Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 10*g* Release 2

Oracle Maximum Availability Architecture White Paper
June 2009

Maximum Availability Architecture

Oracle Best Practices for High Availability

ORACLE

Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 10*g* Release 2

Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 10*g* Release 2

INTRODUCTION

This white paper provides Oracle Maximum Availability Architecture (MAA), instructions and best practices for configuring automatic client failover for OCI, OLE DB, and JDBC clients in a Data Guard configuration.

The information provided complements previous MAA best practice papers that describe manual and automatic database failover in "Switchover and Failover Best Practices: Oracle Data Guard 10*g* Release 2" [1] and "Fast-Start Failover Best Practices: Oracle Data Guard 10*g* Release 2" [2]. Updates to these and many other MAA white papers may be found on MAA page [3] on the Oracle Technology Network.

OVERVIEW

Failovers can be sorted into one of the following broad categories:

1. Complete-site failover utilizes a secondary location to host a Data Guard standby database and a completely redundant set of middle-tier application servers. When failing over to the secondary site, the middle-tier servers are started and a network load balancer is redirected to the new primary site. A Data Guard failover will transition the standby database at the secondary location to the primary role. The Oracle MAA paper, Oracle Database High Availability Best Practices [4], provides guidance for implementing complete site failover.
2. When a node within an Oracle RAC fails, clients attached to the failed node must be quickly notified that a failure has occurred, and must reconnect to the surviving nodes in the cluster to continue processing. The technical white paper, Workload Management with Oracle Real Application Clusters [5], describes details for handling node failure within an Oracle RAC.
3. Partial-site failover occurs when the primary database has become unavailable but the primary site remains intact, and affected clients must be redirected to a new primary database at a secondary location following a Data Guard [6] failover. This best practice paper addresses automating client failover for this type of failure.

At a high level, automating client failover in a Data Guard configuration includes relocating *Database Services* to the new primary database as part of a Data Guard failover, notifying clients that a failure has occurred in order to break them out of TCP timeout, and redirecting clients to the new primary database.

Service Relocation

Oracle Database 10g introduced an automatic workload management facility for Oracle RAC and single instance (non-RAC) databases, called Database Services [7], that enable you to group database workloads and easily designate computing resources to service that workload. You can define a database service for a particular application, such as the application `'sales'` used in the examples in this paper, and should the primary database offering the service fail, the service can be automatically relocated to the new primary database a part of a Data Guard failover.

Within an Oracle RAC cluster users are able to access a service independent of the instance providing it because, using listener registration, all listeners in a cluster are aware of which instance is currently providing the service when a connection request is received. If the instance providing the service fails, Oracle RAC quickly relocates the service to a surviving instance within the cluster.

The same concept of service relocation applies to a Data Guard configuration. To illustrate how this occurs, assume a simple configuration with a single node primary and standby database, and a database service named `'sales'` configured with the appropriate HA settings (discussed in detail later in this paper). Client applications connect to the service `'sales'` using an Oracle Net alias that includes all hosts, both primary and standby, in the configuration. Unwanted connection attempts to the standby database are prevented because the service name used in the Oracle Net alias only runs on the instances for the primary database. Relocation of the service to the new primary is automated by use of a trigger (described in detail later in this paper) that fires when a Data Guard role change transitions the standby database to the primary role, starting the service on the new primary database.

Client Notification and Reconnection

Continuing with the example above, client notification and reconnection prevents clients that are connected to the original primary at the time of failure from falling into a hung-state waiting for lengthy TCP timeouts to expire. Oracle will notify these clients that a failure has occurred, break them out of TCP timeout, and have both new and existing connections directed to surviving RAC nodes, or to the new primary database following a Data Guard failover.

OCI Clients: Notification and reconnection is accomplished for OCI clients in a Data Guard configuration using Fast Application Notification (FAN) and Transparent Application Failover (TAF) in the following manner:

- When an OCI client makes a connection to a primary database, the `REG$` table is updated to includes the IP address for the client. This data is automatically replicated to the physical standby.
- Upon failover, the Data Guard broker inserts a DB Down event into the database alert queue on the standby which is then sent to all clients having an entry in the `REG$` table. Note that this method of automating client failover requires the failover to be managed using the Data Guard broker.
- When clients receive the DB Down event they immediately disconnect from the failed primary.
- TAF enables OCI clients to automatically attempt to create a new connection to the service `'sales'`, used in the above example.

JDBC Clients: Notification and reconnection is accomplished for JDBC clients in a Data Guard configuration using Oracle Notification Services (ONS) and Fast Connection Failover (FCF) in the following manner:

- At startup, JDBC applications establish communication channels with ONS daemons running on all primary and standby hosts.
- FAN ONS events are created when database services are started/stopped or Oracle instances within a cluster are started/stopped. These FAN ONS events are published via the ONS daemons. JDBC applications consume the events and take action to their connection cache appropriate to the event.
    - **Note:** When a single node primary database or all nodes of a RAC primary database become unavailable, there is no longer a daemon present to publish FAN ONS events for the failed primary database. Automating client failover requires that the standby database be configured to publish database down events on behalf of the failed primary.
- Upon a Data Guard failover, a trigger is fired based on the DB_ROLE_CHANGE system event to call a publisher program (provided by Oracle and described in this paper). The publisher program reads a configuration file previously created by the administrator and generates a database down FAN ONS event on behalf of the failed primary database.
- The ONS daemons on the standby host publish the database down event. JDBC clients still connected to the failed primary receive the event and perform the appropriate action to their connection cache.
- FCF enables JDBC clients to automatically attempt to create a new connection to the service `'sales'`, used in the above example.

