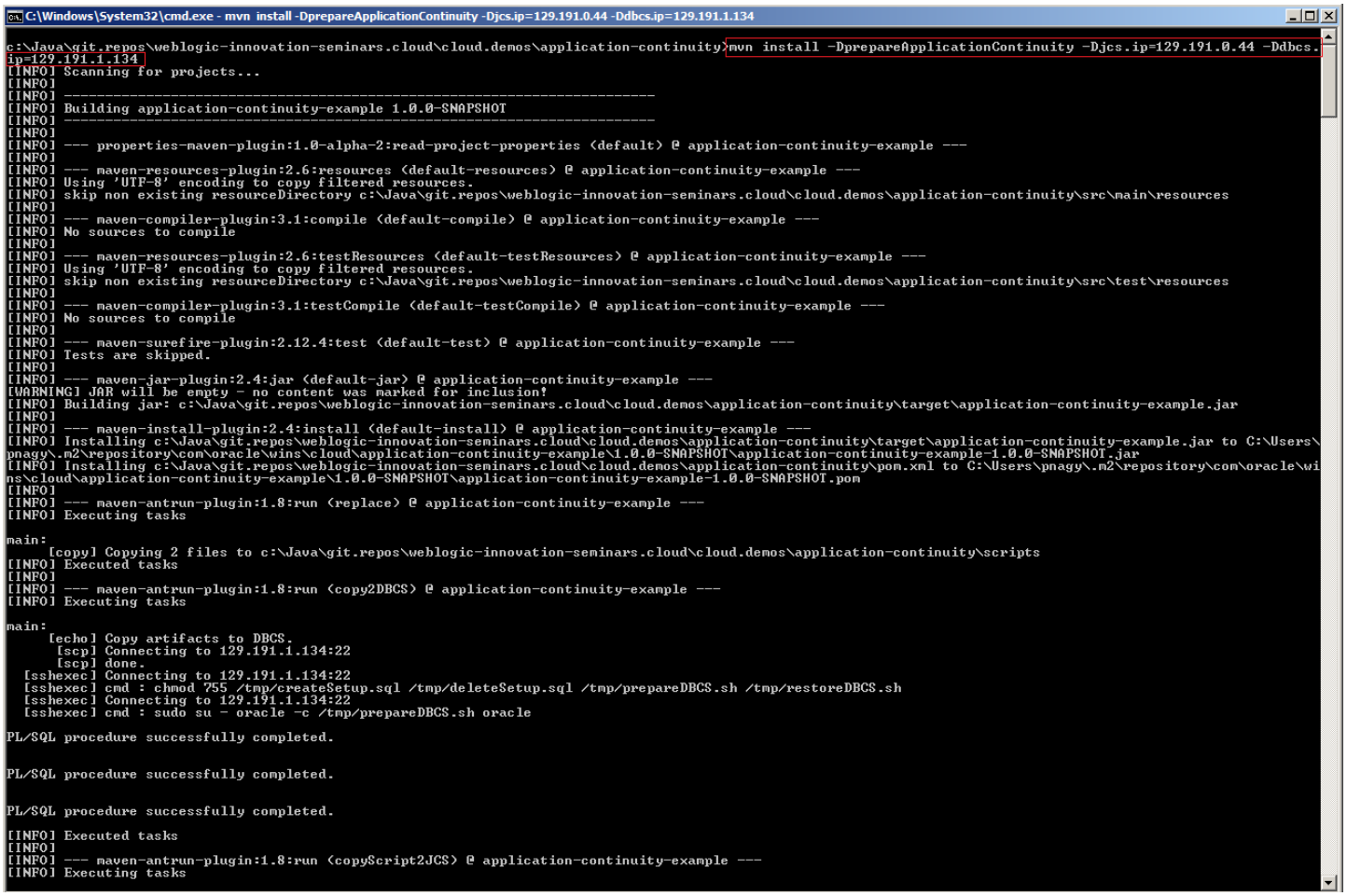
```
main:
     [java] Read all properties from file: environment.properties
     [java] Selected goal: dbcs-get-instance-details
     [java] DBCS get specific instance details----------------------------------------
     [java] Auth: {<any realm>@jaas.oraclecloud.com:443=[principal: peter.nagy@oracle.com]}
     [java] URI: http://jaas.oraclecloud.com/paas/service/dbcs/api/v1.1/instances/jcsdemo027/winstestDB2
     [java] Executing request GET http://jaas.oraclecloud.com/paas/service/dbcs/api/v1.1/instances/jcsdemo027/winstestDB2 HTTP/1.1
     [java] Response: HTTP/1.1 200 OK
     [java] Output from Server ....
     [java]
     [java] {
     [java]   "service_name": "winstestDB2",
     [java]   "version": "12.1.0.2",
     [java]   "status": "Running",
     [java]   "description": "winstest db instance through REST",
     [java]   "identity_domain": "jcsdemo027",
     [java]   "creation_time": "Tue Oct 27 18:9:59 UTC 2015",
     [java]   "last_modified_time": "Tue Oct 27 18:9:59 UTC 2015",
     [java]   "created_by": "peter.nagy@oracle.com",
     [java]   "service_uri": "https:\/\/jaas.oraclecloud.com:443\/paas\/service\/dbcs\/api\/v1.1\/instances\/jcsdemo027\/winstestDB2",
     [java]   "num_nodes": 1,
     [java]   "level": "PAAS",
     [java]   "edition": "EE",
     [java]   "shape": "oc3",
     [java]   "subscriptionType": "MONTHLY",
     [java]   "creation_job_id": "501843",
     [java]   "num_ip_reservations": 1,
     [java]   "backup_destination": "BOTH",
     [java]   "cloud_storage_container": "Storage-jcsdemo027\/wins_container_2",
     [java]   "failover_database": false,
     [java]   "rac_database": false,
     [java]   "sid": "ORCL",
     [java]   "pdbName": "PDB1",
     [java]   "listenerPort": 1521,
     [java]   "em_url": "https:\/\/129.191.1.134:5500\/em",
     [java]   "connect_descriptor": "winstestDB2:1521\/PDB1.jcsdemo027.oraclecloud.internal",
     [java]   "apex_url": "https:\/\/129.191.1.134\/apex\/pdb1\/",
     [java]   "glassfish_url": "https:\/\/129.191.1.134:4848",
     [java]   "dbaasmonitor_url": "https:\/\/129.191.1.134\/dbaas_monitor",
     [java]   "pdbss_url": "https:\/\/129.191.1.134\/apex\/f?p=PDBSS",
     [java]   "jaas_instances_using_service": "winsTEST2wls"
     [java] }
     [java]
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] wins-cloud ......................................... SUCCESS [  0.266 s]
[INFO] cloud-api .......................................... SUCCESS [  5.983 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.377 s
[INFO] Finished at: 2015-11-04T14:43:47+01:00
[INFO] Final Memory: 14M/322M
[INFO] ------------------------------------------------------------------------
```

Or this information is also available on the Database Cloud Service console as in case of Java Cloud Service. Click on

the desired DBCS instance and you can see the IP address.



