Oracle Application Server 10g Fast Connection Failover Configuration Guide

1. - Introduction ............................................................................................................. 3
2. - Configuring Fast Connection Failover in OracleAS 10g........................................... 4
   2.1.- Types of JDBC Connection URLs supported for Fast Connection Failover.......................................................... 4
   2.2.- Enabling Implicit connection Cache and Fast Connection Failover in OracleAS 10g Middle Tiers ........................................... 5
      2.2.1- Enabling Implicit Connection Cache and Fast Connection Failover in OracleAS 10g R2 Datasources ........................................... 5
      2.2.2.- Enabling Implicit Connection Cache and Fast Connection Failover in OracleAS 10g R3 Datasources ........................................... 6
   2.3.- Configuring Oracle Notification Server to propagate events from the RAC nodes to the Middle Tiers ........................................... 8
3. - Tests and verifications for Fast Connection Failover in OracleAS environments ........................................................................................................... 11
   3.1.- Verifying ONS communication between RAC nodes and middle tiers .................................................................................. 12
   3.2.- Verifying that the JDBC drivers are processing ONS events and performing failover correctly ........................................... 15
Summary.- .................................................................................................................. 17
APPENDIX.- ........................................................................................................... 18
1. - INTRODUCTION

Oracle Application Server’s High Availability solutions can be used in concert with the Oracle Database’s high availability features, thus offering end-to-end High Availability that no other vendor can match.

Fast Connection Failover is a clear demonstration of the high degree of integration between the technologies in the middle tier and the database tier in an Oracle environment. The failure notification mechanism of these two layers, based on Oracle Notification Server (ONS) have been integrated, allowing the database quickly notify the middle tiers whenever a failure happens in a DB instance.

From the implementation point of view, Fast Connection Failover (FCF) is a Real Application Cluster (RAC) 10g client provided by the JDBC Implicit Connection Cache (ICC), a feature of Oracle 10g JDBC drivers. Its primary purpose is to guarantee the validity and availability of a connection in the cache. It provides the main following features:

- Rapid Dead Connection Detection (DCD) of connections in the Implicit Connection Cache
- Removal of such stale or bad connections from the cache
- Propagation of errors to the caller to facilitate retries at higher layers
- Connection redistribution when a new a database instance joins a Real Application Cluster

![Diagram of Fast Connection Failover](image)
Starting with Oracle Application Server 10g R3, the Implicit Connection Cache, and hence, Fast Connection Failover, are fully integrated into Oracle Containers for J2EE (OC4J) datasources, thus making the configuration of this feature very simple and also enabling the full capacity of datasource related aspects (like container managed persistence for EJBs) available when used in conjunction with FCF. Previous releases of Oracle Application Server can leverage as well the power of FCF for maximum availability but with a few specific configuration changes. This document is intended to

1.- Provide a configuration guide for enabling Fast Connection Failover in the different Oracle Application Server releases

2.- Provide a set of tests and troubleshooting tips to verify that Fast Connection Failover between an OracleAS Middle Tier and an Oracle Database RAC is working properly.

It is not the intention of this white paper to provide a detailed technical description of the different features and technologies that Fast Connection Failover relies on. Additional documentation and details are provided in the documents referenced in the Appendix.

2.- CONFIGURING FAST CONNECTION FAILOVER IN ORACLEAS 10G

In summary, the steps required to configure Fast Connection Failover in the different Oracle Application Server 10g releases are the same:

- The jdbc connection url must use service names to connect to the database; it cannot use service IDs.
- Implicit Connection Cache and FCF are enabled in the corresponding Oracle Containers for J2EE datasource
- Oracle Notification Server (ONS) is configured to propagate events between the database (10g R1 or 10g R2 Real Application Clusters (RAC)) and the OracleAS Middle Tier.

The following sections describe each one of the steps above in more detail for the different OracleAS 10g Releases. It is strongly recommended to verify the latest patch availability on both tiers (database and middle tier) through Metalink (Metalink.oracle.com) to overcome possible environment-specific issues.

