
Connecting to an Oracle Real Application Clusters (RAC) System

Connecting to an Oracle RAC system is similar to connecting to a single instance of an Oracle database. When connecting to a single Oracle database instance, you specify either the SID or ServiceName of the instance to which you want to connect in the connection string. For example, the following connection string establishes a connection to the database instance Accting1:

```
"host=server1;Port=1521;ServiceName=Accting1"
```

In a RAC environment, multiple Oracle instances share the same physical data. In addition to the SID or ServiceName for each Oracle instance in the Oracle RAC system, a ServiceName exists for the entire Oracle RAC system. When an application uses the Oracle RAC system's ServiceName, the Oracle RAC system appears to be a single Oracle instance to the application. For example, the following connection string establishes a connection to an Oracle instance in the Oracle RAC system named Accounting:

```
"host=server1;Port=1521;ServiceName=Accounting"
```

The specific instance that is connected to is determined by a number of factors, including which instances are available and the load on those instances. Typically, the application does not need to know the instance to which it is connected.

Retrieving Connection Information from a tnsnames.ora File

The DataDirect Connect *for* ADO.NET Oracle data provider also supports retrieving specific connection information, including connection failover and client load balancing instructions, from a tnsnames.ora file. The type of information the DataDirect Connect *for* ADO.NET Oracle data provider allows you to retrieve from a tnsnames.ora file includes:

- Oracle server name and port
- Oracle System Identifier (SID) or Oracle service name
- Server process type (shared or dedicated)
- Connection failover and client load balancing instructions

The DataDirect Connect *for* ADO.NET Oracle data provider allows you to specify multiple tnsnames.ora file locations in the connection string. Allowing for multiple tnsnames.ora files provides an extra layer of failover capability when the machine on which the tnsnames.ora file resides is unavailable.

The following example shows the syntax when the TNSNames File connection string option specifies two tnsnames.ora files:

```
TNSNames File= (F:\server2\oracle\tnsnames.ora,
C:\oracle\product\10.1.0\db_1\network\admin\tnsnames.ora)
```

tnsnames.ora File Example

In a tnsnames.ora file, connection information for Oracle services is associated with a net service name. The following code example shows connection information in a tnsnames.ora file configured for an Oracle RAC system identified by the net service name entry, ARMSTRONG.ACCT.

```
ARMSTRONG.ACCT =
  (DESCRIPTION =
    (ADDRESS_LIST=
      (ADDRESS= (PROTOCOL = TCP)(HOST = server1)(PORT = 1521))
      (ADDRESS= (PROTOCOL = TCP)(HOST = server2)(PORT = 1521))
      (ADDRESS= (PROTOCOL = TCP)(HOST = server3)(PORT = 1521))
      (FAILOVER = on)
      (LOAD_BALANCE = on)
    )
    (CONNECT_DATA=
      (SERVICE_NAME = acct.us.yourcompany.com)
    )
  )
```

If the DataDirect Connect *for* ADO.NET Oracle data provider referenced the net service name entry ARMSTRONG.ACCT as shown in this example, the data provider would connect to the Oracle RAC system identified by the service name acct.us.yourcompany.com

(SERVICE_NAME=acct.us.yourcompany.com). In addition, the data provider would enable connection failover (FAILOVER=on) and client load balancing (LOAD_BALANCE=on) for all connections to that system.

Alternatively, DataDirect Connect *for* ADO.NET provides a way to enable connection failover and client load balancing through options specified in the data provider connection string. For example, the connection string in the following code example enables both of these features:

```
DbProviderFactory Factory = DbProviderFactories.GetFactory("DDTek.Oracle");
DbConnection Conn = Factory.CreateConnection();
Conn.ConnectionString =
  "Host=server1;Port=1521;Service Name=ORCL;" +
  "Alternate Servers=Host=server2;Port=1521;Service Name=TEST, "+
  "Host=255.210.11.25;Port=1600;Service Name=TEST2"; "+
  "Load Balancing=true";
Conn1.Open();
```

Failover

Oracle RAC systems provide two methods of failover to provide reliable access to data:

- *Connection failover.* If a connection failure occurs *at connect time*, the application can fail over the connection to another active node in the cluster. Connection failover ensures that an open route to your data is always available, even when server downtime occurs.
- *Transparent Application Failover (TAF).* If a communication link failure occurs after a connection is established, the connection fails over to another active node. Any disrupted transactions are rolled back, and session properties and server-side program variables are lost. In some cases, if the statement executing at the time of the failover is a *Select* statement, that statement may be automatically re-executed on the new connection with the cursor positioned on the row on which it was positioned prior to the failover.

