ORACLE
REAL APPLICATION CLUSTERS

STEP BY STEP GUIDE TO FAST CONNECTION FAILOVER WITH JDBC

1. Verify your RAC database environment is configured.

   a. RAC will publish FAN events via Oracle Notification Service (ONS). Check the Oracle Notification Service is running on all nodes in the cluster. Use onsctl ping command to see if the ONS daemon is active on all nodes

```
      >onsctl ping
      ONS is running
```

   b. If your application tier is running an ONS, then make the cluster aware of the application ONS (you can also have your application tier point to the Server).

```
racgons.bin add_config hostname:port [hostname:port]

>racgons.bin add_config appnode1:6200 appnode2:6200
```

   c. A service has been created with DBCA or SRVCTL and is enabled on at least one (preferably multiple nodes) in the cluster.

*Use crs_stat to check the Oracle Clusterware is managing the service.

```
                >crs_stat
                NAME=ora.barb.test.barb1.srv
                TYPE=application
                TARGET=ONLINE
                STATE=ONLINE on pmrac1

                NAME=ora.barb.test.barb2.srv
                TYPE=application
                TARGET=ONLINE
                STATE=ONLINE on pmrac2
```

   d. Server-Side Connection Load Balancing is configured for service

      i. The VIP is the hostname used for each node in the listener.ora (DBCA will have configured this for you)

```
# listener.ora.pmrac1 Network Configuration File:
/u01/app/oracle/product/10.2.0/asm/network/admin/listener.ora.pmrac1
# Generated by Oracle configuration tools.

SID_LIST_LISTENER_PMRAC1 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /u01/app/oracle/product/10.2.0/asm)
      (PROGRAM = extproc)
    )
  )

LISTENER_PMRAC1 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP)(HOST = pmvip1)(PORT = 1521)(IP = FIRST))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 144.25.214.45)(PORT = 1521)(IP = FIRST))
    )
  )
```

ii. All VIPs are resolvable from all nodes in the cluster and all client or middle tiers being used. Oracle recommends that the VIPs are defined in DNS however if this is not possible, then they must be defined in the /etc/hosts file for any client/mid-tier being used.

iii. LOCAL_LISTENER and REMOTE_LISTENER parameters are set in the initialization parameters. Your LOCAL_LISTENER parameter may be blank if you are using the default port 1521.

Check the listener is listening for the service using lsnrctl services command.

LSNRCTL SERVICES

```
[oracle@pmrac1 asm]$ lsnrctl services

LSNRCTL for Linux: Version 10.2.0.1.0 - Production on 11-
JAN-2006 08:31:54
Copyright (c) 1991, 2005, Oracle.  All rights reserved.
Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
Services Summary...
Service "TEST" has 2 instance(s).
  Instance "barb1", status READY, has 2 handler(s) for
this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
      "DEDICATED" established:0 refused:0 state:ready
         REMOTE SERVER
         (ADDRESS=(PROTOCOL=TCP)(HOST=pmrac1)(PORT=1521))
  Instance "barb2", status READY, has 1 handler(s) for
this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         REMOTE SERVER
         (ADDRESS=(PROTOCOL=TCP)(HOST=pmrac2)(PORT=1521))
```

2. Verify your application tier is configured.

   a. Your JDBC application must be using the Oracle JDBC driver to match the database version you are using (I.E. If database is 10.2, then client must use 10.2)

   b. Your JDBC datasource must be configured to use the Implicit Connection Cache, Fast Connection Failover (ONSConfiguration is 10.2 only)

```
ods.setUser("Scott");
ods.setPassword("tiger");
ods.setConnectionCachingEnabled(True);
   Properties prop = new Properties();
   prop.setProperty("MinLimit", "5");
   prop.setProperty("MaxLimit", "40");
   prop.setProperty("InitialLimit", "10");
   ods.setConnectionCacheProperties(prop);
ods.setFastConnectionFailoverEnabled(True);
ods.setConnectionCacheName("MyCache");
ods.setConnectionCacheProperties(cp);
ods.setONSConfiguration("nodes=racnode1:4200,racnode2.:4200");
ods.setURL("jdbc:oracle:thin:@(DESCRIPTION=
```