2.1.- Types of JDBC Connection URLs supported for Fast Connection Failover

It is only allowed to use service names identifiers in the URLs that are used in the data source definition of OracleAS when FCF is going to be enabled. The following examples are valid URLs to be used in OracleAS

```
url="jdbc:oracle:oci:@TNS_ALIAS"

or

url="jdbc:oracle:oci:@(DESCRIPTION=(LOAD_BALANCE=on))
```

```
or

url = "jdbc:oracle:thin:@//host:port/service_name"
url = "jdbc:oracle:thin:@//cluster-alias:port/service_name"

or

url = "jdbc:oracle:thin:@(DESCRIPTION=
(LOAD_BALANCE=on)
(ADDRESS=(PROTOCOL=TCP)(HOST=host1) (PORT=1521))
(ADDRESS=(PROTOCOL=TCP)(HOST=host2)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=service_name)))"

Notice that only service_name-based URLs are allowed, i.e. the traditional url based on a SID for the database, are invalid.

Invalid Usage: url = "jdbc:oracle:thin@host:port:SID"

2.2.- Enabling Implicit connection Cache and Fast Connection Failover in OracleAS 10g Middle Tiers

The degree of integration of the Implicit Connection Cache into OracleAS datasources has evolved with the different OracleAS Releases. The first OracleAS 10g releases (10g R1 and 10g R2) allowed a direct replacement of its data source with the Implicit Connection Cache’s. In the latest releases (10g R3 and 10g SOA), OracleAS provides a full integration of the Implicit Connection Cache into OracleAS managed datasources. The following subsections describe how to enable ICC in the OracleAS datasource configuration for the different 10g releases.

2.2.1- Enabling Implicit Connection Cache and Fast Connection Failover in OracleAS 10g R2 Datasources

In OracleAS 10g R2 Implicit Connection Cache support is limited to native data sources. Emulated and non-emulated data sources do not support this feature. (This means that you can’t use, for example, Container Managed Persistence EJBs with FCF in this release). It needs to be noted that the default JDBC drivers included in OracleAS 10g R2 and R3 are JDBC 10.1.0.5. Depending on the database release being used in the topology being configured, it may be required to upgrade the JDBC drivers. Upgrade and certification of the JDBC drivers included with OracleAS is limited to specific versions of these JDBC drivers. Please check Metalink for available/certified drivers and patches that may be required. The

---

1 See more details on the different datasources here  
2 10.1.0.4.2/10.1.0.5
upgrade or change of the JDBC thin drivers can be performed as described here. For testing and debugging purposes, direct overriding of the existing drivers with the updated versions will work. However, this direct replacement should in general be avoided due to possible classloading issues.

To enable ICC and FCF in an OracleAS instance, it is required to modify the datasource configuration (data-sources.xml file under OracleASInstanceHOME/j2ee/oc4j_home/config) with the following:

- It should specify “oracle.jdbc.pool.OracleDataSource as the datasource class
- It should specify the “connectionCacheName” and “connectionCachingEnabled=true” properties
- It should specify the “fastConnectionFailoverEnabled=true” property

The following sample data-sources.xml reflects in red these changes:

```xml
<data-source
  class="oracle.jdbc.pool.OracleDataSource"
  name="OracleDS"
  location="jdbc/OracleCache"
  connection-driver="oracle.jdbc.driver.OracleDriver"
  username="scott"
  password="tiger"
  url="jdbc:oracle:thin:@(DESCRIPTION=
    (LOAD_BALANCE=on)
    (ADDRESS=(PROTOCOL=TCP)(HOST=host1) (PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=host2)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=service_name))))">

  <property name="connectionCacheName" value="ICC"/>
  <property name="connectionCachingEnabled" value="true"/>
  <property name="fastConnectionFailoverEnabled" value="true"/>
</data-source>
```

2.2.2.- Enabling Implicit Connection Cache and Fast Connection Failover in OracleAS 10g R3 Datasources

In OracleAS 10g R3, Implicit Connection Cache is supported both with managed datasources and native data sources [2]. If required, the jdbc drivers included with OracleAS 10g R3, can be upgraded as well. Always check Metalink for available/certified drivers and patches. The procedure required to perform this upgrade is different than the one used to upgrade the OracleAS 10g R2 jdbc drivers. This procedure is described here.

To enable ICC in an OracleAS 10g R3 Managed Datasource, it is required to:

- Use the "connection-pool-name" attribute in each XML element in the same file to specify the name of the connection cache that this managed data source uses to pool its connections.
- Specify the “connectionCacheName” and “connectionCachingEnabled=true” properties
- Specify the “fastConnectionFailoverEnabled=true” property

The following sample data-sources.xml reflects in red the changes for enabling ICC and FCF in an OracleAS 10gR3 managed datasource:

```xml
<managed-data-source jndi-name="jdbc/ManagedDS" description="Managed DataSource" connection-pool-name="myConnectionPool" name="ManagedDS"/>
<connection-pool name="myConnectionPool" min-connections="10" max-connections="30" inactivity-timeout="30">
  <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource" user="scott" password="tiger" url="jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)
  (ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=host2)(PORT=1521))
  (CONNECT_DATA=(SERVICE_NAME=service_name)))">
    <property name="loginTimeout" value="30"/>
    <property name="connectionCacheName" value="ICC"/>
    <property name="connectionCachingEnabled" value="true"/>
    <property name="fastConnectionFailoverEnabled" value="true"/>
  </connection-factory>
</connection-pool>
```

In the case of a Native Datasources the changes are the same as in OracleAS 10g R2:

- It is required to use “oracle.jdbc.pool.OracleDataSource” as the datasource class
- It is needed to specify the “connectionCacheName” and “connectionCachingEnabled” properties
- Specify the “fastConnectionFailoverEnabled=true” property

The following sample data-sources.xml reflects in red the changes required for enabling ICC and FCF in an OracleAS 10gR3 native datasource:

```xml
<native-data-source name="nativeDataSource" jndi-name="jdbc/nativeDS" description="Native DataSource" data-source-class="oracle.jdbc.pool.OracleDataSource" user="scott" password="tiger" url="jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)
  (ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=host2)(PORT=1521))
  (CONNECT_DATA=(SERVICE_NAME=service_name)))">
  <property name="connectionCacheName" value="ICC"/>
  <property name="connectionCachingEnabled" value="true"/>
  <property name="fastConnectionFailoverEnabled" value="true"/>
</native-data-source>
```
2.3.- Configuring Oracle Notification Server to propagate events from the RAC nodes to the Middle Tiers

This is probably the most critical part of a correct Fast Connection Failover configuration. The detection and clean up of JDBC connections by the middle tier depends completely on the correct propagation of database events from the RAC nodes to the middle tiers. As explained in the introduction, these notifications are issued by Oracle Notification Server (ONS). There are different ONS versions that are used with the different Oracle Application Server releases. The following summarizes the different Oracle Notification Server releases that are bundled with the different products:

- Oracle Database (either 10gR1 or 10g R2) include ONS 10.1.x and 10.2.x respectively.
- OracleAS 10g R2 uses ONS 10.1.2
- OracleAS 10gR3 uses ONS 10.1.3

The configuration of each ONS may present slight differences for each product/release combination. This are the files where the configuration resides for each product release:

- ONS definitions in the RAC nodes (either 10g R1 or 10g R2) will be under OHOME/opmn/conf/ons.config
- ONS configuration in OracleAS 10g R2 is specified in OHOME/opmn/ons.conf
- ONS configuration in OracleAS 10g R3 resides in OHOME/opmn/conf/opmn.xml

As far as Fast Connection Failover is concerned, these files will contain the ONS address and port for ONS to send and receive notifications. The port that ONS uses for Fast Connection Failover events is the one tagged as “remoteport”. ONS communication can be secured through SSL by specifying the correct wallet configuration in both the middle tiers and the RAC nodes ONS configuration. However, in general, Db nodes will typically run in secured tiers and it may be better to disable SSL for ONS ([4],[5]). When this is done, the disabling needs to take place in the RAC nodes and in all middle tiers that are directly communicated with them.

Independently of where the final parameters are stored/persisted, the ONS configuration can be done in different ways also depending on the product release. In general the Db clients should be configured with the address of the Db nodes and not the other way around (i.e., Db nodes do not need to be configured with the ONS listeners’ addresses of all the middle tiers). However there is an exception to this case with OracleAS 10g R2 middle tiers. This is described bellow.

2.3.1.- FCF ONS configuration in 10g R1 Db

---

3 Notice that the ONS included with OracleAS 10g R3 is not compatible with the ONS included with 10g R1 RAC database. Updating the RAC DBs ONS to ONS 10.2 is required in this case. Check Metalink for latest patches and procedures to do this
In Oracle 10g R1 DB, it is required to edit the ons.config file under OracleDB_HOME/opmn/conf. Example:

```
localport=6100
remoteport=6202
nodes=rac_node1.us.oracle.com:6200, rac_node2.us.oracle.com:6200
```

In order for these changes to be effective you will need to restart ONS with the `onsctl reconfig` command. This has to be done in all instances in the RAC cluster and all DB instances should all nodes share the same Ons configuration to elude ONS timeouts and connectivity issues.

### 2.3.2.- FCF ONS configuration in 10g R2 Db

For Oracle 10g R2 Dbs created using DBCA, the hosts/ports configuration for ONS is added automatically. For other cases, it is needed to add the hosts manually using the `racgons` utility from the Oracle Clusterware home. To add hosts manually, ensure that ONS daemons are running on the cluster and that each daemon is aware of all other hosts in the configuration. Use `racgons` from the Oracle Clusterware bin directory (`ORA_CRS_HOME/bin/racgons`):

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

Example (to add rac_node2 to rac_node1 ONS):

```
racgons add_config rac_node2.us.oracle.com:6200
```

### 2.3.3.- FCF ONS configuration in OracleAS 10g R2

For OracleAS standalone instances that are not using dem to maintain the configuration of a farm, the ONS configuration can be modified by directly editing the ons.conf file under OracleASHome/opmn/conf. Example:

```
localport=6100
remoteport=6202
nodes=rac_node1.us.oracle.com:6200, rac_node2.us.oracle.com:6200,
     mt_node1.us.oracle.com:6202, mt_node2.us.oracle.com:
```

However, when a middle tier instance is part of a farm (either file-based or database-based), OracleAS Distributed Configuration Management (DCM) overwrites any configuration changes made to the ONS configuration file with the information about instances in the farm as reported by the OracleAS configuration repository. The OracleAS configuration repository only reports OracleAS instances and not database instances. Hence, in this type of “farm scenarios”, DCM constantly overwrites the modifications to ons.conf and eliminates any additions of RAC nodes (this will happen whenever an update of the configuration through DCM occurs). In this case, it is required to include the middle tier nodes in the ONS configuration of the Db nodes, this way the modifications done by DCM become irrelevant as it is the RAC nodes who maintain the ONS topology information.

For example, in order to configure middle tiers `mt_node1.us.oracle.com` and `mt_node2.us.oracle.com` in a FCF configuration with db nodes `rac_node1.us.oracle.com` and `rac_node2.us.oracle.com`, it is required to have the following configuration in the RAC nodes (edit ons.config under...
OracleDBHome/opmn/conf or use racgons depending on the Db version as per 2.3.1 and 2.3.2 above):

localport=6100
remoteport=6202
nodes=rac_node1.us.oracle.com:6200, rac_node2.us.oracle.com:6200,
     mt_node1.us.oracle.com:6202, mt_node2.us.oracle.com

The middle tier ONS configuration can be left as managed by DCM (i.e. only including the middle tiers as reported by the farm’s repository)

2.3.4.- FCF ONS configuration in OracleAS 10g R3

For OracleAS 10g R3 middle tiers, it is possible to modify opmn.xml in the middle tiers home to include the RAC nodes (no issues with DCM). This can be a better option than adding all the middle tiers to the RAC ONS configuration (if there is a large number of middle tiers than join and leave the ONS topology [5], it may become difficult to maintain the configuration on the DB side). Also, ONS has been enhanced in OracleAS 10g R3 ([5]) to admit different topologies that provide greater flexibility and efficiency. Addition of the RAC nodes to the middle tiers’ ONS configuration is achieved in this release by direct edit of the opmn.xml file in the middle tiers. Example:

<notification-server>
   <port local="6104" remote="6204" request="6007"/>
   <ssl enabled="false" wallet-file="&ORACLE_HOME;\opmn\conf\ssl.wlt\default"/>
   <topology>
      <nodes list="mt_node1.us.oralce.com:6200,rac_node1.us.oracle.com:6200,rac_node2.us.oracle.com:6200"/>
   </topology>
</notification-server>

The list after “nodes” node should list the RAC nodes’ addresses and ports as specified in the ons.config in the RAC nodes. If a firewall is present between Db and Middle Tier, alternative topologies can be used (gateways topology [5]). Notice that the <nodes> element for the RAC nodes is an addition to the ONS topology configuration, i.e. there may be additional <nodes> or <discovery> elements ([5]) in the topology configuration of the OracleAS instance that should not be modified/removed. To make these changes effective, opmnctl reload should be executed after modifying and saving the opmn.xml file.

In OracleAS 10 R3 SOA, these changes can be made directly form the OracleAS Control screens (in this case, it is not needed to run opmnctl reload). To do this, from the main “Cluster Topology” screen in OracleAS Control, click on the “Topology Network Configuration” link at the bottom right. Enter the list of RAC nodes in the different fields available for the topology [5].
Notice however, that OracleAS Control allows the definition of only one type of topology. i.e. if the topology element in opmn.xml includes more than one topology type, like:

```xml
<topology>
  <nodes list="mt_node1.us.oracle.com:6200, mt_node2.us.oracle.com:6200, rac_node1.us.oracle.com:6200,rac_node2.us.oracle.com:6200"/>
  <discover list="#225.0.0.1:6879"/>
</topology>
```

you will not be able to use OracleAS control to add elements to the nodes list while maintaining the discover entry (OracleAS Control allows the configuration of only one topology type as reflected in the above screen)

3. - TESTS AND VERIFICATIONS FOR FAST CONNECTION FAILOVER IN ORACLEAS ENVIRONMENTS

With the configuration steps described in the previous sections, the JDBC drivers should be ready to process Database events through ONS (failover of invalid connections and reassignment of connections to new RAC nodes should take place based on changes in the Db instances’ availability). It is however a good practice to verify that ONS is working properly and that intermediate elements such as firewalls or other network pieces are not preventing the correct behavior. The following aspects need to be verified for Fast Connection Failover to work properly:
o The ONS communication between RAC nodes and middle tiers works properly and ONS events generated by the database instances are correctly received by middle tiers
o The JDBC drivers are processing correctly these ONS events and failover and connection clean up works properly

3.1.- Verifying ONS communication between RAC nodes and middle tiers.

First, it should be verified that ONS is running in the RAC nodes. In order to do this execute:

```bash
onsctl ping
```

from OHOME/opmn/bin. This should confirm that ONS is running in each RAC node. In the middle tier nodes:

```bash
opmnctl ping
```

from OHOME/opmn/bin, should report the availability of ONS. Next, it is required to verify that ONS communication between RAC nodes and Middle tiers happens correctly. In the RAC nodes invoke:

```bash
onsctl debug
```

from OHOME/opmn/bin. This should report the ONS debug information. The Middle tier connections and the other Databases connections should be listed like in the following screen
In the middle tier nodes

opmnctl debug

from OHOME/opmn/bin should similarly list the connections from all the tiers (middle tiers and database nodes) in the debug report.
Additionally an **ONS tester** is available to test ONS communications between different tiers\(^4\). The tester can run as a sender of ONS notifications and also as a receiver. To run the tester, it is required to unzip the file in one of the RAC nodes and execute:

```
java -Doracle.ons.oraclehome=$ORACLE_HOME ONSTest -s 1 -r 0 -p -n 30
```

This will start a sender thread in the RAC nodes that will send 30 ONS messages of type “ons/test/shutdown”. An ONS receiver should then be started by executing in one of the OracleAS nodes:

```
java -Doracle.ons.oraclehome=$ORACLE_HOME ONSTest -s 0 -r 1 -p
```

The reception of notifications should be logged in the receiver’s console (in the OracleAS node) certifying that ONS notifications flow correctly between RAC nodes and OracleAS nodes. The following picture reflects the sample output that should be reported by the tester. Since other middle tiers may be generating ONS events, it should be verified that, among the events received, there are events of type “ons/test/*” which are the events that the sender thread in the tester will issue.

\(^4\) This tester is intended merely for debugging and testing purposes.
3.2.- Verifying that the JDBC drivers are processing ONS events and performing failover correctly

To verify that the JDBC drivers are receiving and processing correctly the ONS events, it is required to use JDBC tracing. JDBC tracing in Oracle 10g JDBC requires specific JDBC libraries. The libraries need to override the default/installed JDBC drivers available with OracleAS. The tracing drivers require jdk1.4.jar and can be downloaded from http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

**NOTE:** There are separate libraries for 10g R1 RAC and 10g R2. It needs to be noted that the manifest classpath in OracleAS 10g R2's oc4j.jar includes $OH/jdbc/lib/classes12dms.jar into the effective classpath and not ojdbc14dms.jar, so it is required to replace $OH/jdbc/lib/classes12dms.jar with $OH/jdbc/lib/ojdbc14dms_g.jar. This is just for tracing purposes and should be reverted after the tracing verifications have been completed.

To enable JDBC tracing, it is required to:

1.- Add the line

```
oracle.jdbc.driver.level=FINEST
```

to the end of the default $OH/jdk/jre/lib/logging.properties file.

2.- Edit $OH/opmn/conf/opmn.xml to start OC4J VMs with the system property