Both connection failover and TAF provide a *connection retry* feature that allows a connection to be retried automatically until a connection with another RAC node is successfully re-established.

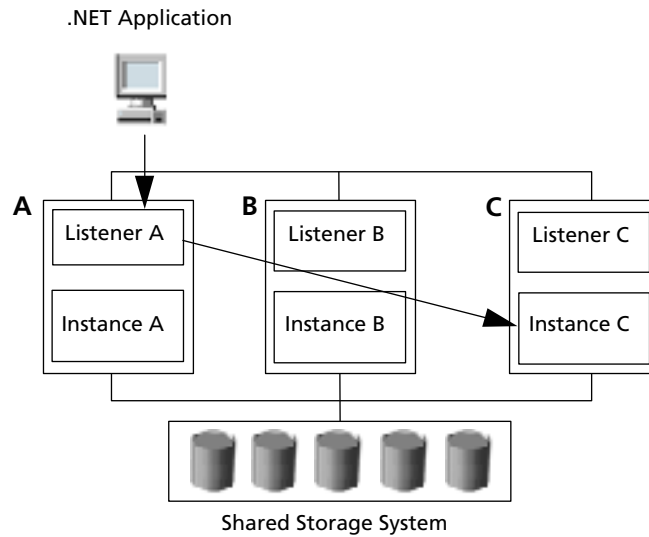
The primary difference between connection failover and TAF is that the former method provides protection for connections at connect time and the latter method provides protection for connections that have already been established. Also, because the state of the transaction must be stored at all times, TAF requires more processing overhead than connection failover.

Connection Failover

Enabling connection failover allows a data provider to connect to another node if a connection attempt on one node fails. When an application requests a connection to an Oracle database server through the data provider, the data provider does not connect to the database server directly. Instead, the data provider sends a connection request to a listener process, which forwards the request to the appropriate Oracle database instance. In an Oracle RAC system, each active Oracle database instance in the RAC system registers with each listener configured for the Oracle RAC.

For example, if we look at the Oracle RAC Nodes A, B, and C in Figure 2, Instances A, B, and C are registered with Listeners A, B, and C. If the service name in the connection request specifies the RAC system database name, the requested listener selects one of the registered instances to forward the connection request to, based on the load each of the instances is experiencing. For example, if Instances A and B are operating under a heavy load, a connection request to Listener A results in the connection being forwarded to Instance C.

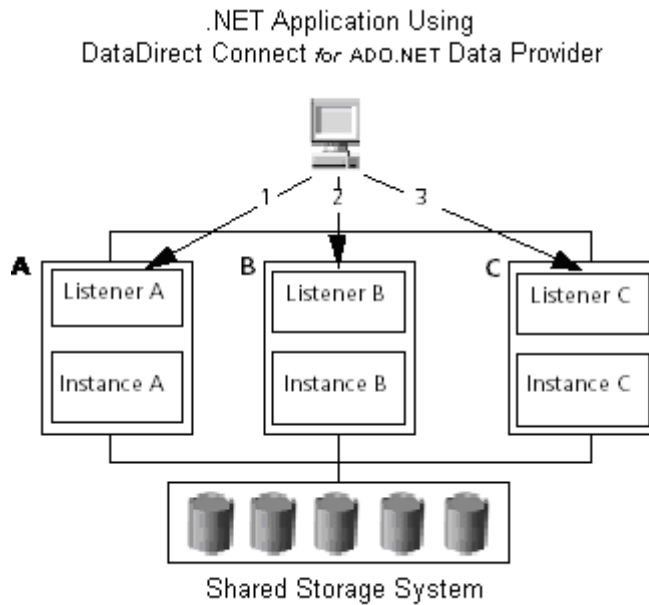
Figure 2: Oracle RAC with Connection Failover



Because the requested listener selects from a set of active instances in the RAC system to forward connection requests to, it should not route the connection request to an instance that is not running. You may think that connection failover is not needed in an Oracle RAC system; however, if the requested listener is down or the timing of an instance going down is such that the requested listener is not yet aware that an instance is down, the connection request can fail.

The connection failover feature provided by the DataDirect Connect for ADO.NET Oracle data provider handles the case where the requested listener or the server selected by the listener is down by allowing you to specify multiple listeners to which to connect. For example, as shown in Figure 3, if Listener A is down, the DataDirect Connect for ADO.NET Oracle data provider can be configured to try Listener B, and then Listener C.

