# Transparent Application Failover

*An Oracle White Paper*
*June 2006*

ORACLE

# Transparent Application Failover

## INTRODUCTION

As businesses become increasingly dependent upon their database systems, interruptions in service become more and more disruptive. OracleDatabase 10g contains many availability features that ensure applications recover from system faults, with minimum effect on end-users. Minimizing interruption means more than just fast recovery. It also means preventing the disruption that occurs to users during a failover. Transparent Application Failover (TAF) can completely mask many failures from end-users, by preserving the state of their application and resuming any work that had been in progress at the time of failure. TAF also provides developers with tools to build applications that leverage TAF and make all failures, including those affecting transactions, transparent to end-users.

## FAILOVER CONCEPTS

Failover allows a database to recover on another system within a cluster. Figure 1 illustrates a typical database cluster configuration. Although the example shows a two-system cluster, larger clusters can be constructed. In a cold failover configuration, only one active instance can mount the database at a time. With Oracle Real Application Clusters, multiple instances can mount the database, speeding recovery from failures.

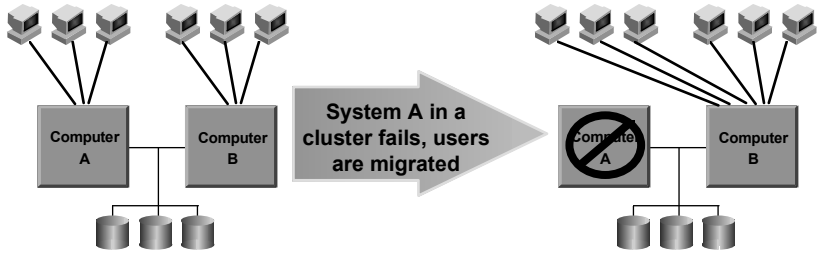


*Figure 1: Two System Cluster*

The failure of one instance will be detected by the surviving instances, which will assume the workload of the failed instance. Clients connected to the failed instance will migrate to a surviving instance. The mechanics of this migration will depend upon the cluster configuration. Transparent Application Failover feature will automatically reconnect client sessions to the database and minimize disruption to end-user applications.

## TAF FEATURES

### Automatic Reconnection

Oracle clients communicate with the database through an API called the Oracle® Call Interface (OCI). An OCI client initially connects to a listener process, usually located on the primary server in the cluster. The listener then provides the client with a connection to a server process. That server process will run on the same system as the primary instance, or, when using Real Application Clusters, may be located on another server, and utilize another Oracle instance. When a failure occurs, the OCI library on the client machine intercepts the error message and starts the transparent failover process. OCI will request another connection from the listener. The listener will then connect the client to a server process on the surviving system.

If the listener has failed, the OCI library can connect to a backup listener on another system in the cluster. This is done either using IP failover, or using the Network Services ability to connect to backup services in the event of failures.

There may be a delay associated with failing over to another server. Upon the loss of the connection, Oracle can perform a callback function that advises users that a failover is in progress, and to wait while the failover completes. This ensures users do not attempt to restart their clients, because they perceive this delay as a client failure. The client application developer can register this callback function with OCI, and it will be called by the library during the failover.

Once the new connection has been established and activated, OCI will perform other operations that mask the failover from the client application.

### Package and Session State Restoration

Database servers are not stateless applications. Each client session has its state maintained on the database, such as prepared cursors, PL/SQL™ package state, and language preferences. Prepared cursors will be transparently reestablished, whereas package and session state can be restored through the registered callback function after completion of the failover. To determine the context of the call, the callback function uses a parameter that the library passes, and can reestablish any package state lost during the failure. In addition, the callback function can replay at this time any ALTER SESSION commands normally executed when a client application first connects to the database. The function can also notify the user that failover has completed. If, for some reason, the failover does not succeed, the function can inform the user that it was unable to failover. If a client application has established multiple sessions with the database, the OCI library will call the function each time it fails over a session. Again, this is to allow the function to replay any desired ALTER SESSION commands.

As shown in Figure 2, following a failure, TAF can use callback functions to preserve application state on the server. Traditional IP failover techniques alone do

not preserve state on the database, and cannot completely mask a failure from a client application.