```
-Doracle.jdbc.LogFile=jdbc.log.
```

This will result in a file $OH/j2ee/Oc4J_home/jdbc.log containing exclusively JDBC driver logging output.
Alternatively, a specific logging configuration can be used by creating and using a separate tracing-properties file. The location of this file should be passed as a parameter to the OC4J JVM using the java.util.logging.config.file environment variable. Tracing should be enabled using the system property

```
oracle.jdbc.trace=true
```

Example: opmn.xml java-options are modified for the OC4J container running the FCF application with

```
-Doracle.jdbc.trace=true -Djava.util.logging.config.file="c:\tracing.properties"
```

The c:\tracing.properties files should contain

```
handlers= java.util.logging.FileHandler
.level= INFO

# 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
```

Once tracing is enabled per either one of the above procedures, RAC DB failures and connection failovers should be logged by the JDBC drivers, i.e the jdbc_tracing.log file should contain the corresponding messages for the FCF events as processed by the JDBC drivers. These events should include:

- Calls to initFailoverParameters
- FCF eventType and eventBody (showing the full event)
- Calls to abortConnection (for connections that being cleanedup)

The following portion includes the messages logged in a jdbc tracing log sample file:

```
FINE: OracleFailoverEventHandlerThread.handleEvent():<database/event/service>
Mar 3, 2005 3:07:43 PM oracle.jdbc.pool.OracleConnectionCacheManager verifyAndHandleEvent
FINER: eventType=256eventBody=<VERSION=1.0 service=serv1 instance=Rac3 database=RacStres host=hpcpq-7 status=down reason=user >
FINE:OracleImplicitConnectionCache.processFailoverEvent(eventType=256,instName=Rac3,dbUniqName=RacStres,hostName=hpcpq-7,status=down, card=0)
FINE:OracleImplicitConnectionCache.markDownLostConnections(serviceDownEvent=true,hostName=hpcpq-7,status=down)
FINE:OracleImplicitConnectionCache.cleanupFailoverConnections(serviceDownEvent=true,hostName=hpcpq-7,status=down)
```

---

5 When failures happen on the DB side, the failover of connections happens based on the notification triggered from the surviving RAC nodes and is not based on the ONS in the middle tier noticing this network failure. So the failover period is based on RAC tuning and not on ONS tuning. This failure detection period is database dependent and can be tuned. [10][11]
doForEveryCachedConnection
FINE: OracleImplicitConnectionCache.doForEveryCachedConnection(task=2)
Long Query Exception: java.sql.SQLException: Io exception: Socket closed
FINE: OracleFailoverEventHandlerThread.handleEvent():<database/event/service>
Mar 3, 2005 3:07:49 PM oracle.jdbc.pool.OracleConnectionCacheManager verifyAndHandleEvent
FINER: eventType=256eventBody=<VERSION=1.0 service=serv1 instance=Rac1 database=RacStres
host=hpcpq-5 status=up card=1 reason=user >

(Some “invalid ons event received” sqlexceptions may appear in the jdbc driver –
level tracing output. These can be disregarded as some non-relevant ONS events
are simply rejected in the jdbc processing)

SUMMARY.-

Fast Connection Failover is a clear demonstration of the high degree of integration
between OracleAS middle tiers and Oracle Real Application Cluster (RAC)
databases. FCF is a 10g RAC client provided by the JDBC Implicit Connection
Cache (ICC) that uses Oracle Notification Server (ONS) as the communication
vehicle for event communication between the two tiers. Fast Connection Failover
requires proper datasource and ONS configuration between middle tiers and
databases. This configuration may vary depending on the OracleAS and the Oracle
DB release. The flow of ONS notifications between these two tiers and the proper
consumption of them by the JDBC drivers are critical for a correct Fast
Connection Failover configuration. It is highly recommended to verify both as
described in this document
APPENDIX.

Additional documentation and further details on JDBC configuration, ONS configuration, Fast Connection Failover and Real Application Clusters is available per the following:

[1] OracleAS 10g R2 Datasources
[2] OracleAS 10g R3 Datasources
[3] Implicit Connection Cache and OC4J datasources
[4] OracleAS 10g R2 OPMN documentation
[5] OracleAS 10g R3 OPMN documentation
[6] Oracle 10g R1 DB FCF documentation
[7] Oracle 10g R2 DB FCF documentation
[8] Oracle 10g R2 DB racpons utility
[10] Oracle 10g R2 DB CSS reconfiguration
[11] Oracle 10g R1 DB CSS reconfiguration