Connect Timeout: For both OCI and JDBC clients you must also insure that once clients have been notified of the failure they are not subject to TCP timeouts if they subsequently attempt to reconnect to a failed host. This is accomplished using SQLNET outbound connect timeout such that clients will quickly attempt to connect to the next host in an address list if the first is not available.

Configuration details for OCI, OLE DB, and JDBC clients along with examples of triggers used to relocate database services and publish HA events for JDBC clients are described later in this best practice paper.

Client Failover Components

In Oracle Database 10*g* Release 2, the following features provide for the timely failover of clients and minimize the impact of outages for both planned and unplanned scenarios.

- **Connect Time Failover**: Redirects failed connection requests to a secondary listener.
- **Transparent Application Failover (TAF)**: Used for OCI clients, enables client applications to automatically reconnect to a database if the original connection fails. TAF only fails over the session and `SELECT` statements. While `SELECT` statements are automatically restarted in the new session (providing that TAF is configured for SELECT failover), any `INSERT`, `UPDATE`, or `DELETE` transactions must be rolled back by the application. In addition, any session customizations (for example, `ALTER SESSION` statements) must be re-executed by the application. Other types of process state such as PL/SQL session level variables are not reestablished, but can be via a TAF callback.
- **Fast Application Notification (FAN)**: Provides quick notification when a resource (such as an instance, service, node, or database) fails. FAN is available to all applications by using either Fast Connection Failover [8] with a FAN-integrated Oracle client (clients using JDBC, OCI, OLE DB) or by using the FAN API to directly read FAN events.
- **Fast Connection Failover**: Provides fast failover of database connections by allowing you to configure FAN-integrated JDBC clients to automatically subscribe to FAN HA events and react to service, instance, and database `UP` and `DOWN` events.
- **DB_ROLE_CHANGE system event**: Is fired when the primary database is first opened after a Data Guard role transition has occurred. Using this system event, a trigger can be written to perform vital post-role change actions.

Client Wait Times for Site, Node, and Instance Outages

Because not all of the features listed above are available for all Oracle Database releases, the time required for clients to respond to various outages will vary by

release. Specific timings are provided in Table 1. The time required for failover in certain cases is a direct function of TCP/IP network timeouts.

Table 1: Typical Wait Times for Client Failover

Oracle Version	Client Type	Site Failure (RAC and Non-RAC)	RAC Node Failure	Non-RAC Instance Failure	RAC Instance Failure
9*i*	All	TCP timeout	TCP timeout	Seconds to minutes *	Seconds
10.1	JDBC	TCP timeout	Seconds	Seconds to minutes *	Seconds
	OCI	TCP timeout	TCP timeout	Seconds to minutes *	Seconds
	OLE DB	TCP timeout	TCP timeout	Seconds to minutes *	Seconds
10.2	JDBC	Seconds	Seconds	Seconds	Seconds
	OCI	Seconds	Seconds	Seconds	Seconds
	OLE DB	Seconds	Seconds	Seconds	Seconds

* The wait times required in non-RAC instance failures is determined by how much time is needed to activate the standby database as the new primary database and for the client to establish a new connection.

The best practices for the configurations described in this paper make it possible to achieve the following client-failover times for OCI, OLE DB, and JDBC clients for Oracle Database 10g Release 2:

- Less than 10 seconds for any instance or node failures in a RAC database
- Less than 30 seconds for any cluster-wide, network, or site failures using Data Guard Fast-Start Failover (excluding time required to satisfy the Fast-Start Failover threshold, discussed below)

Note: In non-RAC configurations, or in the event of a Data Guard failover, ODP.Net clients will incur an outage equal to that of TCP timeout. ODP .Net clients are not included as part of this study therefore configuration recommendations for ODP .Net clients are not provided at this time.

OCI CLIENTS

Database Configuration for OCI Clients

The following configuration steps assume that the OCI application can subscribe to FAN events. If the OCI application doesn't meet the requirements for FAN then efficient failover can still be achieved using timeouts and application retries (see Appendix C for more information).

Configure your RAC or single instance, (non RAC) databases to automate failover for OCI clients according to the following instructions:

1. The steps below require the Data Guard configuration to be managed by the Data Guard Broker [9] if you wish HA notifications to be sent at failover time in order to break OCI clients out of TCP timeout.

2. If you are configuring a RAC database, Oracle recommends using the Oracle Enterprise Manager Cluster Managed Services Page to create database services that client applications will use to connect to the primary database. For complete instructions on creating database services, please see the chapter on Workload Management in the *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide* [7].

3. If you have created the database service as instructed in step 2, above, use the `DBMS_SERVICE` package to modify the service to enable high-availability notification to be sent through Advanced Queuing (AQ) by setting the `AQ_HA_NOTIFICATIONS` attribute to `TRUE`. To configure server-side TAF settings, set the `FAILOVER` attributes, as shown in the following example:

```
exec DBMS_SERVICE.MODIFY_SERVICE(
 service_name => 'salesOCI',
        aq_ha_notifications => true,
        failover_method => 'BASIC',
        failover_type => 'SELECT',
        failover_retries => 180,
        failover_delay => 1);
```

   Note: Refer to the *Oracle Database PL/SQL Packages and Types Reference* [10] for more information about the `DBMS_SERVICE` package.