Figure 3: Oracle RAC with Connection Failover



Connection failover provides protection for new connections only and does not preserve states for transactions or queries, so your application needs to provide failure recovery for transactions and queries.

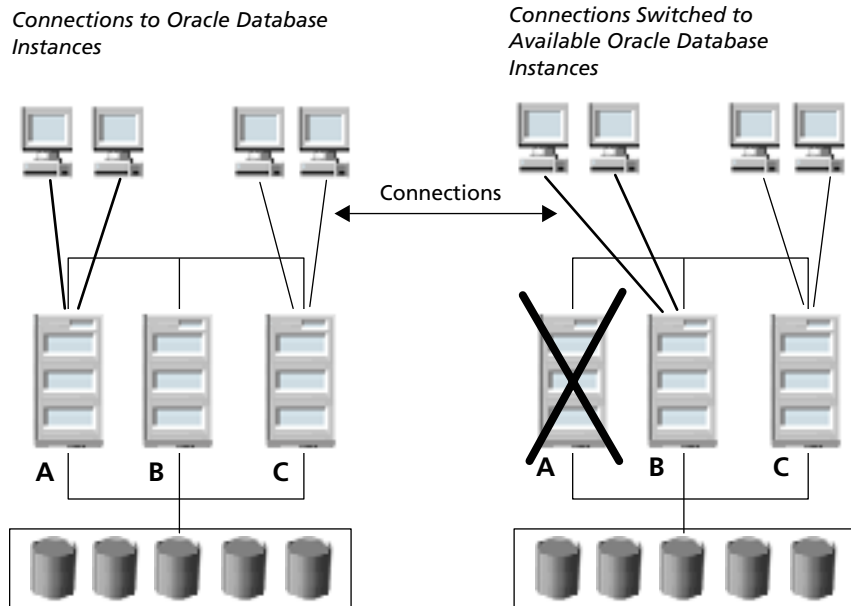
This feature is configured through the data provider's Alternate Servers connection string option or through the tnsnames.ora file. The following code example shows a connection string that enables connection failover for the DataDirect Connect for ADO.NET Oracle data provider:

```
DbProviderFactory Factory = DbProviderFactories.GetFactory("DDTek.Oracle");
DbConnection Conn1 = Factory.CreateConnection();
Conn1.ConnectionString =
    "Host=Accounting;Port=1521;User ID=scott;Password=tiger; " +
    "Service Name=ORCL;Min Pool Size=50";Alternate Servers="Host=server2;"+
    "Port=1521;Service Name=TEST, Host=255.210.11.25;Port=1600;"
    "Service Name=TEST2";Load Balancing=true";
Conn1.Open();
```

Transparent Application Failover (TAF)

With TAF, if a communication link failure occurs after a connection is established, the connection is moved to another active Oracle RAC node in the cluster without the application having to re-establish the connection. For example, suppose you have the Oracle RAC environment shown in Figure 4 with multiple connections to Oracle RAC nodes: A, B, and C. As shown in the first case, connections are distributed among the nodes in an Oracle RAC system.

Figure 4: Transparent Application Failover (TAF)



When a communication link failure occurs between an Oracle node and the application as shown in the second case, the data provider automatically switches the connection to another available node.

When a user session fails over to an alternate RAC node, the following items are not persisted to the failover node and must be reinitialized by the application:

- In-use stored procedures
- Application changes to session state
- In-flight write transactions (local transactions doing database updates)
- Global transactions

Although Oracle documentation refers to this functionality as transparent, the preceding list shows that it is not completely transparent to an application. The application programmer must include code to handle the necessary clean-up caused by rolled back transactions or lost session states. Because of these restrictions, the situations where application failover is beneficial when implemented by the data provider are limited.

Applications can perform a failover using the DataDirect Connect for ADO.NET Oracle data provider by performing the following steps:

1. Catch the communication error exception generated by the data provider.
2. Take the necessary steps to deal with current transactions that were rolled back.
3. Re-establish the connection to the server.

4. Re-initialize the session state.
5. Re-run any transaction that was rolled back.

To make it easy for applications to detect when the connection with the server is lost, all communication error exceptions thrown by the DataDirect Connect *for* ADO.NET data providers have a SQL state that begins with 08.

DataDirect is currently evaluating ways to enhance the failover functionality in the DataDirect Connect *for* ADO.NET data providers for a future release.

