Managing Database Workload Using Services

Using workload management, you can distribute the workload across database instances to achieve optimal database and cluster performance for users and applications. This chapter contains the following sections:

About Workload Management

Applications using a clustered database generally want to load balance their workload across the cluster. Oracle Real Application Clusters (Oracle RAC) includes a highly available (HA) application framework that provides the necessary service and integration points between Oracle RAC and custom enterprise applications.

You can deploy Oracle RAC and single-instance Oracle database environments to use workload management features in many different ways. Depending on the number of nodes and your environment's complexity and objectives, your choices for the optimal workload management and high availability configuration depend on a variety of considerations, as described in this chapter.

To implement workload management for an Oracle RAC database, you can use a number of different features. This section contains the following topics:

About Oracle Services

Oracle Database 10*g* introduced an automatic workload management facility, called database services. Database services (services) are logical abstractions for managing workloads in Oracle Database. Services divide workloads into mutually disjoint

groupings. Each service represents a workload with common attributes, service-level thresholds, and priorities.

A single service can represent an application, multiple applications or a subset of a single application. For example, the Oracle E-Business suite defines a service for each responsibility, such as general ledger, accounts receivable, order entry, and so on. A single service can be associated with one or more instances of an Oracle RAC database, and a single instance can support multiple services.

Note:
A database service can only be offered on a single network.

Services provide the following benefits:

- A single entity for managing applications that compete for the same resources
- Allow each workload to be managed as a unit
- Hide the complexity of the cluster from the client

To manage workloads, you can define services that you assign to a particular application or to a subset of an application's operations. You can also use services to manage the workload for different types of work. For example, online users can use one service while batch processing can use a different service and reporting can use yet another service type.

Traditionally an Oracle database provided a single service and all users connected to the same service. A database always has this default database service that is the database name. This service cannot be modified and always enables you to connect to the database.

Note:
Do not use the default database service for application workloads. Create at least one service as described in "Creating Services".

When a user or application connects to a database, Oracle recommends that you use a service for the connection. Oracle Database automatically creates one database service when the database is created. For many installations, this may be all you need. For more flexibility in the management of the workload using the database, Oracle Database enables you to create multiple services and specify which database instances offer the services.

Caution:
By default, any named user may create a server pool. To restrict the operating system users that have this privilege, Oracle strongly recommends that you add specific users to the CRS Administrators list. See *Oracle Clusterware Administration and Deployment Guide* for more information about adding users to the CRS Administrators list.

You can define services for both policy-managed and administrator-managed databases.

- **Policy-managed database**: When you define services for a policy-managed database, you assign the service to a server pool where the database is running. You can define the service as either uniform (running on all instances in the server pool) or singleton (running on only one instance in the server pool).
- **Administrator-managed database**: When you define a service for an administrator-managed database, you define which instances normally support that service.These are known as the `PREFERRED` instances. You can also define other instances to support a service if the preferred instance fails. These are known as `AVAILABLE` instances.

Services are integrated with the Database Resource Manager, which enables you to restrict the resources that are used by a service within an instance. In addition, Oracle Scheduler jobs can run using a service, as opposed to using a specific instance.

About Service Failover in Administrator-Managed Databases

When you specify a preferred instance for a service, the service runs on that instance during normal operation. Oracle Clusterware attempts to ensure that the service always runs on all the preferred instances that have been configured for a service. If the instance fails, then the service is relocated to an available instance. You can also manually relocate the service to an available instance.

If a service fails over to an available instance, then the service is not moved back to its preferred instance automatically. However, you can automate the relocation of a service to its preferred instance by using a callout. For more information about callouts, see "About FAN Callouts". An example callout script for relocating services back to their preferred instances is available on Oracle Technology Network at: http://www.oracle.com/technetwork/database/enterprise-edition/twpracwkldmgmt-132994.pdf

You do not have to specify available instances for a service. If you do not specify preferred or available instances when you create a service, then by default, every instance in the Oracle RAC database is a preferred instance for that service. However, if you configure a preferred instance for a service, but do not specify at least one available instance for the service, then the service does not relocate to another instance if the preferred instance fails.

You can also specify an instance as Not Used. This setting means the service does not run on the instance, even if the preferred instance for the service fails.

About Service Failover in Policy-Managed Databases

When you specify that a service is UNIFORM, Oracle Clusterware attempts to ensure that the service always runs on all the available instances for the specified server pool. If the instance fails, then the service is no longer available on that instance. If the cardinality of the server pool increases and a instance is added to the database, then the service is started on the new instance. You cannot manually relocate the service to a specific instance.

When you specify that a service is SINGLETON, Oracle Clusterware attempts to ensure that the service always runs on only one of the available instances for the specified server pool. If the instance fails, then the service fails overs to a different instance in the server pool. You cannot specify which instance in the server pool the service should run on.

For SINGLETON services, if a service fails over to an new instance, then the service is not moved back to its original instance when that instance becomes available again.

See Also:

- "About FAN Callouts"
- "Creating Services"
- "About Workload Management"

About Automatic Starting of Services

When you define a service, you can also define the management policy for that service. You can choose either an automatic or manual management policy:

- **automatic**: The service always starts when the database starts.

  Note:

  When you use automatic services in an administrator-managed database, during planned database startup, services may start on the first instances that become available rather than their preferred instances.