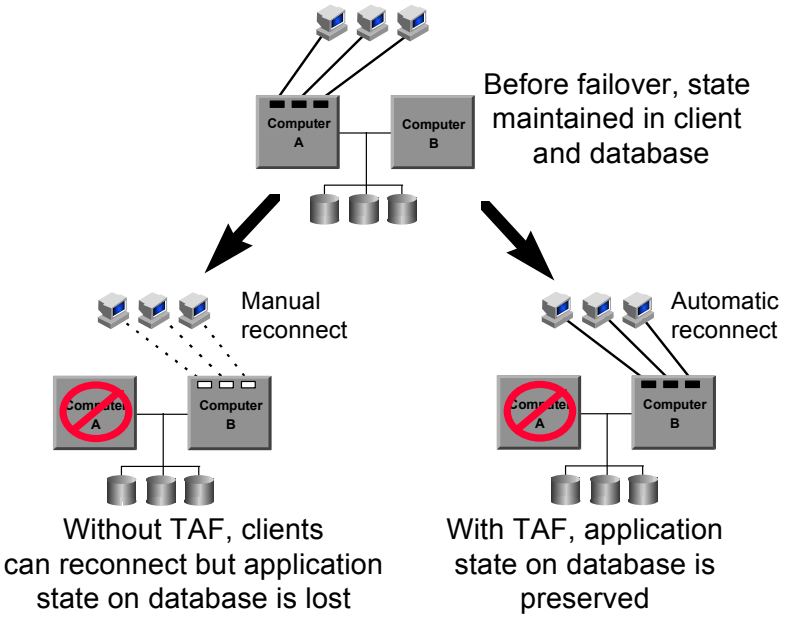


*Figure 2: Preserving Application State*

### Resumable Queries

Following a failure, if TAF is configured with TYPE=SELECT, cursors that were open at the time of failure will be implicitly re-positioned to the current row by the OCI library. As shown in figure 3, the open cursors are re-executed and the rows that had been previously returned to the application prior to the failure are discarded. Applications can then fetch from an open cursor to retrieve the next row in the result set. The users perceive that only one select statement had returned the result set. Oracle's read-consistency mechanism ensures that the results that the second execution returns are the same as the results that the original select statement returned. If the application has issued a query with an ORDER BY clause, consistent ordering between the two queries will guarantee the same rows are discarded from the second query's result set. As a final check, the OCI library calculates a checksum on the discarded rows, and compares it against those rows already returned. If a mismatch occurs, an error is returned.

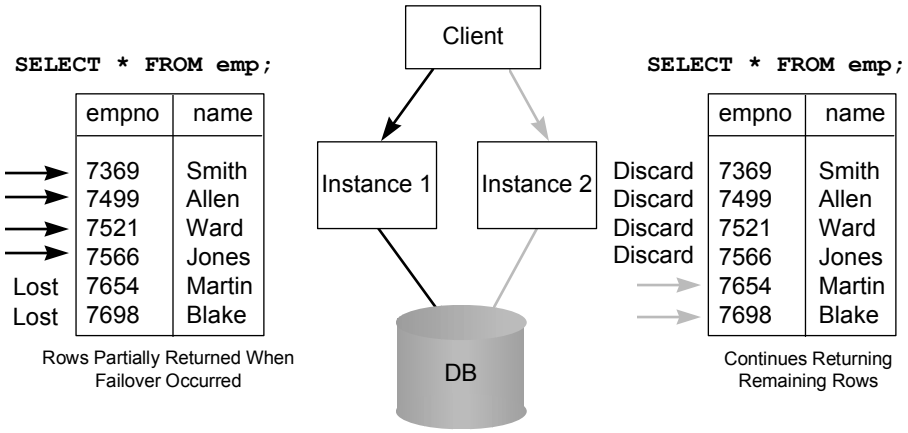


*Figure 3: Transparent Queries*

### Session Migration

Transparent Application Failover can allow an administrator to migrate sessions from one instance to another. This may be useful for balancing load or in preparation for maintenance of an instance or system. Sessions, whether individual sessions, or all sessions, can be disconnected, and TAF will re-establish those sessions on another instance or even at a standby or replica database. When preparing for system maintenance, once all sessions have migrated, the database shuts down and the administrator can perform maintenance activities. Such migrations can be deferred until all active update transactions are completed, which ensures the failover is transparent to end-users.

## TAF CONFIGURATION

The TAF configuration is specified in the connect string that clients use to connect to the database. In the connect string, an administrator can indicate whether sessions are reestablished and SELECT statements replayed, only sessions are reestablished, or no automatic failover is attempted. Using an LDAP server to define connect string aliases allows an administrator to quickly configure TAF for all clients on the network.