4. If you are creating database services for a single instance (non-RAC) database, or if you are creating services for a RAC database and are not using Enterprise Manager, use the CREATE_SERVICE subprogram illustrated in the following example to both create the database service and enable high-availability notification and configure server-side TAF settings:

```
exec DBMS_SERVICE.CREATE_SERVICE (
        service_name => 'salesOCI',
        network_name => 'salesOCI',
        aq_ha_notifications => true,
        failover_method => 'BASIC',
        failover_type => 'SELECT',
        failover_retries => 180,
        failover_delay => 1);
```

5. Create a trigger that fires on the system startup event to relocate the database service `'salesOCI'`, in the above example, to a Data Guard standby database (RAC or non-RAC) after it has transitioned to the primary role.

```
CREATE OR REPLACE TRIGGER manage_OCIservice
after startup on  database
```

```
DECLARE
    role VARCHAR(30);
BEGIN
    SELECT DATABASE_ROLE INTO role FROM V$DATABASE;
    IF role = 'PRIMARY' THEN
       DBMS_SERVICE.START_SERVICE('salesOCI');
END IF;
    END;
```

If the Data Guard configuration is not using real-time apply, then archive the current redo log file to be sure that the changes are applied to the standby database:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

OCI Client Configuration

The following configuration steps assume that the OCI application can subscribe to FAN events. If the OCI application doesn't meet the requirements for FAN then efficient failover can still be achieved using timeouts and application retries (see Appendix C for more information).

Configure OCI clients to receive notification of FAN HA events and to avoid re-connecting to a failed instance:

1. Refer to MetaLink Note 405120.1 for prerequisites required to configure automatic client failover for Oracle Database 10.2.0.2 or 10.2.0.3.
2. Create an Oracle Net service name that includes all primary and standby hosts in the ADDRESS_LIST. The following example

```
SALESOCI =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON1)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON2)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO1)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO2)(PORT = 1521))
     (LOAD_BALANCE = yes)
   )
   (CONNECT_DATA=
     (SERVICE_NAME=salesOCI)
   )
  )
```

3. Initialize the environment with the `OCI_EVENTS` parameter to enable OCI clients to receive FAN notifications. For example:

```
OCIEnvCreate(...OCI_EVENTS...)
```

See the *Oracle Call Interface Programmer's Guide* [11] for more information.

4. Link the OCI client applications with the thread library (`libthread` or `libpthread`).
5. In the client side SQLNET.ORA file, set the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter. This parameter enables clients to quickly traverse an address_list in the event of a failure. For example, if a client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter, after which the client attempts to connect to the next host in the address_list. This behavior continues for each host in the address_list until a connection is made. Setting the parameter to a value of 3 seconds will suffice in most environments.
6. After AQ HA notifications have been enabled, clients and applications can register a callback that is invoked whenever an HA event occurs. This enables an application to perform an action, such as replay the last transaction, whenever the callback is received (the callback is received as part of the event notification). Consider the following example of how to initialize an event callback:

```
OCIAttrSet(envhp, (ub4) OCI_HTYPE_ENV, (dvoid *)evtcallback_fn,
    (ub4) 0, (ub4)OCI_ATTR_EVTCBK, errhp);
OCIAttrSet(envhp, (ub4) OCI_HTYPE_ENV, (dvoid *)evtctx,
    (ub4) 0, (ub4)OCI_ATTR_EVTCTX, errhp);
```

Note: After registering an event callback and context, OCI will call the registered function once per HA event. For more information on registering a callback, refer to *Oracle Call Interface Programmer's Guide* [11].

Note: See Appendix B for information about troubleshooting OCI Client failover.

OLE DB CLIENTS

To configure automatic client failover for OLE DB Clients perform the identical database configuration steps described above for OCI Clients, and then configure the OLE DB clients to receive notification of FAN HA events as described below:

1. Set the following OraOLEDB connection string attributes:
    a. DBNotifications = true
    This can also be set via the registry.
    b. DBNotificationPort = [unsigned integer]
    Setting the DBNotificationPort attribute allows the port to specifically specified. If this attribute is not set then the port is randomly selected.
2. In the client side SQLNET.ORA file, set the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter. This parameter enables clients to quickly traverse an address_list in the event of a failure. For example, if a client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter, after which the client attempts to connect to the next host in the address_list. This behavior continues for each host in the address_list until a connection is made. Setting the parameter to a value of 3 seconds will suffice in most environments.

JDBC CLIENTS

The configuration differences between JDBC and OCI clients derive from the fact that JDBC clients use FCF and not TAF. This explains why in the example below, Database Services for JDBC clients are not configured for AQ HA events. Instead, a trigger is required to notify JDBC clients when a Data Guard failover occurs.

If your application only supports JDBC clients in your configuration, then simply follow the instructions below to configure automatic client failover. If you support a mixture of OCI and JDBC clients, you will need to follow both the OCI and JDBC sections of this paper, and configure database services with HA notifications appropriate for each client.

Instructions for JDBC clients are provided below.

Database Configuration for JDBC Clients

The following configuration steps assume that the JDBC application can subscribe to FAN events. If the JDBC application doesn't meet the requirements for FAN then efficient failover can still be achieved using timeouts and application retries (see Appendix C for more information).

Configure your RAC or single instance, (non RAC) databases to automate failover for JDBC clients as follows:

1. If you are configuring a RAC database, Oracle recommends using Oracle Enterprise Manager Cluster Managed Services Page to create database services that client applications will use to connect to the primary database. For complete instructions on creating database services, please see the chapter on Workload Management in the *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide* [7]. Add all hosts in the cluster to the RAC ONS configuration.

2. If you are creating database services for a single instance (non-RAC) database, or if you are creating services for a RAC database and are not using Enterprise Manager, use the CREATE_SERVICE subprogram illustrated in the following example to create the database service `'salesJDBC' for JDBC clients`:

```
exec DBMS_SERVICE.CREATE_SERVICE (
       service_name => 'salesJDBC',
       network_name => 'salesJDBC',
```

3. If you are creating a single-instance (non-RAC) database, skip to step 4. If you are creating a RAC database using database creation assistant (DBCA), the hosts are added automatically and you can also skip this step. Otherwise, you must add or remove nodes manually using `racgons` from the Oracle Clusterware home. To add or remove nodes manually, ensure that ONS daemons are running on the cluster and that each daemon is aware of all other nodes in the configuration.