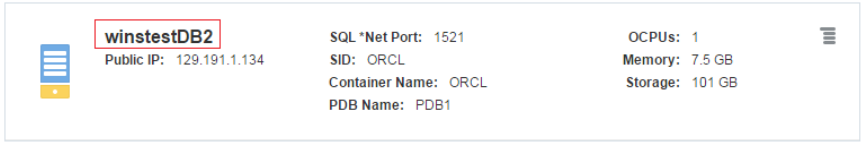
2) Once we have the JCS and DBCS instance's public IP addresses change directory to `WINS_SOURCE_REPOSITORY\cloud.demos\application-continuity`
Execute the maven build with the following options:
`mvn install -DprepareApplicationContinuity -Djcs.ip=129.191.0.44 -Ddbcs.ip=129.191.1.134`
Don't forget to change the IP addresses according to your environment.
The process will copy the scripts and application archive to the JCS and/or DBCS instance. Than execute the setup scripts. The result output should be the following:


C:\Windows\System32\cmd.exe - mvn install -DprepareApplicationContinuity -Djcs.ip=129.191.0.44 -Ddbcs.ip=129.191.1.134
c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity>mvn install -DprepareApplicationContinuity -Djcs.ip=129.191.0.44 -Ddbcs.ip=129.191.1.134
[INFO] Scanning for projects...
[INFO] Building application-continuity-example 1.0.0-SNAPSHOT
[INFO] --- properties-maven-plugin:1.0-alpha-2:read-project-properties (default) @ application-continuity-example ---
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ application-continuity-example ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity\src\main\resources
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ application-continuity-example ---
[INFO] No sources to compile
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ application-continuity-example ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity\src\test\resources
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ application-continuity-example ---
[INFO] No sources to compile
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ application-continuity-example ---
[INFO] Tests are skipped.
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ application-continuity-example ---
[WARNING] JAR will be empty - no content was marked for inclusion!
[INFO] Building jar: c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity\target\application-continuity-example.jar
[INFO] --- maven-install-plugin:2.4:install (default-install) @ application-continuity-example ---
[INFO] Installing c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity\target\application-continuity-example.jar to C:\Users\pnagy\.m2\repository\com\oracle\wins\cloud\application-continuity-example\1.0.0-SNAPSHOT\application-continuity-example-1.0.0-SNAPSHOT.jar
[INFO] Installing c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity\pom.xml to C:\Users\pnagy\.m2\repository\com\oracle\wins\cloud\application-continuity-example\1.0.0-SNAPSHOT\application-continuity-example-1.0.0-SNAPSHOT.pom
[INFO] --- maven-antrun-plugin:1.8:run (replace) @ application-continuity-example ---
[INFO] Executing tasks
main:
[copy] Copying 2 files to c:\Java\git.repos\weblogic-innovation-seminars.cloud\cloud.demos\application-continuity\scripts
[INFO] Executed tasks
[INFO] --- maven-antrun-plugin:1.8:run (copy2DBCS) @ application-continuity-example ---
[INFO] Executing tasks
main:
[echo] Copy artifacts to DBCS.
[scp] Connecting to 129.191.1.134:22
[scp] done.
[sshexec] Connecting to 129.191.1.134:22
[sshexec] cmd : chmod 755 /tmp/createSetup.sql /tmp/deleteSetup.sql /tmp/prepareDBCS.sh /tmp/restoreDBCS.sh
[sshexec] Connecting to 129.191.1.134:22
[sshexec] cmd : sudo su - oracle -c /tmp/prepareDBCS.sh oracle
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
[INFO] Executed tasks
[INFO] --- maven-antrun-plugin:1.8:run (copyScript2JCS) @ application-continuity-example ---
[INFO] Executing tasks

```
main:
     [echo] Copy artifacts to JCS.
      [scp] Connecting to 129.191.0.44:22
      [scp] done.
  [sshexec] Connecting to 129.191.0.44:22
  [sshexec] cmd : chmod 755 /tmp/createSetup.py /tmp/deleteSetup.py /tmp/prepareJCS.sh /tmp/restoreJCS.sh /tmp/otrade.ear
[INFO] Executed tasks
[INFO]
[INFO] --- maven-antrun-plugin:1.8:run (wlstprepare) @ application-continuity-example ---
[INFO] Executing tasks

prepare-WLS:
     [echo] Execute WLST to initialize WLS environment.
  [sshexec] Connecting to 129.191.0.44:22
  [sshexec] cmd : sudo su - oracle -c /tmp/prepareJCS.sh oracle

Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

Connect to nm...
/u01/data/domains/winsTEST2wls_domain
Calculated adminserver name: winsTEST_adminserver
Connecting to Node Manager ...
<Nov 4, 2015 1:52:23 PM UTC> <Info> <Security> <BEA-090905> <Disabling the CryptoJ JCE Provider self-integrity check for better startup performance. To enable t
his check, specify -Dweblogic.security.allowCryptoJDefaultJCEVerification=true.>
<Nov 4, 2015 1:52:23 PM UTC> <Info> <Security> <BEA-090906> <Changing the default Random Number Generator in RSA CryptoJ from ECDRBG128 to FIPS186PRNG. To disab
le this change, specify -Dweblogic.security.allowCryptoJDefaultPRNG=true.>
<Nov 4, 2015 1:52:23 PM UTC> <Info> <Security> <BEA-090909> <Using the configured custom SSL Hostname Verifier implementation: weblogic.security.utils.SSLWLSHos
tnameVerifier$NullHostnameVerifier.>
Successfully Connected to Node Manager.

RUNNING

winsTEST_adminserver status...RUNNING
Killing server winsTEST_adminserver ...
Successfully killed server winsTEST_adminserver
Starting server winsTEST_adminserver ...
Successfully started server winsTEST_adminserver ...
Successfully disconnected from Node Manager.
Create datasources...
Connecting to t3://winsTEST2wls-wls-1.compute-jcsdemo027.oraclecloud.internal:7001 with userid weblogic ...
Successfully connected to Admin Server "winsTEST_adminserver" that belongs to domain "winsTEST2wls_domain".

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

Location changed to edit tree. This is a writable tree with
DomainMBean as the root. To make changes you will need to start
an edit session via startEdit().

For more help, use help('edit')

Starting an edit session ...
Started edit session, please be sure to save and activate your
changes once you are done.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
Starting an edit session ...
Started edit session, please be sure to save and activate your
changes once you are done.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
```

```
Activation completed
Starting an edit session ...
Started edit session, please be sure to save and activate your
changes once you are done.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
Deploying application from /tmp/otrade.ear to targets winsTEST_adminserver (upload=false) ...
<Nov 4, 2015 1:53:31 PM UTC> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating deploy operation for application, otrade [archive: /tmp/otrade.ear], to winsT
EST_adminserver .>
.Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
Starting application otrade.
<Nov 4, 2015 1:53:35 PM UTC> <Info> <J2EE Deployment SPI> <BEA-260121> <Initiating start operation for application, otrade [archive: null], to winsTEST_adminser
ver .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State : completed
Deployment Message : no message
<Nov 4, 2015 1:53:38 PM UTC> <Warning> <JNDI> <BEA-050001> <WLContext.close() was called in a different thread than the one in which it was created.>
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:46 min
[INFO] Finished at: 2015-11-04T14:53:39+01:00
[INFO] Final Memory: 18M/322M
[INFO] ------------------------------------------------------------------------
```