### Supported Configurations

Transparent Application Failover can be implemented on a wide variety of system configurations. All Oracle needs is another instance to reconnect to, in the case of a failure. In general, that instance should be available right away, but that is not required. TAF can be used with a single system, where the new instance is restarted on the same system once any repairs are made. TAF also supports all cluster failover topologies, including Real Application Clusters and Oracle Fail Safe (cold failover). TAF can even be used with remotely replicated or standby databases. To prevent any inconsistency, in case the underlying databases are not

identical, Oracle uses a checksum to validate that discarded rows are the same as those previously returned, and returns an error if they differ.

In order for a client application to use TAF, it must be written to use Oracle Call Interface Version 8. SQL*Plus®, ODBC and JDBC drivers, Oracle's .NET provider, and the Pro*C™ Precompiler all use Oracle Call Interface Version 8 and will support TAF. Any third party application written to use Oracle Call Interface Version 8 can also support TAF.

### Three-Tier Environments

The previous discussions have focused on using TAF in a two-tier environment. TAF can also be used in a three-tier environment, where an application server or transaction monitor sits between the database server and the client. In the event of a failure, TAF will transparently failover connections between an application server and a database server, reestablish all package and session states on the database server, and replay any queries that were in progress at the time of the failure. Using TAF, developers can create multi-tier solutions that completely mask database server failures from other tiers in the environment.

## CONFIGURING TRANSPARENT APPLICATION FAILOVER

TAF involves manual configuration of a net service name that includes the FAILOVER_MODE parameter included in the CONNECT_DATA portion of the connect descriptor.

### FAILOVER_MODE Parameters

The FAILOVER_MODE parameter must be included in the CONNECT_DATA portion of a connect descriptor. FAILOVER_MODE may contain the following parameters:

| tnsnames.ora file Parameter | Description |
|---|---|
| TYPE **(Required)** | Specifies the type of failover. Three types of Net8 failover functionality are available by default to Oracle Call Interface (OCI) applications:<br>SESSION: Fails over the session; that is, if a user's connection is lost, a new session is automatically created for the user on the backup. This type of failover does not attempt to recover selects.<br>SELECT: Enables users with open cursors to continue fetching on them after failure. However, this mode involves overhead on the client side in normal select operations.<br>NONE: This is the default. in which no failover functionality is |

| tnsnames.ora file Parameter | Description |
|---|---|
| | used. This can also be explicitly specified to prevent failover from happening. |
| METHOD | Determines how fast failover occurs from the primary node to the backup node:<br>BASIC: Establishes connections at failover time. This option requires almost no work on the backup server until failover time.<br>PRECONNECT: Pre-establishes connections. This provides faster failover but requires that the backup instance be able to support all connections from every supported instance. |
| BACKUP | Specifies a different net service name for backup connections. Specifying a BACKUP is strongly recommended for BASIC methods; otherwise, reconnection may first attempt the instance that has just failed, adding additional delay until the client reconnects. You may not specify a BACKUP if LOAD_BALANCING=ON. |
| RETRIES | Specifies the number of times to attempt to connect. If DELAY is specified, RETRIES defaults to five retry attempts. Note this only affects reconnection after failure. It does not affect the initial connection attempt. This parameter is ignored if a callback is registered. |
| DELAY | Specifies the amount of time in seconds to wait between connect attempts. If RETRIES is specified, DELAY defaults to one second. This parameter is ignored if a callback is registered. |

### Implementing TAF with Connect-Time Failover and Client Load Balancing

TAF can be implemented with connect-time failover and client load balancing for multiple addresses. In the following example, Net8 connects randomly to one of the listener addresses on sales1-server or sales2-server. If the instance fails after the connection, Net8 fails over to the other node's listener, resuming any SELECT statements in progress.

```
sales.us.acme.com=
 (description=
  (load_balance=on)
```

```
 (failover=on)
 (address=
     (protocol=tcp)
     (host=sales1-server)
     (port=1521))
(address=
     (protocol=tcp)
     (host=sales2-server)
     (port=1521))
 (connect_data=
   (service_name=sales.us.acme.com)
   (failover_mode=
    (type=select)
    (method=basic))))
```