c. Either you must have ONS running on the application tier, started by the same user as your application OR with 10g Release 2, your datasource is configured to use a remote ONS
`(ods.setONSConfiguration("nodes=racnode1:4200,racnode2.:4200");)`

d. **Note:** JDBC has some ordering in parameters. For example, you must set up the connection cache with fast connection failover before naming the cache
`(ods.setConnectionCacheName("MyCache");).`

e. The ONS.JAR must be part of the classpath for the application. The ONS.JAR can be found in the Oracle Client installation. A Windows example:

```
set
JAVA_CLASSPATH=%JAVA_DIR%\lib\tools.jar;%JAVA_DIR%\src.jar;%ORA
CLE_HOME%\jdbc\lib;%ORACLE_HOME%\opmn\lib\ons.jar
```

Troubleshooting Your Environment

If you are using a 10.1 environment, the first thing to do is verify that the ONS on your mid-tier is receiving the alerts from the database.

1. Check the ons.config on the mid-tier and server is properly configured. In 10.1 it is not recommended to use the parameter useocr=on to store configuration information in the OCR. The userocr parameter is only valid on a RAC server. All client/mid-tiers and servers should be in the ons.config file.

```
localport=6100      # port ONS is writing to on this node
remoteport=6200     # port ONS is listening on this node
# This is the list of hosts and ports ONS is posting to.
# Include RAC and client nodes.
nodes=sun880-1.us.oracle.com:6200,sun880-2.us.oracle.com:6200,
sun880-3.us.oracle.com:6200,sun880-4.us.oracle.com:6200,
sun6800-1.us.oracle.com:6200
```

2. Use onsctl debug to verify the ONS is talking to the ONS on the server

   `Server connections` `show the servers and ports that this daemon is aware of. Initially this will correspond to the nodes entry in ons.config, but as other daemons are contacted any hosts they are in contact with will appear in this section.`
   Once you have caused an event at the server, check that the number of notifications received has increased

3. You can use the ons_subscriber java program included with this document using the runoh.bat (on Windows). This program connects to the ONS and prints any events received.

```
>runoh onc_subscriber
```

Once you have the ONS receiving the events from the RAC database, you need to enable a Java program with the correct datasource. The FCFDemo2 program provided with this document can be used as a simple program to test your configuration. Note: This program was written to work with 10g Release 2. You can run the program using the runoh.bat (on windows) with the parameter of the program name

```
>runoh FCFDemo2
```

Sample output from running the program is below. The program shows the url being used to connect as well as the Active and Available connections in the cache. In this example, the error occurs when Instance barb1 crashes, JDBC cleans up the cache, and the application reconnects to a surviving instance.

```
Url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(LOAD_BALANCE=ON)(
ADDRESS=(PROTOCOL=TCP)(HOST=pmrac1.us.oracle.com)(PORT=1521))(ADDRE
SS=(PROTOCOL=TCP)(HOST=pmrac2.us.oracle.com)(PORT=1521)))(CONNECT_D
ATA=(SERVICE_NAME=test)))
Instance name: barb1
FCF Activ(cache): 1
FCF Avail(cache): 9

Url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(LOAD_BALANCE=ON)(
ADDRESS=(PROTOCOL=TCP)(HOST=pmrac1.us.oracle.com)(PORT=1521))(ADDRE
SS=(PROTOCOL=TCP)(HOST=pmrac2.us.oracle.com)(PORT=1521)))(CONNECT_D
ATA=(SERVICE_NAME=test)))
java.sql.SQLException: Closed Connection at
oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java :111)
at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:145)
at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:207)
at oracle.jdbc.driver.OracleStatement.ensureOpen(OracleStatement.java:3512)
at oracle.jdbc.driver.OracleStatement.executeQuery(OracleStatement.java:1244)
at FCFDemo.main(FCFDemo2.java:61)
getErrorCode=17008
getSQLState=null
getMessage=Closed Connection
Url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(LOAD_BALANCE=ON)(
ADDRESS=(PROTOCOL=TCP)(HOST=pmrac1.us.oracle.com)(PORT=1521))(ADDRE
SS=(PROTOCOL=TCP)(HOST=pmrac2.us.oracle.com)(PORT=1521)))(CONNECT_D
ATA=(SERVICE_NAME=test)))
Instance name: barb2
FCF Activ(cache): 1
FCF Avail(cache): 4
```