3) Testing the environment for the Application Continuity demo:
   a. Go to the browser and type the URL: https://<AdminServer_VM_IP_Address> :7002/otrade/setup.jsp
   b. Enter the following:
      i. Listener Addresses: <DBCS_INSTANCE_NAME>:1521
      For example **winstestDB2** in the case below:

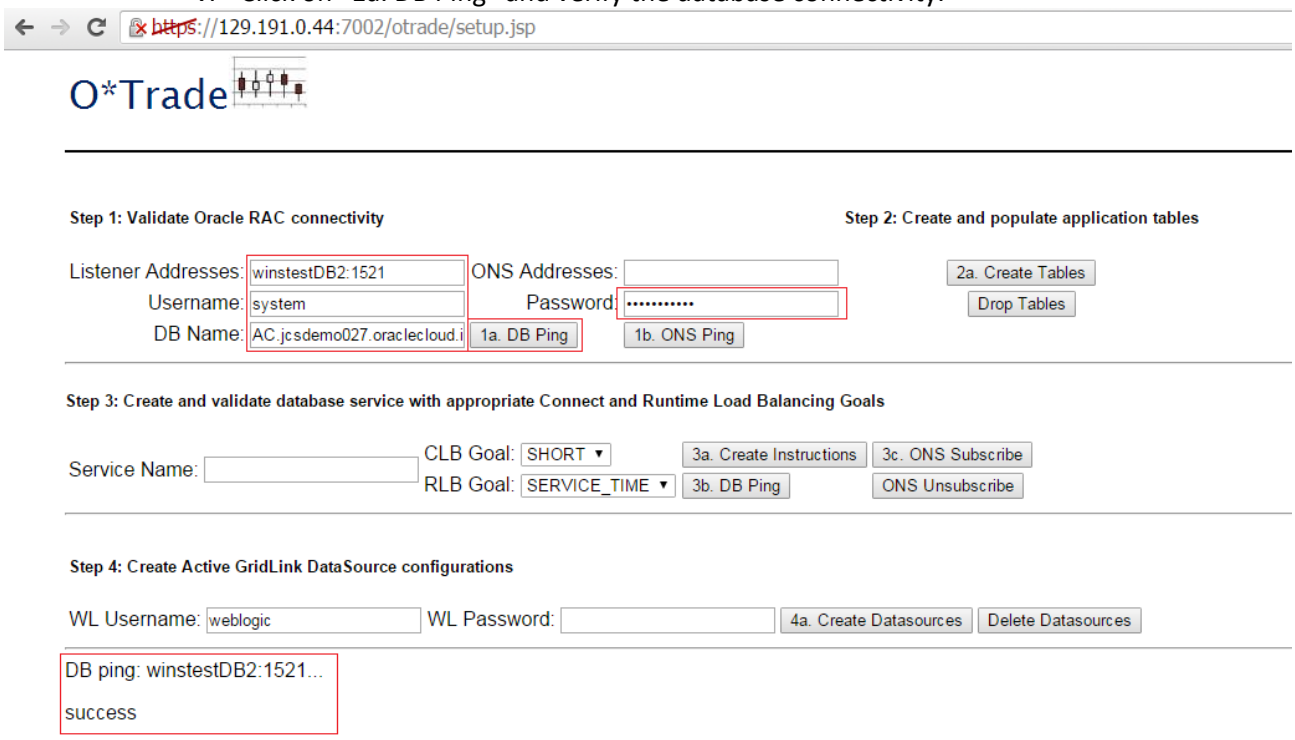


ii. Username: system
iii. Password: password specified during creation of DBCS and in environment.properties
Important! If the password or username contains non-alphanumeric characters then during the submit request can be corrupted. Please convert any special character to URLEncoded format. For example the **#** character encoded format is **%23**.
iv. DB Name: AC.<CLOUD_IDENTITY_DOMAIN>.oraclecloud.internal
You can check your full DBCS network postfix (including identity domain) on the console. For example using the information below we can get the following name:
AC.jcsdemo027.oraclecloud.internal