### Retrying a Connection

TAF also provides the ability, with the RETRIES and DELAY parameters, to automatically retry re-connecting on failover if the first re-connection attempt fails. In the following example, (after a failure) Net Services tries to connect to the listener on sales1-server. If the connection attempt fails, Net Services waits 15 seconds before trying to connect again. Net Services attempts to connect up to 20 times. Note, DELAY and RETRIES only affects attempts to reconnect after a failure. It does not affect initial connection attempts, or initial pre-connection attempts. In addition, DELAY and RETRIES are ignored if a callback is registered. In this case the callback is responsible for determining the appropriate number of retries and the delay.

```
sales.us.acme.com=
 (description=
  (address=
     (protocol=tcp)
     (host=sales1-server)
     (port=1521))
  (connect_data=
    (service_name=sales.us.acme.com)
    (failover_mode=
     (type=select)
     (method=basic)
     (retries=20)
     (delay=15))))
```

### Pre-Establishing a Connection

A backup connection can be pre-established. The initial and backup connections must be explicitly specified. In the following example, Net Services connects to the

listener on sales1-server. If sales1-server fails after the connection, Net Services fails over to sales2-server, resuming any SELECT statements in progress.

```
sales.acme.com=
 (description=
  (address=
      (protocol=tcp)
      (host=sales1-server)
      (port=1521))
  (connect_data=
     (service_name=sales.us.acme.com)
     (instance_name=sales1)
     (failover_mode=
       (backup=sales2.acme.com)
       (type=select)
       (method=preconnect))))
sales2.acme.com=
 (description=
  (address=
      (protocol=tcp)
      (host=sales2-server)
      (port=1521))
  (connect_data=
     (service_name=sales.us.acme.com)
     (instance_name=sales2)))
```

### Verification

You can query FAILOVER_TYPE, FAILOVER_METHOD, and FAILED_OVER columns in the V$SESSION view to verify that TAF is correctly configured.

## SPECIAL CONSIDERATIONS

There are a few special considerations that should be taken into account when using TAF. The following section explains these considerations.

### Transactions

Sessions with active update transactions (UPDATE, INSERT, DELETE) at the time of the failure will automatically be reconnected to a new session, but their uncommitted transactions will be rolled back during recovery. Oracle will return an error message to the application, stating that a rollback must be issued. The application, if failure aware, can automatically issue the rollback and reissue the transaction, completely masking the failure from the end-user. Alternatively, the user must initiate the rollback and reissue the transaction. In reality, the rollback is

performed automatically during recovery. Oracle only requires the explicit rollback statement, to ensure the application or user is aware the transaction has been rolled back.

Applications that wish to completely mask failures should rollback and then retry the transaction if they encounter a TAF error. Many applications will already use the same rollback/retry technique to handle deadlock errors reported by the database; it is often possible to amend existing program logic for deadlock errors to allow applications to cater for errors reported by TAF.

If there is no existing program logic for deadlock errors – for example if the application simply reports an error and then closes the database connection – then a similarly basic approach might also be appropriate for TAF errors.

| Term | Explanation |
|---|---|
| Active Transaction | One or more database updates has been performed |
| No Active Transaction | No database update has been performed. Queries may have been performed since completing the previous active transaction with COMMIT or ROLLBACK. |
| Active Connection | A client/server database interaction is active at the time of failure |
| Inactive Connection | No client/server database interaction is active at the time of failure; the client and server are idle |

If the connection has an Active Transaction at the time of the failure then:

- The transaction will be rolled back
- An error will be reported
    - For the current database interaction for an Active Connection
    - For the next database interaction for an Inactive Connection
- The application must issue a ROLLBACK statement before any subsequent database interactions will succeed

If the connection has No Active Transaction at the time of the failure then:

- No error need be reported depending on Oracle Net configuration parameters

- The application can start a new Active Transaction by performing database updates in the usual way

### Network

If a system crashes, or the network fails, there may be nothing on the other end of the network to quickly identify a failure. When using Oracle Database 10g Release 2 with Real Application Clusters, clients can subscribe to FAN events (see below) to quickly receive notification of a failure. However, if not using database and client release 10.2 and later, clients must identify failures because a timeout expires. The timeout will vary, depending upon the client state at the time of the failure. The client may be in one of the following states:

- **Disconnected**: The client is not currently connected to the failed node
- **Connected/with interrupt**: The client is connected to a node that incurs an outage. At the time of the failure, the socket is closed cleanly. The client will receive an interrupt and the connection will close.
- **Connected/no interrupt/not waiting**: The client is connected to a node that incurs an outage. It is not waiting for a read() or write() call to return. The outage panics the node and fails to close sockets (for example, a kernel panic). The next time the session tries to communicate with the instance, it will waits until it realizes the instance is no longer available.
- **Connected/no interrupt/waiting**: The client is connected to a node that incurs an outage. It is waiting for a read() or write() call to return. The outage panics the node and fails to close sockets (for example, a kernel panic). The sessions will wait an extended period of time until they realize there has been a failure and they attempt to connect to a backup instance.