If you are testing your JDBC environment and want to see more detail of what is actually happening, you can turn on logging from your JDBC Data Source. Here are the steps with a sample properties file. The JDBC demo.zip that is downloadable from Oracle Technology Network, includes a sample logging properties file -- OracleLog.properties.

1. Use JDK 1.4
2. Use debug jar ojdbc14_g.jar from $ORACLE_HOME/jdbc/lib. I.E. rename the ojdbc14.jar (something like ojdbc14.jar_save, and then rename ojdbc14_g.jar to ojdbc.jar)
3. Include a properties file, with contents (be careful of the word line wrapping):

```
handlers= java.util.logging.ConsoleHandler
# default file output is in user's home directory
java.util.logging.FileHandler.pattern = jdbc.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter=java.util.logging.XMLFormatter
# Setting this to SEVERE avoids duplicate output from
#  default logger
java.util.logging.ConsoleHandler.level = SEVERE
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter
oracle.jdbc.level = FINEST
oracle.jdbc.pool.level = FINEST
```

   Some output goes to StandardOut so you may want to redirect to a file.

4. Set the properties as shown below (a Windows example) when starting the test:

```
    %JAVA_DIR%\bin\java
    -Doracle.ons.oraclehome=C:\oracle\product\10.1.0\Db_1
    -classpath %CLASSPATH% -Doracle.jdbc.trace=true
    -Djava.util.logging.config.file= c:\test\jdbc\OracleLog.properties
    mytestprog  >>test.log
```

Your JDBC log should contain the following calls, do a simple search for the bold text below:

- Calls to **initFailoverParameters** - this shows you that Fast Connection failover is enabled for your datasource.

```
Oct 13, 2005 9:01:10 PM oracle.jdbc.pool.OraclePooledConnection
registerImplicitCacheConnectionEventListener
FINE:
OraclePooledConnection.registerImplicitCacheConnectionEventListener(oracle.jdbc.pool.OracleCon
nectionCacheEventListener@cfec48)
Oct 13, 2005 9:01:10 PM oracle.jdbc.pool.OracleImplicitConnectionCache initFailoverParameters
FINE: OracleImplicitConnectionCache.initFailoverParameters()
Oct 13, 2005 9:01:10 PM oracle.jdbc.pool.OracleImplicitConnectionCache initFailoverParameters
FINER: Instance Name =
Oct 13, 2005 9:01:10 PM oracle.jdbc.pool.OracleImplicitConnectionCache initFailoverParameters
FINER: Host Name =
Oct 13, 2005 9:01:10 PM oracle.jdbc.pool.OracleImplicitConnectionCache initFailoverParameters
FINER: Service Name =
Oct 13, 2005 9:01:10 PM oracle.jdbc.pool.OracleImplicitConnectionCache initFailoverParameters
FINER: DBUniq Name =
```

- FCF **eventType** and **eventBody** (showing the full event) – this shows you JDBC got the event when a failure occurred.
- Calls to **abortConnection** – this shows you that JDBC is cleaning up the connections to the failed instance when it receives an event (instance failure, service down, shutdown etc.). If you do not see this, FCF is not working.

```
Oct 13, 2005 9:02:36 PM oracle.jdbc.pool.OracleImplicitConnectionCache abortConnection
```