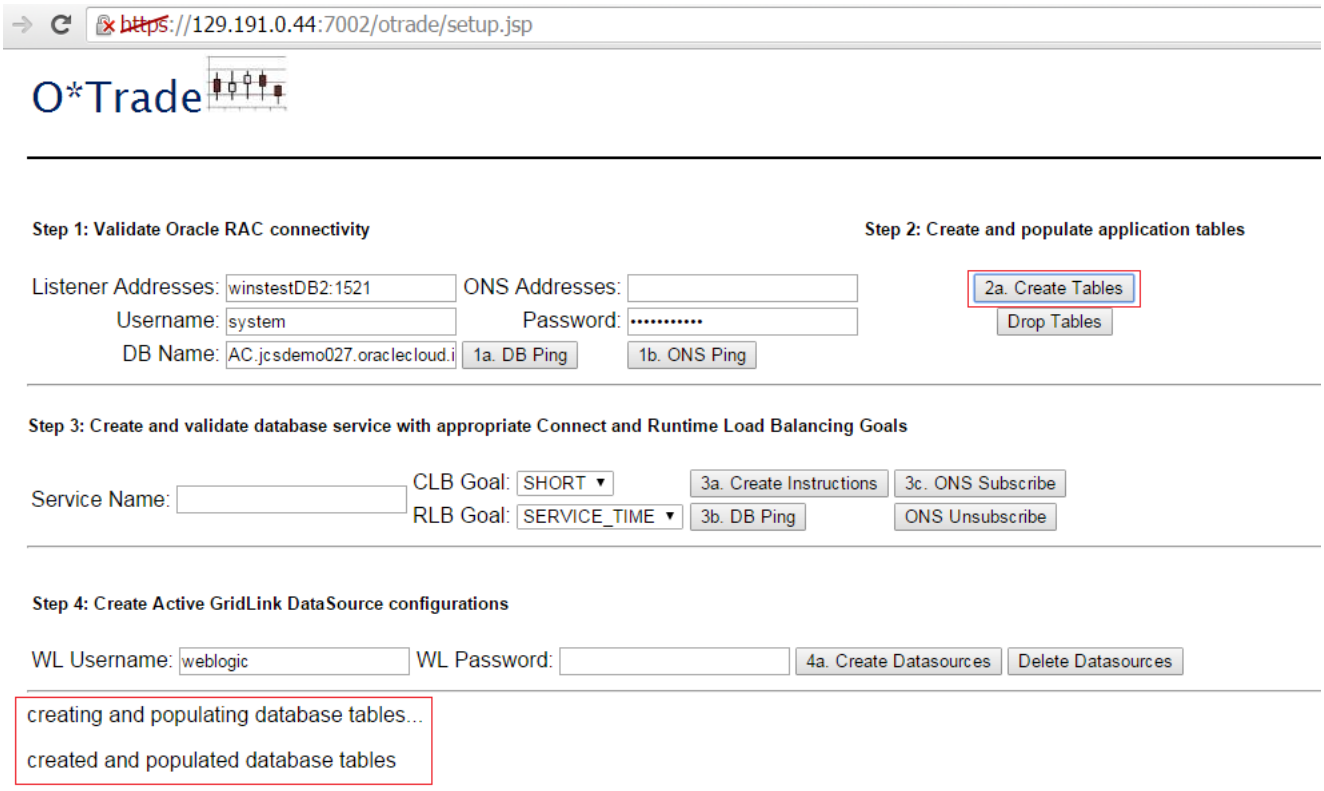
Additional Information

Edition:	Enterprise Edition
Service Level:	Oracle Database Cloud Service
Subscription Type:	Monthly
Created On:	Oct 27, 2015 6:09:59 PM UTC
Created By:	peter.nagy@oracle.com
Identity Domain:	jcsdemo027
Connect String:	winstestDB2:1521/PDB1 jcsdemo027.oraclecloud.internal
Backup Destination:	Both Cloud Storage and Local Storage
Cloud Storage Container:	Storage-jcsdemo027/wins_container_2

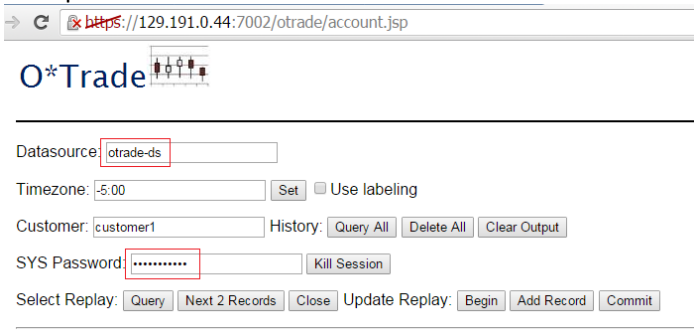
The AC service created by the setup script. The demo requires additional service because the application continuity settings are different comparing to the default (PDB1) service.
For more information check the setup sql script in the repository.
v. Click on "1a. DB Ping" and verify the database connectivity.



vi. Click on "*2a. Create Tables*" and ensure that corresponding success message is printed.

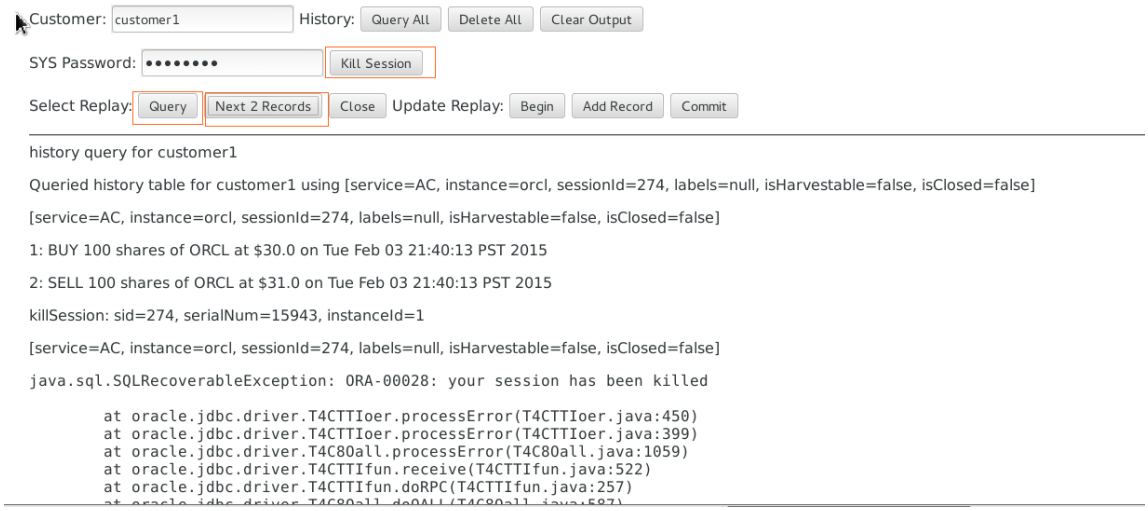


4) Testing the Application without Application Continuity (using normal data source)
   a. Open a new tab in the browser and type the following URL:
      https://<AdminServer_VM_IP_Address> :7002/otrade/account.jsp
   b. Enter the following:
      i. Datasource:otrade-ds
      ii. SYS Password: <password specified during creation of DBCS>
      Important! If the password or username contains non-alphanumeric characters then during the submit request can be corrupted. Please convert any special character to URLEncoded format. For example the **#** character encoded format is **%23**.



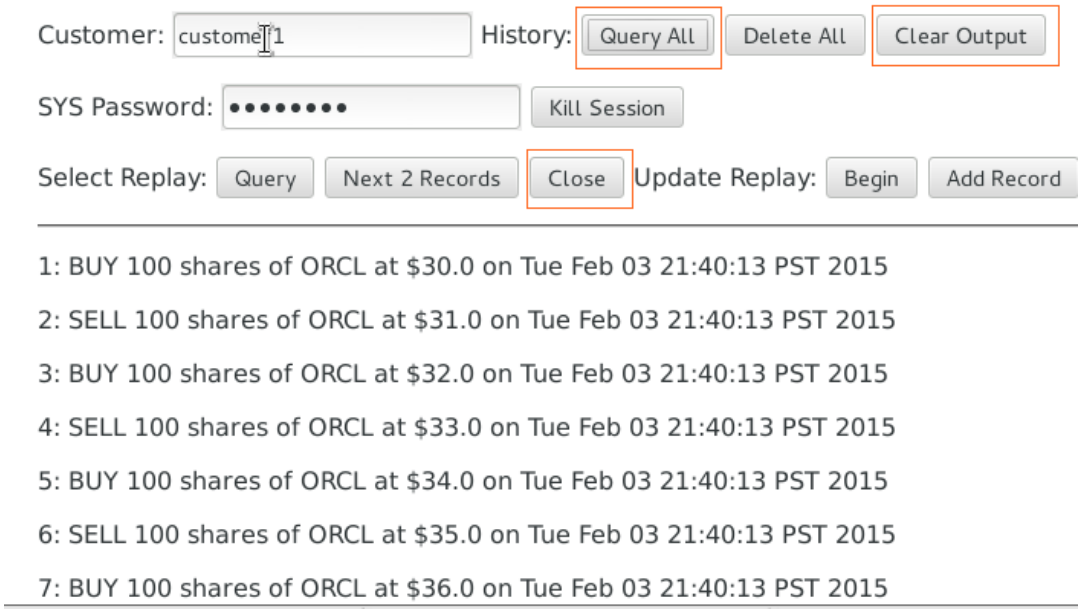
      iii. Click on "**Query->Next 2 Records->Kill Session->Next 2 Records**".
      At the first instant, two records will be printed on the screen. At the next step when the session is killed and we try to get the next two records – we get an exception which also indicates to the fact