The table below describes behavior in each scenarios. The default values are for HP:

| Client State at time of failure | Behavior | TCP parameter affecting delay | Default (HP) |
|---|---|---|---|
| Disconnected | When the client attempts to connect to the node using the failed node's IP address, it will wait for the connection to succeed until the tcp_ip_abort_cinterval is exceeded. It will then fail and attempt | tcp_ip_abort_cinterval | 75 seconds |

| | | | |
|---|---|---|---|
| | to connect to the backup (as specified in the TNSnames) | | |
| Connected/with interrupt | The client will be immediately be disconnected and ready to reconnect. See note 1. | n/a | n/a |
| Connected/no interrupt/idle | As soon as the client attempts to use the connection, it will wait for a timeout to expire before failing. If the operation is a write, it will wait tcp_ip_abort_interval before failing, and if it is a read, it will wait tcp_keepalive_interval before failing. See note 1. | tcp_ip_abort_interval (write) tcp_keepalive_interval (read) | 10 minutes (write) 2 hours (read) |
| Connected/no interrupt/waiting | Client will wait for a response for tcp_keepalive_interval before failing. See note 1. | tcp_keepalive_interval | 2 hours |

*Note 1: After the connections fail, the clients will be ready to reconnect. If a TAF backup is specified, the client will then automatically connect to the backup with no additional delay. If a TAF backup is not specified, the client will first attempt to reconnect to the failed node as described in the Disconnected scenario, and will experience the tcp_ip_abort_cinterval delay. This assumes Net Services is configured to try addresses in the same order. If client load balancing is in use, the client may try to connect to the backup node rather than the failed node, depending on which address it randomly chooses.*

A good way to reduce delays is to move the IP address of the failed server to a surviving server. This will reduce the reduce the delays for new connections. It can also interrupt existing, but idle connections. Connecting to the database using a virtual IP that can be moved after a failure is a general best practice, recommended for use with both RAC and cold failover clusters. Beginning with release 10.1 of Oracle Database 10g, this capability is inherent in the included Oracle Clusterware, and thus available for all major platforms.

If you choose to use Net Services alone to provide failover, you may want to consider tuning the above parameters to reduce failover time. Unfortunately, this will have possibly undesirable performance effects on network traffic and also may cause false failovers. Some suggestions:

- Reducing the TCP connect abort interval (tcp_ip_abort_cinterval) to optimise connect time failover. This optimisation is not usually required on systems with virtual IP capabilities.
- Tuning TCP keepalive to optimize failover for clients waiting for a response from the server:
    - Enable keepalive, by specifying the "(ENABLE=BROKEN)" parameter in the DESCRIPTION clause in the connect descriptor.
    - Reduce the keepalive interval (tcp_keepalive_interval) from the default (2 hours) to a value that more closely approximates the database failover time (for example 20 seconds to one minute).

### Session State

TAF does not reconstruct database session state following connection failover:

- Any database changes performed by an active transaction are rolled back and any locks are released
- Any PL/SQL session state is lost
- The effects of any ALTER SESSION commands are lost

You can register a call-back function to be triggered on failover. This function can potentially re-establish some types of lost state.

### TAF Error Codes

The error codes reported by TAF are in the range ORA-25400 to ORA-25425.

Any other error codes encountered following the failure of an existing connection imply that TAF has been unable to failover the connection. The application should assume that the connection has been lost; it will not be possible to use a rollback/retry mechanism.

In practice only the following error codes are likely to be reported by TAF:

| Error Code | Error Message | Notes |
|---|---|---|
| ORA-25401 | can not continue fetches | Usually for Oracle Net TYPE=SESSION with a query on an Active Connection with No |

| | | |
|---|---|---|
| | | Active Transaction |
| ORA-25402 | transaction must roll back | Most common error code. Reported for all subsequent database interactions until a ROLLBACK is performed. |
| ORA-25405 | transaction status unknown | Very rare due to the short duration of a COMMIT operation |
| ORA-25408 | can not safely replay call | Only for the first database update that creates an Active Transaction |
| ORA-25425 | connection lost during rollback | Usually rare due to the low likelihood of performing a ROLLBACK operation at the time of the failure, or after a failover but prior to any other database interaction |

### Non-TAF Error Codes

If TAF is unable to establish a new connection then other error messages may be encountered. These error codes are not TAF errors but are typically the same errors that might be reported when attempting an initial connection to an Oracle database.