   To add more middle-tier nodes or to update the RAC nodes, use `racgons` in the Oracle Clusterware `bin` directory:

   - To add the ONS daemon configuration, issue the following command:
     `racgons add_config hostname:port [hostname:port] ...`
   - To remove the ONS daemon configuration, issue the following command:
     `racgons remove_config hostname[:port] [hostname:port] ...`

4. In the previous steps, ONS daemons used to send and receive FAN events on the primary and standby were installed automatically with the Oracle Clusterware installation. However, in a single-instance (non-RAC) environment, configuring and starting the ONS daemons must be done manually. Perform the following steps to configure and start ONS daemons

on all hosts that have the potential run a single-instance (non-RAC) primary database:

Use an ONS configuration file to configure ONS. This file should exist in $ORACLE_HOME/opmn/conf directory after installation of the Oracle software stack. It should be configured similar to the following example:

```
localport=6100
remoteport=4200
loglevel=3
nodes=halinux03:6200,halinux04:6200
walletfile=/u01/app/oracle/product/10.2.0/opmn/conf/ssl.wlt/default
```

In this example, the `nodes` parameter points to the primary and standby hosts followed by the remote port for the ONS daemon running on that port.

The walletfile parameters point to the walletfile name. A wallet file is used by the Oracle Secure Sockets Layer (SSL) to store SSL certificates. If a wallet file is specified to ONS, it will use SSL when communicating with other ONS instances and require SSL certificate authentication from all ONS instances that try to connect to it. This means that if you want to turn on SSL for one ONS instance, then you must turn it on for all instances that are connected. Oracle recommends using SSL for all ONS communications. See Oracle Database JDBC Developer's Guide and Reference [12] for information about Fast Connection Failover and configuring ONS with SSL.

Once the configuration file has been created, you can start the ONS daemon on the middle tier or client nodes by issuing the following command:

```
$onsctl start
```

5. Create a trigger that fires on the system startup event that is used to relocate the database service `'salesJDBC'`, to a Data Guard standby database after it has transitioned to the primary role. An example of such a trigger is:

```
CREATE OR REPLACE TRIGGER manage_JDBCservice
after startup on  database
DECLARE
    role VARCHAR(30);
BEGIN
    SELECT DATABASE_ROLE INTO role FROM V$DATABASE;
    IF role = 'PRIMARY' THEN
       DBMS_SERVICE.START_SERVICE('salesJDBC');
END IF;
       END;
```

6. In order for JDBC clients to have complete notification following a Data Guard failover, you must create a second trigger enabled for the

DB_ROLE_CHANGE system event that calls a C program called the *FAN ONS Publisher* (a program provided by Oracle and described in Appendix A).

The trigger is required because primary host where the ONS daemons reside will no longer be available. By calling the FAN Publisher program based on a trigger enabled on the DB_ROLE_CHANGE system event, JDBC clients can be notified of the primary site failure and instructed to reconnect to the new primary database.

Configure the FAN ONS Publisher as follows:

- In $ORACLE_HOME/dbs, create a file named cfo{$ORACLE_SID}.ora. Refer to Appendix A, Formatting the ONS Publisher Configuration File, for an example of how to configure this file to provide the FAN ONS Publisher program the necessary information to create the event payloads.
- If you are on 10.2.0.3, the publisher executable (named "cfo") is already in the $ORACLE_HOME/bin directory. For versions prior to 10.2.0.3 refer to MetaLink note 405120.1. The publisher program publishes DOWN events for the failed primary database and UP events for the new primary. (See Appendix A for more details about the FAN ONS Publisher)
- Build a wrapper script around the Publisher program. This wrapper script should set the environment variables for the database and call the Publisher program with the correct arguments. Once the wrapper script has been created, set the permissions so that the Oracle user can execute the Publisher program.

  The following code example shows an example wrapper script (referred to as cfo.sh later on):

```
#!/bin/ksh
export TZ=PST8PDT
export ORACLE_SID=sales
export ORACLE_HOME=/u01/app/oracle/product/10.2.0
export LD_LIBRARY_PATH=/u01/app/oracle/product/10.2.0/lib
export PATH=/u01/app/oracle/product/10.2.0/bin:$PATH
/u01/app/oracle/product/10.2.0/bin/cfo r
```

  Note that when configuring the Publisher on a RAC cluster the ORA_CRS_HOME variable must be set so that the Publisher will utilize the ONS daemons in the Oracle CRS home instead of the ONS daemons in the Oracle database home.

- Create the trigger as follows:

  The trigger written around the `DB_ROLE_CHANGE` system event calls the external FAN ONS Publisher to notify JDBC clients of the new primary. Note that this is only required for JDBC clients, OCI and OLE DB clients are notified automatically using the database alert queue.

  On the primary database, create the role-change trigger. The following example shows how to create the trigger:

```
CREATE OR REPLACE TRIGGER ons_JDBCpublish
AFTER DB_ROLE_CHANGE ON
    DATABASE
BEGIN
      dbms_scheduler.create_job(
      job_name=>'publish_events',
      job_type=>'executable',
      job_action=>'/u01/oracle/product/10.2.0/db_1/bin/cfo.sh',
      enabled=>TRUE
      );
END;
```

If the Data Guard configuration is not using real-time apply, then archive the current redo log file to be sure that the changes are applied to the standby database:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

JDBC Client Configuration

The following configuration steps assume that the JDBC application can subscribe to FAN events. If the JDBC application doesn't meet the requirements for FAN then efficient failover can still be achieved using timeouts and application retries, (see Appendix C for more information).