that the session had been killed



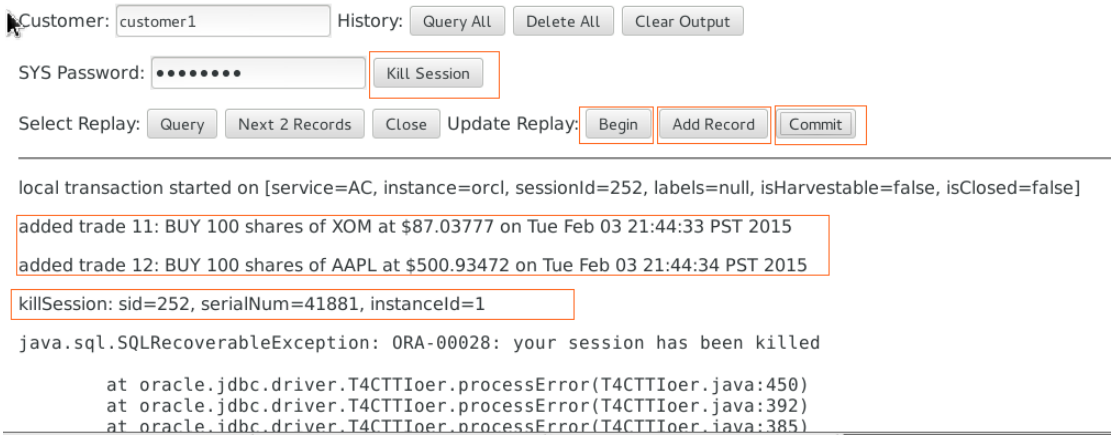
iv. Click on "**Close**->**Clear Output**->**Close** ->**Clear Output**".

v. Click on "**Query All**" to get all the data stored in the table. If you get an error, wait and then execute "**Query All**" again.

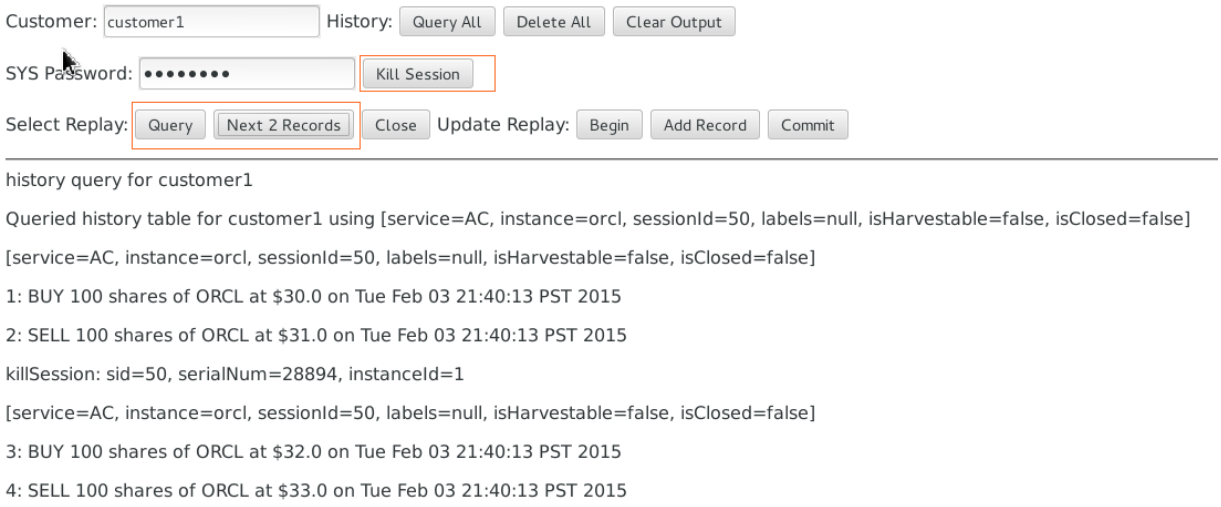


Note: There are 10 entries in the table currently.

vi. Click on "**Begin**" to start a local transaction. Then add two records by clicking on "**Add Record**" twice.
Now push the "Kill Session button to break the session.
Initially, its printed on the screen that two records have been added, but before committing, the database session was killed. So, the two records would not be stored / available in the database.

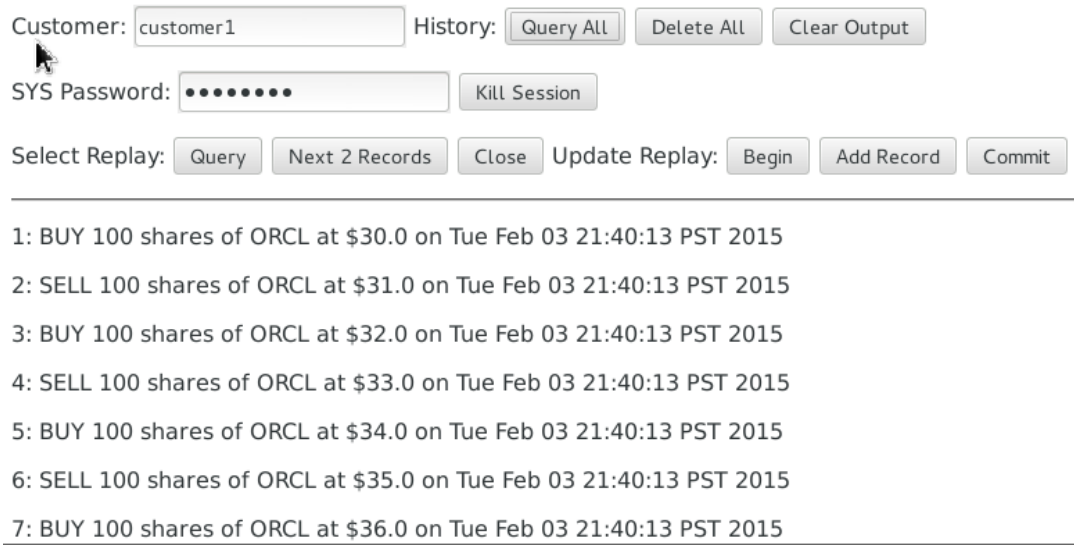


5) Testing the Application with new data source (supporting JDBC Replay)
   a. Open a new tab in the browser and type the following URL:
      https://<AdminServer_VM_IP_Address> :7002/otrade/account.jsp
   b. Enter the following:
      i. Datasource: otrade-replay-ds
      ii. SYS Password: <password specified during creation of DBCS>
   c. Click on "**Query**->**Next 2 Records**->**Kill Session**->**Next 2 Records**".
      At the first instant, two records will be printed on the screen. At the next step when the session is killed and we try to get the next two records – we do not get an exception in this case. Even though the session had been killed, still due to JDBC Replay and Application Continuity, we are able to retrieve the next two records with the help of the new 12C JDBC driver.