Connection Retry

DataDirect Connect *for* ADO.NET data providers support a connection retry feature that works with connection failover. You can customize the data provider to attempt to reconnect a specific number of times and at a specific time interval. Connection retry can be used in environments that have only one server or can be used as a complementary feature in connection failover scenarios with multiple servers.

Connection retry can be an important strategy in recovering from failures that can bring down an Oracle RAC system. For example, suppose you have a power failure scenario in which both the client and the Oracle RAC system go down. When the power is restored and all computers are restarted, the client may be ready to attempt a connection before an Oracle RAC system has completed its startup routines. If connection retry is enabled, the client application continues to retry the connection until a connection is successfully accepted by a node in the Oracle RAC system.

For example, the connection string in the following code fragment instructs the data provider to cycle through the list of servers (the primary server and alternate servers) up to 10 more times if the data provider was unable to establish a connection to any of the servers in the list during the initial pass. The data provider waits 10 seconds before it cycles through the list of servers again.

```
DbProviderFactory Factory = DbProviderFactories.GetFactory("DDTek.Oracle");
DbConnection Conn1 = Factory.CreateConnection();
Conn1.ConnectionString =
    "Host=server1;Port=1521;Service Name=ORCL; Min Pool Size=50"; " +
    "Alternate Servers="Host=server2;Port=1521;Service Name=TEST," " +
    "Host=255.210.11.25;Port=1600;Service Name=TEST2"; " " +
    "Load Balancing=true;Connection Retry Count=10;
    "Connection Retry Delay=10;";
Conn1.Open();
```

In the following example, the connection string instructs the data provider to attempt to connect to the primary server up to 10 more times if the data provider was unable to establish a connection during the initial pass. The data provider waits 10 seconds before attempting to connect again:

```
DbProviderFactory Factory = DbProviderFactories.GetFactory("DDTek.Oracle");
DbConnection Conn2 = Factory.CreateConnection();
Conn2.ConnectionString =
    "Host=server1;Port=1521;Service Name=ORCL;" +
    "Connection Retry Count=10;Connection Retry Delay=10;";
```

The connection retry feature also applies when the TNSNames File option of DataDirect Connect for ADO.NET Oracle data provider includes one or more tnsnames.ora files. The following connection string for the DataDirect Connect for ADO.NET Oracle data provider:

```
DbProviderFactory Factory = DbProviderFactories.GetFactory("DDTek.Oracle");
DbConnection Conn3 = Factory.CreateConnection();
Conn3.ConnectionString =
    "Data Source=Accounting Server;User ID=Scott;Password=Tiger;" +
    "Alternate Servers="Data Source=AccountingBackupServer," "+
    "Data Source=AccountingBackupServer2);Connection Retry Count=10;"+
    "Connection Retry Delay=10;TNSNames File= "
    "(C:\\oracle\\product\\10.1.0\\db_1\\network\\admin\\tnsnames.ora," +
    "F:\\server2\\oracle\\tnsnames.ora)";
```

instructs the data provider to attempt to cycle through the list of servers (the primary server and alternate servers) up to 10 more times if the data provider was unable to establish a connection during the initial pass. The data provider also cycles through the list of tnsnames.ora files up to 10 more times if it was unable to retrieve connection information from the tnsnames.ora file during the initial pass. The data provider waits 10 seconds before it cycles through the lists of servers and tnsnames.ora files again.

Load Balancing

Oracle RAC systems provide two types of load balancing for automatic workload management:

- Server load balancing distributes processing workload among Oracle RAC nodes.
- Client load balancing distributes new connections among Oracle RAC nodes so that no one server is overwhelmed with connection requests. For example, when a connection fails over to another node because of hardware failure, client load balancing ensures that the redirected connection requests are distributed among the other nodes in the RAC.

The primary difference between these two methods is that the former method distributes processing and the latter method distributes connection attempts.

Server Load Balancing

With Oracle9i RAC systems, a listener service provides automatic load balancing across nodes. The query optimizer determines the optimal distribution of workload across the nodes in the RAC based on the number of processors and current load.

Oracle 10g also provides load-balancing options that allow the database administrator to configure rules for load balancing based on application requirements and Service Level Agreements (SLAs). For example, rules can be defined so that when Oracle 10g instances running critical services fail, the workload is automatically shifted to instances running less critical workloads. Or, rules can be defined so that Accounts Receivable services are given priority over Order Entry services.