Complete the following steps to configure JDBC clients for failover:

1. Refer to MetaLink Note 405120.1 for prerequisites required to configure automatic client failover for JDBC, OCI, and OLE DB clients using Oracle Database 10.2.0.2 or 10.2.0.3.
2. Configure JDBC clients to enable Fast Connection Failover.

   The client application must use implicit JDBC connection cache on its data source by setting the DataSource property `FastConnectionFailoverEnabled` to `true`. For example:

```
OracleDataSource ods = new OracleDataSource()
...
ods.setUser("hr");
ods.setPassword("hr");
ods.setConnectionCachingEnabled(True);
ods.setFastConnectionFailoverEnabled(True);
ods.setConnectionCacheName("MyCache");
ods.setConnectionCacheProperties(cp);
```

3. The client must set the oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR property. This property enables the JDBC client to quickly traverse an address_list in the event of a failure. For example, if the client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the SQLnetDef.TCP_CONNTIMEOUT_STR property after which the client attempts to connect to the next host in the address_list[1]. The behavior continues for each host in the address_list until a connection is made. Setting the property to a value of 3 seconds will suffice in most environments. It is important to note that the SQLnetDef.TCP_CONNTIMEOUT_STR property should be set on the data source and not on the implicit connection cache.

4. Configure JDBC clients to use a connect descriptor that includes an address list that includes the VIP address for each node in the cluster and connects to an existing service. Do not configure TAF with Fast Connection Failover for JDBC thick clients as TAF processing will interfere with FAN ONS processing. The following shows an example of the connect descriptor for two-node primary and standby RAC clusters:

```
SALESJDBC =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON1)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON2)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO1)(PORT = 1521))
     (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO2)(PORT = 1521))
     (LOAD_BALANCE = yes)
    )
   (CONNECT_DATA=
     (SERVICE_NAME=salesJDBC)
   )
  )
```

A JDBC thick (OCI) driver references the Oracle Net alias by calling the alias name. However, a JDBC thin driver does not use Oracle Net and so must include the complete connect descriptor in the URL. It is, however, possible to have the JDBC thin driver resolve the connect descriptor using LDAP, as in the following example:

```
ods.setURL("jdbc:oracle:thin:@ldap:

//halinux08:3061/sales,cn=oraclecontext,dc=halinux08,dc=
com");
```

[1] A good example of this occurs during the short time when a VIP is in transit, being failed over from one node to another

5. Configure a remote ONS subscription on the JDBC client so that an ONS daemon is not required on the client, as shown in the following example:

```
ods.setONSConfiguration("halinux03:6200,halinux04:6200");
```

The above remote ONS subscription should contain all hosts that have the potential to become a primary database. In addition, SSL should be used for all ONS communications. You can enable SSL for communications by using the following on the client:

```
ods.setONSConfiguration("nodes=halinux03:6200,
halinux04:6200 walletfile=/mydir/conf/Wallet");
```

For more information, refer to:

- Appendix B for information about troubleshooting JDBC Client failover.
- *Oracle Database JDBC Developer's Guide and Reference* [12] for information about Fast Connection Failover and configuring ONS.

CONCLUSION

A highly available architecture must achieve fast database and client failover.

- Database failover to a designated, synchronized standby database must occur quickly, and reliably in the event of loss of the primary database.
- Likewise, client failover must enable middle-tier applications (or any client program that connects directly to a database) to quickly and seamlessly fail over to an available database service when the primary database service is unavailable.

Client failover encompasses failure notification, previous connection cleanup, automatic reconnection, and possible query replay. Until Oracle Database 10*g* Release 2, automatic, fast, and transparent client failover (at the session level) has been difficult to achieve for all client types and for all failures.

Oracle Database 10*g* Release 2 client failover provides the capability to integrate automatic database failover with failover procedures at the middle tier to automatically redirect clients and applications to the new primary database at the standby location within seconds of failover – providing an end-to-end solution for achieving business continuity.

APPENDIX A: THE FAN ONS PUBLISHER PROGRAM

This appendix provides information to help you use the FAN ONS Publisher program. The FAN ONS Publisher program can publish ONS FAN notifications and is available with Oracle Database 10*g* Release 10.2.0.3 (for versions prior to 10.2.0.3 refer to MetaLink note 405120.1). The Publisher publishes `DOWN` events for the failed primary database and `UP` events for the new primary database.

Syntax and Arguments

The Publisher program must be located in ORACLE_HOME/bin directory and should have read and execute privileges set for the Oracle user only. The Publisher program has the following syntax:

```
$ cfo [t|r]
```

Table 2 describes the `t` and `r` syntax arguments, both of which are optional.

Table 2: ONS Publisher Program Syntax Arguments

Argument	Description
`t|r`	`t`—Runs the Publisher program in a test mode. It does essentially everything except publish the event. Use the `t` argument during set up of the environment when you do not want to actually publish the `DOWN` and `UP` events. This is the default mode. `r`—Calls the Publisher program in a run mode. The actual events are published provided the environment has been configured correctly.