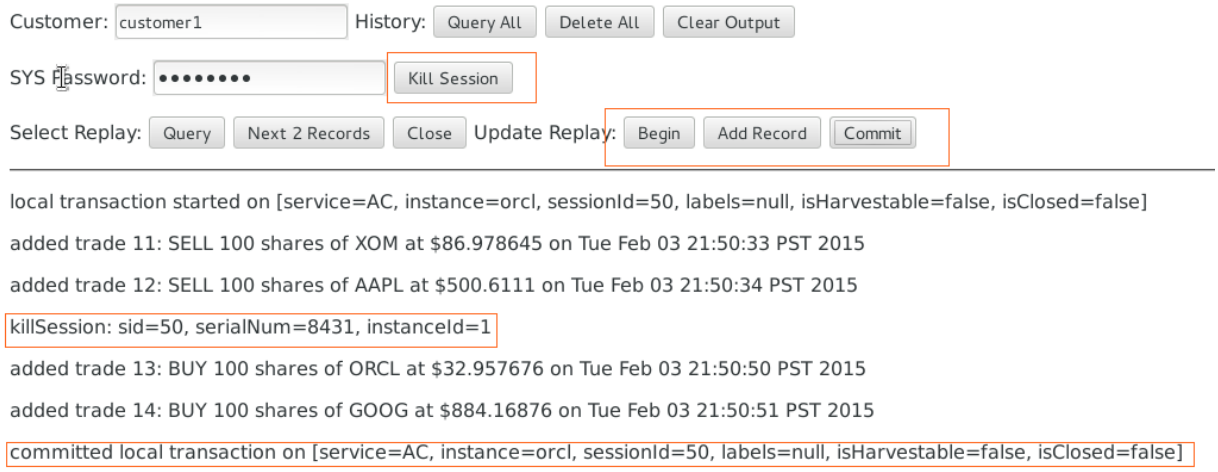


   d. Click on "**Close**->**Clear Output**->**Close** ->**Clear Output**".

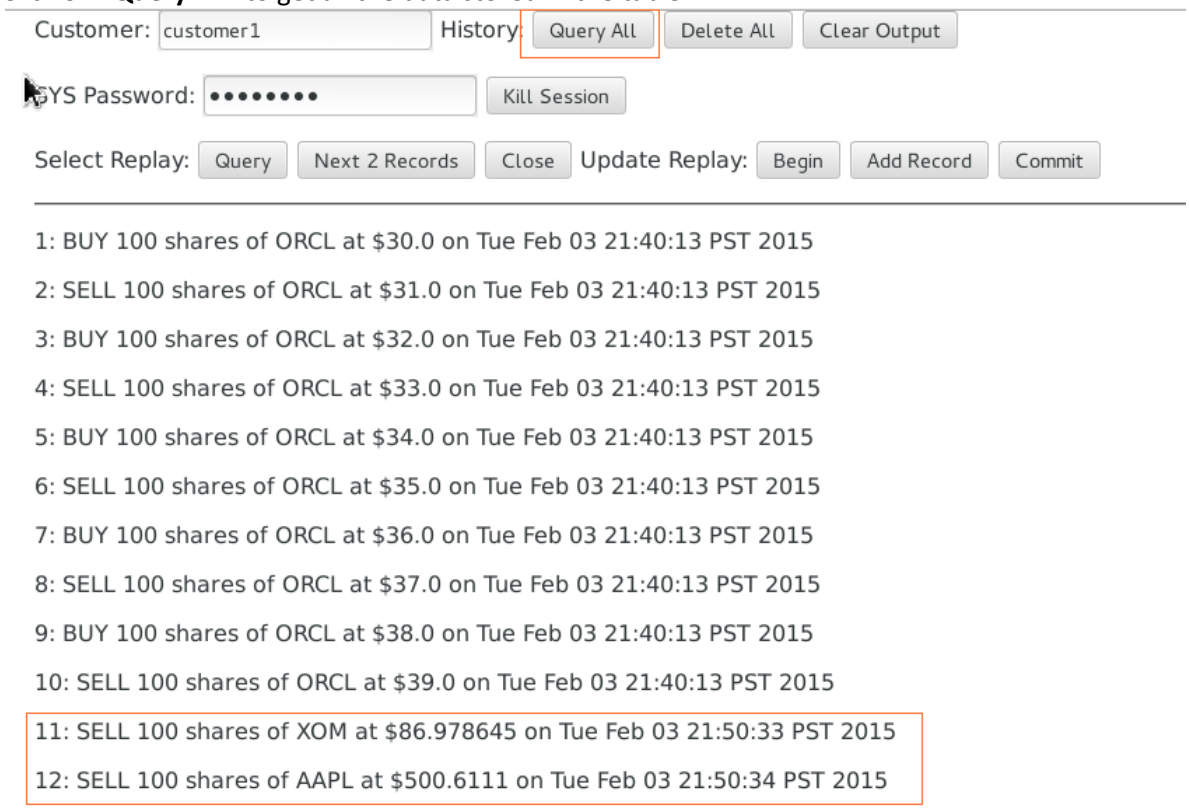
e. Click on "**Query All**" to get all the data stored in the table.



f. Click on "**Begin**" to start a local transaction. Then add two records by clicking on "**Add Record**" twice. Now kill the session by clicking on "Kill Session" and add two more records then commit the transaction.