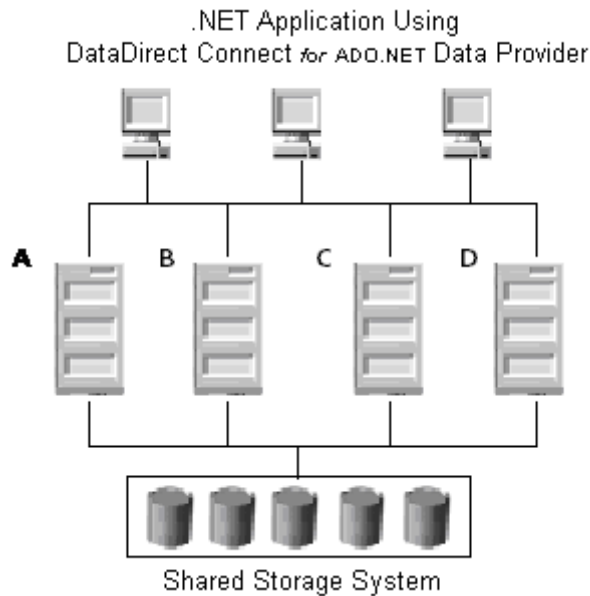
The DataDirect Connect *for* ADO.NET Oracle data provider can transparently take advantage of server load balancing provided by an Oracle RAC without any changes to the application. If you do not want to use server load balancing, you can bypass it by connecting to the service name that identifies a particular RAC node.

Client Load Balancing

Client load balancing helps distribute new connections in your environment so that no one server is overwhelmed with connection requests. When client load balancing is enabled, connection attempts are made randomly among RAC nodes. You can enable client-side load balancing for DataDirect Connect *for* ADO.NET connections through either the data provider connection string using the Load Balancing option, or through the LOAD_BALANCE connect descriptor parameter in the tnsnames.ora file.

Suppose you have the Oracle RAC environment shown in Figure 5 with multiple Oracle RAC nodes, A, B, C, and D. Without client load balancing enabled, connection attempts may be front-loaded, meaning that most connection attempts would try Node A first, then Node B, and so on until a connection attempt is successful. This creates a situation where Node A and Node B can become overloaded with connection requests.

Figure 5: Client Load Balancing



With client load balancing enabled, the data provider randomly selects the order of the connection attempts to nodes throughout the Oracle RAC system. For example, Node B may be tried first, followed by Nodes D, C, and A. Subsequent connection retry attempts will continue to use this order. Using a randomly determined order makes it less likely that any one node in the Oracle RAC system will be so overwhelmed with connection requests that it may start refusing connections.

For example, the following connection string enables client load balancing for the DataDirect Connect for ADO.NET Oracle data provider:

```
DbProviderFactory Factory = DbProviderFactories.GetFactory("DDTek.Oracle");
DbConnection Conn = Factory.CreateConnection();
Conn.ConnectionString =
    "Host=server1;Port=1521;Service Name=TEST;" +
    "Alternate Servers=Host=server2; Port = 1600; " +
    "Service Name=TEST,Host=255.210.11.25; Port = 1521;" +
    "Service Name= RCL";Load Balancing=true;";"
```


FOR MORE INFORMATION

800-876-3101

Worldwide Sales

Belgium (French).....0800 12 045
Belgium (Dutch).....0800 12 046
France.....0800 911 454
Germany0800 181 78 76
Japan0120.20.9613
Netherlands.....0800 022 0524
United Kingdom.....0800 169 19 07
United States.....800 876 3101

Copyright © 2006 DataDirect Technologies Corp. All rights reserved. DataDirect Connect is a registered trademark of DataDirect Technologies Corp. in the United States and other countries. DataDirect XQuery is a trademark of DataDirect Technologies Corp. in the U.S. and other countries. Java and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Other company or product names mentioned herein may be trademarks or registered trademarks of their respective companies.



DataDirect Technologies is focused on standards-based data connectivity, enabling software developers to quickly develop and deploy business applications across all major databases and platforms. DataDirect Technologies offers the most comprehensive, proven line of data connectivity components available anywhere. Developers worldwide at more than 250 leading independent software vendors and thousands of corporate IT departments rely on DataDirect[®] products to connect their applications to an unparalleled range of data sources using standards-based interfaces such as ODBC, JDBC[™] and ADO.NET. Developers also depend on DataDirect to radically simplify complex data integration projects using XML products based on the emerging XQuery and XQJ standards. DataDirect Technologies is an operating company of Progress Software Corporation (Nasdaq: PRGS), a US\$300+ million global software industry leader. Headquartered in Bedford, Mass., DataDirect Technologies can be reached on the Web at <http://www.datadirect.com> or by phone at +1-800-876-3101.