Formatting the ONS Publisher Configuration File

For security purposes, the ONS configuration file must exist in ORACLE_HOME/dbs and should have read and write privileges assigned to the Oracle user only. Use the following example template to format the contents of your Publisher program configuration file:

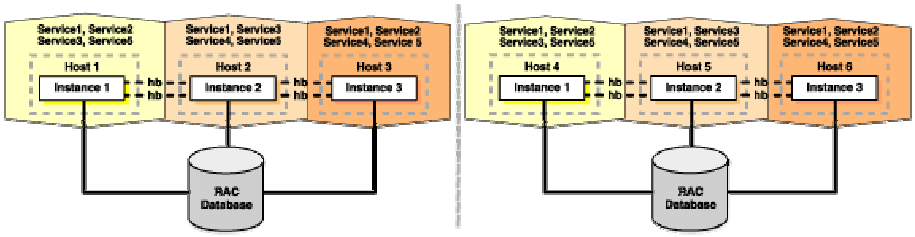
```
<DUN1> peer=<DUN2>
<DUN1> srv=<srv1> location=<host1>,<inst1>:<host2>,<inst2>
<DUN1> srv=<srv2> location=<host2>,<inst2>:<host3>,<inst3>
<DUN1> srv=<srv3> location=<host1>,<inst1>:<host3>,<inst3>

<DUN2> peer=<DUN1>
<DUN2> srv=<srv1> location=<host4>,<inst1>:<host5>,<inst2>
<DUN2> srv=<srv2> location=<host5>,<inst2>:<host6>,<inst3>
<DUN2> srv=<srv3> location=<host4>,<inst1>:<host6>,<inst3>
```

APPENDIX A - CONTINUED

In the above configuration file example the following abbreviations are used:

DUN = db_unique_name
srv = service
inst = instance

The above example template configuration file can be expressed graphically as follows:



Assume that you have a RAC configuration set up with the information shown in Table 3:

Table 3: Sample Configuration Information

Database	Host	Instance Names	DB_UNIQUE_NAME
Primary	halinux03	Sales1	Sales_PRD
Primary	halinux04	Sales2	Sales_PRD
Standby	stacg33	Sales	Sales_DR

The following example show the FAN ONS configuration file configured using the values shown in Table 3:

```
Sales_PRD peer=Sales_DR
Sales_DR peer=Sales_PRD
Sales_PRD service=sales location=halinux03,Sales1:halinux04,Sales2
Sales_DR service=sales location=stacg33,Sales
```

The configuration file will be identical on all hosts (primary and standby) when configured as described in this appendix.

The ONS Publisher log file is located in `$ORACLE_HOME/rdbms/log` with the filename `cfo{$ORACLE_SID}.log`. You can set the environment variable `CFO_DEBUG=TRUE` to also include the contents of the events generated and published in the log file.

APPENDIX B: TROUBLESHOOTING CLIENT FAILOVER

This appendix provides general troubleshooting guidelines for client failover.

OCI Clients

1. Verify that all client configuration requirements have been met as described in this paper.

2. Using Oracle Net tracing,- verify that the client has received the HA event. Oracle Net tracing can be enabled by placing the following parameters in the `SQLNET.ORA` file:

   ```
   trace_level_client=16
   trace_directory_client=<any valid path>
   ```

   After enabling Oracle Net tracing, attempt to perform the failover scenario again. If the client still does not perform a failover, then examine the resulting Oracle Net trace file and verify that the client received the event. The event packet should be similar to the following:

   ```
   nsprecv: 00 1B 22 53 59 53 22 2E  |.."SYS".|
   nsprecv: 22 41 4C 45 52 54 5F 51  |"ALERT_Q|
   nsprecv: 55 45 22 3A 22 48 41 45  |UE":"HAE|
   nsprecv: 5F 53 55 42 22 17 00 00  |_SUB"...|
   nsprecv: 00 01 01 01 00 00 00 4D  |.......M|
   nsprecv: 32 7C 31 2E 30 7C 33 7C  |2|1.0|3||
   nsprecv: 31 7C 32 30 30 35 2D 30  |1|2005-0|
   nsprecv: 34 2D 32 39 20 32 32 3A  |4-29.22:|
   nsprecv: 31 34 3A 32 38 2E 30 30  |14:28.00|
   nsprecv: 30 30 30 30 30 30 30 20  |0000000.|
   nsprecv: 2B 30 30 3A 30 30 7C 73  |+00:00|s|
   nsprecv: 74 61 6A 7A 31 32 7C 48  |tajz12|H|
   nsprecv: 41 4C 58 7C 7C 48 41 4C  |ALX||HAL|
   nsprecv: 58 31 32 7C 68 61 73 65  |X12|hase|
   nsprecv: 72 76 31 7C 00 1A 00 00  |rv1|....|
   nsprecv: 00 04 00 00 00 01 17 00  |........|
   nsprecv: 00 00 11 22 53 59 53 22  |..."SYS"|
   nsprecv: 2E 22 41 4C 45 52 54 5F  |."ALERT_|
   nsprecv: 51 55 45 22 17 00 00 00  |QUE"....|
   nsprecv: 10 F5 EA 99 89 43 51 A3  |.....CQ.|
   nsprecv: A1 E0 30 57 8C ED 19 1C  |..0W....|
   nsprecv: 4D 17 00 00 00 07 48 41  |M.....HA|
   nsprecv: 45 5F 53 55 42 17 00 00  |E_SUB...|
   ```

3. If the client is receiving the event but does not reconnect, then verify that the Oracle Net alias used for connection is pointing to the correct host. To do this:

   a. Enable Oracle Net tracing as described in step 2

   b. Test with a new connection using the Oracle Net alias

APPENDIX B - CONTINUED

4. Verify that the event was consumed and posted by the EMON process. Prior to an event being consumed and posted by EMON, it will be placed into the wri$_alert_outstanding table. Once the EMON process has posted the events to the clients, the event is moved to the wri$_alert_history table. To verify that the event is not remaining in the outstanding events, issue the following query:

```
SQL> SELECT * from  wri$_alert_outstanding;
```