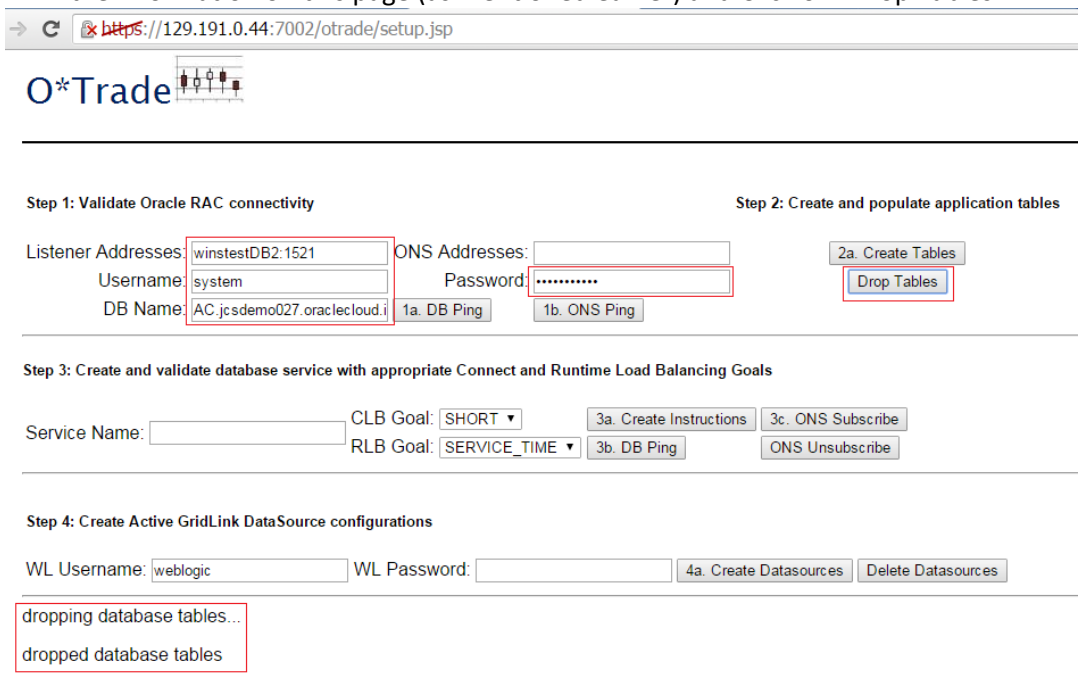
g. Click on "**Query All**" to get all the data stored in the table.



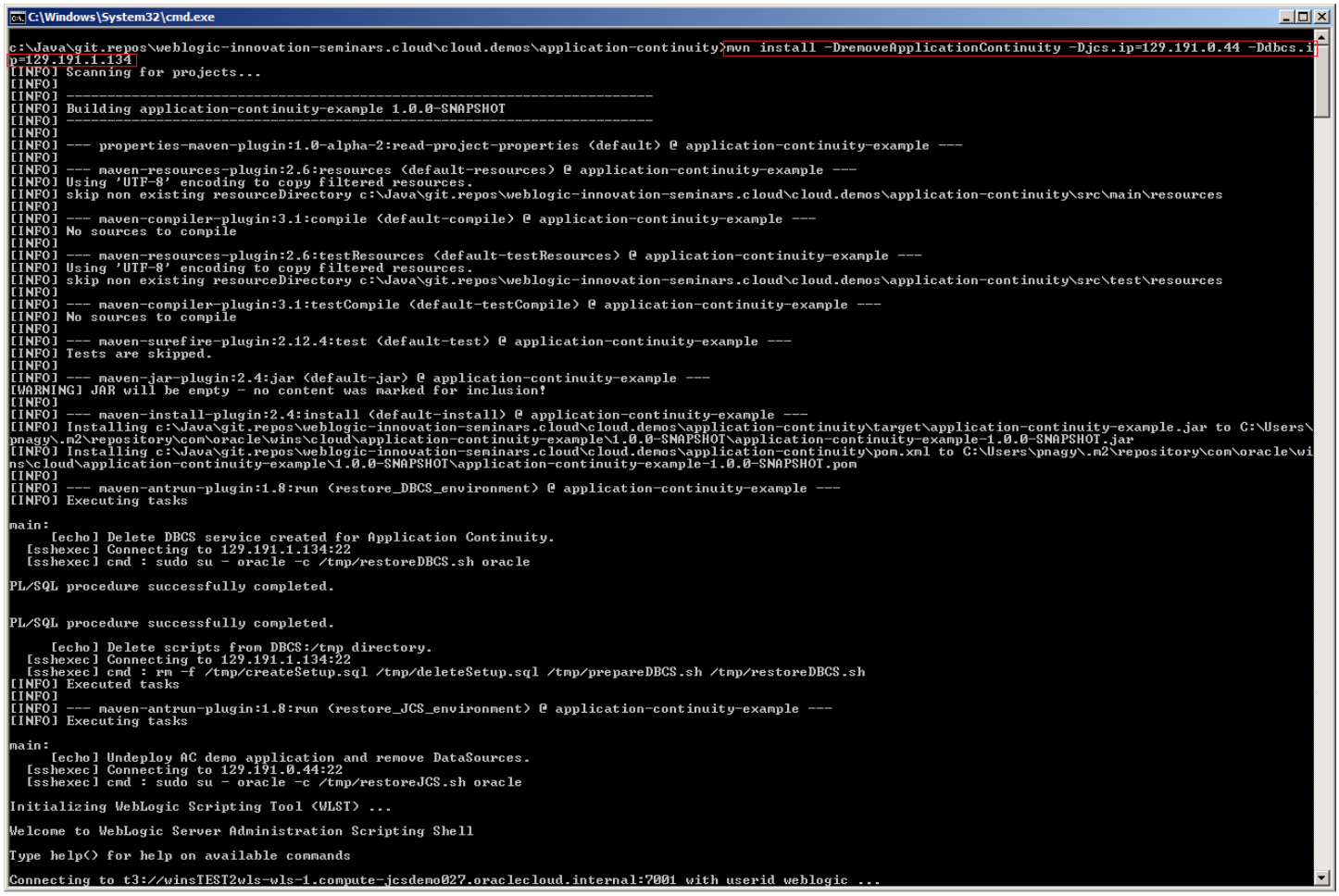
Note: In flight transaction are saved from killing the session. Earlier, when the previous JDBC Driver was being used this was not possible. Now using the new 12C JDBC Driver – even after killing the session, addition of new records in the same numerical order is possible and the data is committed and persisted to the database.

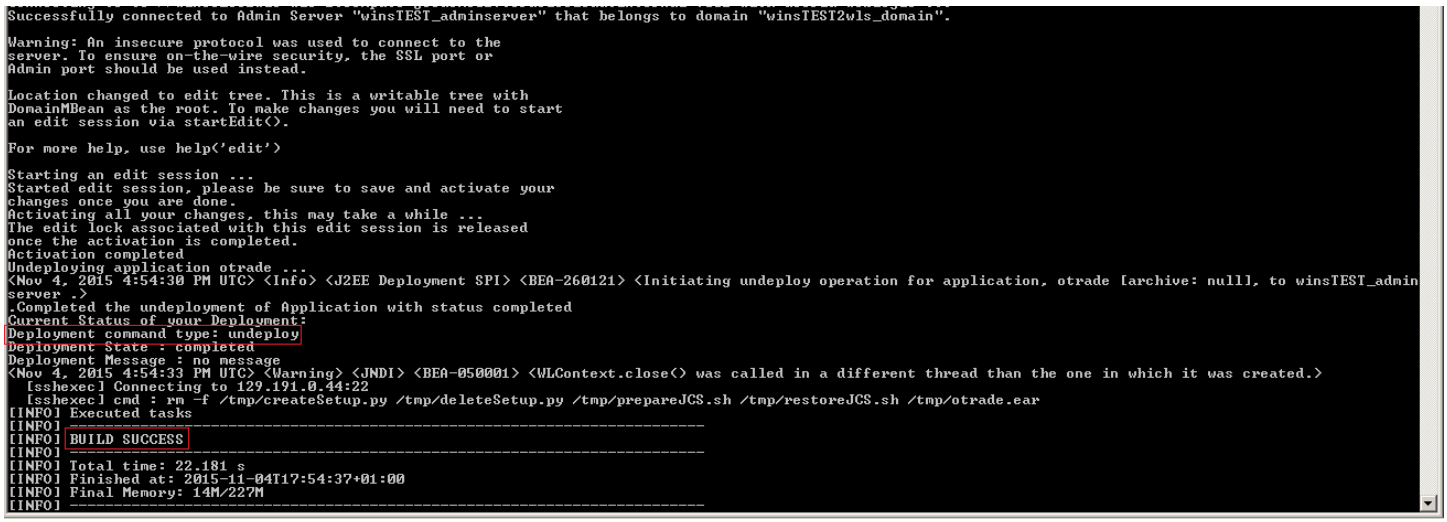
6) Clean up the environment.
   a. Delete the tables.
   b. Go back to the already opened URL: https://<AdminServer_VM_IP_Address> :7002/otrade/setup.jsp
      Or open a new browser window and type the URL above.