TAF may be unable to establish a new connection for a variety of reasons:

- Limitations in the TAF configuration parameters of Oracle Net, for example incorrect setting of RETRIES and DELAY parameters in cold failover clusters
- Target database instance is registered with the appropriate Oracle Net listener but has restricted availability or is starting or stopping
- Target database instance is also unavailable

TAF will also appear to be unable to establish a new connection in the event of syntax errors in the TAF configuration parameters of Oracle Net.

The application should assume that the connection has been lost; it will not be possible to use a rollback/retry mechanism.

The following list of errors covers the most likely error conditions but is not necessarily exhaustive, particularly for Oracle Net errors where site-specific configuration issues may be involved:

| Error Code | Error Message |
|---|---|
| ORA-01012 | not logged on |
| ORA-01033 | ORACLE initialization or shutdown in progress |
| ORA-01034 | ORACLE not available |
| ORA-01089 | immediate shutdown in progress - no operations are permitted |
| ORA-03113 | end-of-file on communication channel |
| ORA-03114 | not connected to ORACLE |
| ORA-12203 | TNS:unable to connect to destination |
| ORA-12500 | TNS:listener failed to start a dedicated server process |
| ORA-12571 | TNS:packet writer failure |
| ORA-12153 | TNS: not connected (returned if FAN, but not TAF, is enabled) |

### Server (Shadow) Process Failures

TAF is designed to protect from instance failures. If the dedicated server process corresponding to a client process fails, the instance has not failed, and, TAF will not failover the client associated with the failed process. Such a condition is very rare, and is likely related to the session state associated with that connection. Failover would recreate the same conditions, likely leading to another failure. More commonly, server processes are intentionally killed in order to attempt to test TAF. If you wish to test a TAF configuration without shutting down the entire instance, use the ALTER SYSTEM DISCONNECT SESSION command. TAF will recover the connection dropped when this command is issued.

## FAST APPLICATION NOTIFICATION

Fast Application Notification (FAN) is implemented as part of the high availability framework implemented through Oracle Cluster Ready Services (CRS) in conjunction with Oracle Database 10g Real Application Clusters (RAC).

FAN eliminates dependencies on error detection, handling and reporting characteristics of the underlying TCP/IP stack. FAN also provides a superior mechanism for notifiying the application of changes in the state of the database and RAC instances.

As of Oracle Database 10g Release 2 (10.2), OCI client applications support FAN events. This means that any client application can subscribe to these events and will receive notifications in the event there has been a failure. This will eliminate the failover delays described above that a client may experience due to network timeouts. Rather than waiting for timeouts to expire, CRS will interrupt clients and quickly force TAF to reconnect.

In order to subscribe to FAN events, clients must be multi-threaded and linked with a 10.2 or later Oracle client.

## FAST CONNECTION FAILOVER

Fast Connection Failover (FCF) is a packaged implementation of FAN included in Oracle Application Server 10g. In the event of a failure, it will act on the FAN event, clean up stale connections, and pass the notification to the application as a SQL exception. The application can then retry the connection and replay the transaction. It works with JDBC connection pools, and is not dependent on OCI or TAF. Thus, you can use FCF with both the thin JDBC driver as well as the thick (OCI) JDBC driver. Oracle does not recommend the use of both TAF and FCF with the same application.

## CONCLUSION

Transparent Application Failover is a powerful tool for improving application availability, by masking failures from client applications. Most importantly, TAF prevents the disruption associated with failures, automatically reestablishes connections to a backup instance, and continues failed queries from the point they stopped. TAF works with many different configurations, including single systems, clusters, and geographically separated systems. It can help administrators perform maintenance and load balancing activities, by eliminating disruption to end-users as they migrate from one system to another. TAF and the rest of the Oracle Database 10g high availability features will keep all your critical data accessible, all the time.

Transparent Application Failover
June 2005
Author: Bob Thome
Contributing Authors: Mike Hallas

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com