5. Verify that the Data Guard broker posted the event to the new primary database. The following message, which is posted in the Data Guard broker log, indicates that the event was posted:

```
DMON: Posting DB_DOWN alert
```

JDBC Clients

1. Verify that Fast Connection Failover is enabled and all requirements described in this paper have been met.
2. Verify that the JDBC client is receiving the appropriate service `DOWN` and `UP` events. This can be accomplished by enabling appropriate JDBC tracing, as described in the following steps:
   a. Ensure you are using JDK Release 1.4
   b. Use debug jar `ojdbc14_g.jar` instead of `ojdbc14.jar`. By placing `ojdbc14_g.jar` in your `CLASSPATH` you can enable additional diagnostic messages.
   c. Create a properties file, with the following contents:

```
============
handlers= java.util.logging.ConsoleHandler
.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.Simpleformatter
.
# 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.driver.level = FINEST
==============
```

APPENDIX B – CONTINUED

d. Set the following parameters when you start testing:

```
...-Doracle.jdbc.Trace=true
-Djava.util.logging.config.file=<properties file location>
...
```

When tracing is enabled, messages are printed to standard output when services `UP` and `DOWN` events are received.

3. If the client is receiving the event but does not reconnect, then verify that the connect descriptor used for connection points to the correct host.

4. Verify that the ONS Publisher program has posted service `DOWN` and `UP` events. The ONS Publisher log file is located in `$ORACLE_HOME/rdbms/log` with the filename `cfo{$ORACLE_SID}.log`.

   You can set the environment variable `CFO_DEBUG=TRUE` to also include the contents of the events generated and published in the log file.

APPENDIX C: CLIENT FAILOVER FOR APPLICATIONS THAT DO NOT SUPPORT FAN

The following sections describe how to configure applications that cannot support FAN events. Even when FAN events cannot be used, applications can still be configured for efficient failover by using timeouts and application retries

OCI APPLICATIONS

Database Configuration for OCI Clients

1. If you are configuring a RAC database, Oracle recommends using the Oracle Enterprise Manager Cluster Managed Services Page to create database services that client applications will use to connect to the primary database. For complete instructions on creating database services, please see the chapter on Workload Management in the *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide* [7].

2. If you have created the database service as instructed in step 2, above, use the `DBMS_SERVICE` package to modify the service to enable high-availability notification to be sent through Advanced Queuing (AQ) by setting the `AQ_HA_NOTIFICATIONS` attribute to `TRUE`. To configure server-side TAF settings, set the `FAILOVER` attributes, as shown in the following example:

```
exec DBMS_SERVICE.MODIFY_SERVICE(
 service_name => 'salesOCI',
        failover_method => 'BASIC',
        failover_type => 'SELECT',
        failover_retries => 180,
        failover_delay => 1);
```

Note: Refer to the *Oracle Database PL/SQL Packages and Types Reference* [10] for more information about the `DBMS_SERVICE` package.

3. If you are creating database services for a single instance (non-RAC) database, or if you are creating services for a RAC database and are not using Enterprise Manager, use the CREATE_SERVICE subprogram illustrated in the following example to both create the database service and enable high-availability notification and configure server-side TAF settings:

```
exec DBMS_SERVICE.CREATE_SERVICE (
        service_name => 'salesOCI',
        network_name => 'salesOCI',
        failover_method => 'BASIC',
        failover_type => 'SELECT',
        failover_retries => 180,
        failover_delay => 1);
```

APPENDIX C – CONTINUED

4. Create a trigger that fires on the system startup event to relocate the database service `'salesOCI'`, in the above example, to a Data Guard standby database (RAC or non-RAC) after it has transitioned to the primary role.

```
CREATE OR REPLACE TRIGGER manage_OCIservice
after startup on  database
DECLARE
    role VARCHAR(30);
BEGIN
    SELECT DATABASE_ROLE INTO role FROM V$DATABASE;
    IF role = 'PRIMARY' THEN
       DBMS_SERVICE.START_SERVICE('salesOCI');
END IF;
    END;
```

If the Data Guard configuration is not using real-time apply, then archive the current redo log file to be sure that the changes are applied to the standby database:

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

OCI Client Configuration

1. Create an Oracle Net service name that includes all primary and standby hosts in the ADDRESS_LIST. The following example

```
SALESOCI =
  (DESCRIPTION =
     (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON1)(PORT = 1521))
        (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON2)(PORT = 1521))
        (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO1)(PORT = 1521))
        (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO2)(PORT = 1521))
       (LOAD_BALANCE = yes)
    )
    (CONNECT_DATA=
       (SERVICE_NAME=salesOCI)
    )
  )
```

APPENDIX C – CONTINUED

2. In the client side SQLNET.ORA file, set the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter. This parameter enables clients to quickly traverse an address_list in the event of a failure. For example, if a client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the SQLNET.OUTBOUND_CONNECT_TIMEOUT parameter, after which the client attempts to connect to the next host in the address_list. This behavior continues for each host in the address_list until a connection is made. Setting the parameter to a value of 3 seconds will suffice in most environments.

Application and Operating System Configuration

1. On the hosts that run the application layer configure the operating system for efficient TCP timeouts. The OS TCP timeouts should be set to the amount of time it takes for the database layer to failover and the database services to be started. Consult your operating system manuals for how to properly configure TCP timeout.
2. Application retries can be automated using TAF configured in the above steps. If the application cannot use TAF then the application must be configured with reconnection logic in the event of an exception. For example, when a session from the connection pool receives any exception which results in a disconnect (such as an ORA-3113 error) the application should automatically attempt to reconnect that session. The reconnection attempts should be configured such that they will continue for the length of time that it takes to failover the database layer and bring the application services online.