c. Fill in the information on this page (as mentioned earlier) and Click on “Drop Tables”.



7) To restore the Oracle Database and Java Cloud Service Instances execute the following maven build from the `WINS_SOURCE_REPOSITORY\cloud.demos\application-continuity` directory:
`mvn install -DremoveApplicationContinuity -Djcs.ip=129.191.0.44 -Ddbcs.ip=129.191.1.134`
Don’t forget to change the IP addresses according to your environment.
The build will undeploy the demo application, delete the datasources and remove the database service.
The result output should be the following:





When to use

- Introduction
    - In today's environment, application developers are required to deal explicitly with outages of the underlying software, hardware, communications, and storage layers. As a result, application development is complex

and outages are exposed to the end users. For example, some applications warn users not to hit the submit button twice. When the warning is not heeded, users may unintentionally purchase items twice or submit multiple payments for the same invoice.

- Application Continuity (also referred to as Replay) is a general purpose, application-independent infrastructure for GridLink and Generic data sources that enables the recovery of work from an application perspective and masks many system, communication, and hardware failures. The semantics assure that end-user transactions can be executed on time and at-most-once. The only time an end user should see an interruption in service is when the outage is such that there is no point in continuing.

- What happens in the absence of Application Continuity
  - Without Application Continuity, it can be almost impossible for an application to mask outages in a safe way, for reasons that include the following:
    - The state at the client remains at present time, with entered data, returned data, and variables cached.
    - If a commit has been issued, the commit message is not durable.
    - Checking on a lost request is no guarantee that it will not commit after checked.
    - Non-transactional database session state that the application needs to operate is lost.
    - If the request can continue, the database and the database session must be in the right state.
  - Application Continuity, however, does this work for the application developer, thus masking many outages in a safe way.

- How it works
  - Following any outage that is due to a loss of database service, planned or unplanned, Application Continuity rebuilds the database session. Once an outage is identified by Fast Application Notification or a recoverable ORACLE error, the Oracle driver:
    - Establishes a new database session to clear any residual state.
    - If a callback is registered, issues a callback allowing the application to re-establish initial state for that session.
    - Executes the saved history accumulated during the request.
  - The Oracle driver determines the timing of replay calls. Calls may be processed chronologically or using a lazy processing implementation depending on how the application changes the database state. The replay is controlled by the Oracle 12c Database Server. For a replay to be approved, each replayed call must return exactly the same client visible state that was seen and potentially used by the application during the original call execution

- How does Application Continuity help?

- Application Continuity improves developer productivity by attempting to mask outages that can be masked. Note, however, that applications still must include error handling for these cases:
  - Non-recoverable errors, such as invalid input data. (Application Continuity applies only to recoverable errors.)
  - Recoverable errors when replay has encountered one of the restrictions, such as use of concrete classes in the application, or when replay has not been able to restore the client-visible state to that on which the client may have made decisions so far.

Typical Questions

- How to write recovery code with Transaction Guard
  - Blogged here.
- Ensuring Application Continuity.
  - Documented here.
- More about DBMS_Service
  - Available here.
  - This should help in understanding the various nuances of the options available during creation of a 'Service'
- Oracle Whitepaper on Oracle WebLogic Server Integration with Oracle Database 12c
  - Available here.
- Oracle Whitepaper on Application Continuity with Oracle Database 12c
  - Available here.
- Oracle Whitepaper on Oracle Database 12c Application Continuity for Java
  - Available here.
- WebLogic Server Blog on 12C Database and WLS - Application Continuity
  - Available here.
- More about Application Continuity
  - Available here.
  - Helps in understanding the working of Application Continuity.
- Limitations with Application Continuity with Database Release 12.1.0.1

- Proxy authentication is not supported. That is, a transaction request will not be replayed and the original java.sql.SQLRecoverableException is thrown if an outage occurs.
- DRCP is not supported. That is, a web request will not be replayed and the original java.sql.SQLRecoverableException is thrown if an outage occurs.
- Cannot be used with PDB tenant switching using ALTER SESSION SET CONTAINER.

Wrap Up / Conclusion

Introduced in Oracle Database 12c Release 1 (12.1.0.1), Application Continuity strengthens the fault tolerance of systems and applications that use an Oracle database.

This new JDBC Driver is available only with Oracle WebLogic Server 12C and Oracle Database 12C.

The WInS cloud tool makes the provisioning, delete easier and faster. Instead of getting REST client tool and/or clicking through the console wizard the instance creation is much easier and less error proning.

Troubleshooting

To get more information it is useful to check the server output. It requires Secure Shell (ssh) connection to the JCS instance. After a successful login switch to oracle user:

```
sudo su – oracle
```

and print the tail of the output log:

```
tail -f
$DOMAIN_HOME/servers/<first_8_character_of_JCS_instance_name>_adminserver/logs/winsTEST_adminserver.out
```

In the log you can see the following entries which are created by the applications when database connection is necessary:

```
==========DB username: system
==========DB password: Welcome1#
==========DB service: AC.jcsdemo027.oraclecloud.internal
==========Connection url: jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=winstestDB2)(PORT=1521)))(CONNECT
_DATA=(SERVICE_NAME=AC.jcsdemo027.oraclecloud.internal)))
==========Server name: Oracle
==========Server version: Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Oracle Label Security option
==========Driver name: Oracle JDBC driver
==========Driver version: 12.1.0.2.0
==========Driver's jar: /u01/app/oracle/middleware/oracle_common/modules/oracle.jdbc_12.1.0/ojdbc7dms.jar
==========Connection url: jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=winstestDB2)(PORT=1521)))(CONNECT
_DATA=(SERVICE_NAME=AC.jcsdemo027.oraclecloud.internal)))
```

Here you can check whether the username and password was submitted successfully in case any of them contains non-alphanumeric characters.

If the preparation or remove script fails check the Weblogic instance through its administration console.