JDBC APPLICATIONS

Database Configuration for JDBC Clients

1. If you are configuring a RAC database, Oracle recommends using Oracle Enterprise Manager Cluster Managed Services Page to create database services that client applications will use to connect to the primary database. If you are creating services for a RAC database and are not using Enterprise Manager, use the CREATE_SERVICE subprogram illustrated in the following example to create the database service `'salesJDBC'` for JDBC clients:

```
exec DBMS_SERVICE.CREATE_SERVICE (
       service_name => 'salesJDBC',
       network_name => 'salesJDBC',
```

APPENDIX C – CONTINUED

2. Create a trigger that fires on the system startup event that is used to relocate the database service `'salesJDBC'`, to a Data Guard standby database after it has transitioned to the primary role. An example of such a trigger is:

```
CREATE OR REPLACE TRIGGER manage_JDBCservice
after startup on  database
DECLARE
    role VARCHAR(30);
BEGIN
    SELECT DATABASE_ROLE INTO role FROM V$DATABASE;
    IF role = 'PRIMARY' THEN
       DBMS_SERVICE.START_SERVICE('salesJDBC');
    END IF;
       END;
```

JDBC Client Configuration

Complete the following steps to configure JDBC clients for failover:

1. Refer to MetaLink Note 405120.1 for prerequisites required to configure automatic client failover for JDBC clients using Oracle Database 10.2.0.2 or 10.2.0.3.
2. The client must set the oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR property. This property enables the JDBC client to quickly traverse an address_list in the event of a failure. For example, if the client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the SQLnetDef.TCP_CONNTIMEOUT_STR property after which the client attempts to connect to the next host in the address_list[i]. The behavior continues for each host in the address_list until a connection is made. Setting the property to a value of 3 seconds will suffice in most environments. It is important to note that the SQLnetDef.TCP_CONNTIMEOUT_STR property should be set on the data source and not on the implicit connection cache.
3. Configure JDBC clients to use a connect descriptor that includes an address list that includes the VIP address for each node in the primary and standby clusters and connects to an existing service. Do not configure TAF. The following shows an example of the connect descriptor for two-node primary and standby RAC clusters:

APPENDIX C – CONTINUED

```
 SALESJDBC =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON1)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = BOSTON2)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO1)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = CHICAGO2)(PORT =
1521))
      (LOAD_BALANCE = yes)
  )
    (CONNECT_DATA=
      (SERVICE_NAME=salesJDBC)
)
   )
```

A JDBC thick (OCI) driver references the Oracle Net alias by calling the alias name. However, a JDBC thin driver does not use Oracle Net and so must include the complete connect descriptor in the URL. It is, however, possible to have the JDBC thin driver resolve the connect descriptor using LDAP, as in the following example:

```
ods.setURL("jdbc:oracle:thin:@ldap:

//halinux08:3061/sales,cn=oraclecontext,dc=halinux08,dc=
com");
```

Application and Operating System Configuration

1. On the hosts that run the application layer configure the operating system for efficient TCP timeouts. The OS TCP timeouts should be set to the amount of time it takes for the database layer to failover and the database services to be started. Consult your operating system manuals for how to properly configure TCP timeout.
2. Within the application configure reconnection logic in the event of an exception. For example, when a session from the connection pool receives any exception which results in a disconnect (such as an ORA-3113 error) the application should automatically attempt to reconnect that session. The reconnection attempts should be configured such that they will continue for the length of time that it takes to failover the database layer and bring the application services online.

REFERENCES

1. "Switchover and Failover Best Practices: Oracle Data Guard 10g Release 2"
   http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_FastStartFailoverBestPractices.pdf
2. "Fast-Start Failover Best Practices: Oracle Data Guard 10g Release 2"
   http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_FastStartFailoverBestPractices.pdf
3. Oracle Maximum Availability Architecture
   http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm
4. Oracle Database High Availability Best Practices (Part #B25159)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b25159
5. Workload Management with Oracle Real Application Clusters
   http://www.oracle.com/technology/products/database/clustering/pdf/twpracwkldmgmt.pdf
6. Oracle Data Guard
   http://www.oracle.com/technology/deploy/availability/htdocs/DataGuardOverview.html
7. Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide (Part #14197)
   http://download-west.oracle.com/docs/cd/B19306_01/rac.102/b14197/toc.htm
8. Workload Management with Oracle Real Application Clusters 10g (Provides a detailed explanation of the implementation of Services, FAN and Fast Connection Failover in a RAC environment.):
   http://www.oracle.com/technology/products/database/clustering/pdf/twpracwkldmgmt.pdf
9. Oracle Data Guard Broker (Part #B14230)
   http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14230
10. Oracle Database PL/SQL Packages and Types Reference (Part #B14261)
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14261
11. Oracle Call Interface Programmer's Guide (Part #B14250)
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14250
12. Oracle Database JDBC Developer's Guide and Reference (Part #B14355)
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14355
13. Oracle Data Guard Concepts and Administration (Part #B14239)
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14239
14. Oracle Data Provider for .NET Developer's Guide (Part #B14307)
    http://otn.oracle.com/pls/db102/db102.to_toc?partno=b14307
15. *Oracle Database Advanced Security Administrator's Guide*
    http://download-west.oracle.com/docs/cd/B19306_01/network.102/b14268/asossl.htm - sthref488

Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 10*g* Release 2
June 2009
Author: Michael T. Smith
Contributing Authors: Barb Lundhild, Lawrence To, Vivian Schupmann, Larry Carpenter, Joseph Meeks, Ashish Ray

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