- **manual**: Requires you to start the service manually after the database starts. Prior to Oracle Database 11*g* release 2 (11.2), all services worked as though they were defined with a manual management policy.

See Also:

- "Creating Services"
- "About Workload Management"

About the Database Resource Manager

The Database Resource Manager is a database feature you can use to control the database resources allocated to users, applications, and services. This approach ensures that users, applications, and services receive their share of the available database resources. The Database Resource Manager enables an Oracle RAC database running on one or more nodes to support multiple applications and mixed workloads with optimal efficiency.

The Database Resource Manager provides the ability to prioritize work within an Oracle database or your Oracle RAC environment. For example, high priority users, such as online workers, would get more resources to minimize response time, while lower priority users, such as batch jobs or reports, would get fewer resources, and could take longer to run. This allows for more granular control over resources.

Resources are allocated to users according to a resource plan specified by the database administrator. The following terms are used in specifying a resource plan:

- A *resource plan* specifies how the resources are to be distributed among various users (resource consumer groups).
- *Resource consumer groups* allow the administrator to group user sessions by resource requirements. Resource consumer groups are different from user roles; one database user can have different sessions assigned to different resource consumer groups.
- *Resource allocation methods* are the methods or policies used by the Database Resource Manager when allocating for a particular resource. Resource allocation methods are used by resource consumer groups and resource plans. The database provides the resource allocation methods that are available, but the DBA determines which method to use.
- *Resource plan directives* are a means of assigning consumer groups to particular plans and partitioning resources among consumer groups by specifying parameters for each resource allocation method.
- *Subplans*, which the DBA can create within a resource plan, allow further subdivision of resources among different users of an application.
- *Levels* provide a mechanism to specify distribution of unused resources among available users. Up to eight levels of resource allocation can be specified.

The Database Resource Manager enables you to map a resource consumer group to a service so that users who connect using that service are members of the specified resource consumer group, and thus restricted to the resources available to that resource consumer group.

To learn more about managing the Database Resource Manager using Enterprise Manager:

1. Access the Database Home page.

2. At the top of the page, click **Server** to display the Server page.
3. In the Resource Manager section, click **Getting Started**.

See Also:

- "About Workload Management"
- *Oracle Database Administrator's Guide* for more information about the Database Resource Manager

About Fast Application Notification (FAN)

One of the main requirements of a highly available application is for it to be quickly notified when something happens to critical system components. This allows the application to execute event-handling programs. The timely execution of such programs minimizes the time it takes to react to cluster resource organizations and the impact of cluster component failures by avoiding costly connection time outs and application time outs.

Fast Application Notification is a notification mechanism that Oracle RAC uses to notify other processes about cluster configuration and service-level information, including status changes such as `UP` or `DOWN` events. FAN `UP` and `DOWN` events can apply to instances, services, and nodes. FAN also publishes Load Balancing Advisory events.

FAN enables the automated recovery of applications when cluster components fail. For cluster configuration changes, the Oracle RAC high availability framework publishes a FAN event immediately when a change occurs regarding the state of the instances in the cluster. Instead of waiting for the application to query the database and detect a problem, applications can receive FAN events and react immediately.

FAN `UP` and `DOWN` events provide the following benefits:

- For `DOWN` events, the disruption to the application can be minimized because sessions that are connected to the failed instance or node can be terminated. Incomplete transactions can be terminated and the application user notified immediately. Application users who request connections are directed to instances that are started and are providing the requested service.
- For `UP` events, when services and instances are started, new connections can be created so that the application can immediately take advantage of the extra resources.

Oracle Clusterware and Oracle RAC utilize Oracle Notification Service (ONS) to propagate FAN messages both within the Oracle cluster and to client or mid-tier machines. ONS is installed with Oracle RAC and the Oracle Clusterware resources to manage the ONS daemon are created automatically during the installation process. ONS

daemons run locally sending messages to and receiving messages from a configured list of nodes (where other ONS daemons are active).

About FAN Callouts

FAN callouts are server-side executable files that Oracle RAC runs immediately when high availability events occur. A callout is essentially a shell script or precompiled executable written in any programming language. Some examples of how you can use FAN callouts to automate the actions performed when events occur in a cluster configuration are as follows:

- Starting and stopping server-side applications
- Relocating low-priority services when high-priority services come online
- Sending text or numeric messages to pagers
- Executing shell scripts

The executable files for FAN callouts are stored in the *Grid_home*/racg/usrco subdirectory. If this subdirectory does not exist in your Grid home, then you must create this directory with the same permissions and ownership as the *Grid_home*/racg/tmp subdirectory.

All executables in the *Grid_home*/racg/usrco subdirectory are executed immediately, in an asynchronous fashion, when a FAN event received through the ONS. A copy of the executable files used by FAN callouts should be available on every node that runs Oracle Clusterware. Example callout scripts are available on Oracle Technology Network at http://www.oracle.com/technetwork/database/enterprise-edition/twpracwkldmgmt-132994.pdf

See Also:

- "About Connection Load Balancing"
- "About the Load Balancing Advisory"
- *Oracle Real Application Clusters Administration and Deployment Guide* for more information about configuring Fast Application Notification and FAN callouts

About the Load Balancing Advisory

The Load Balancing Advisory provides information to applications or clients about the current service levels that the Oracle RAC database instances are providing. Applications can take advantage of the load balancing Fast Application Notification (FAN) events to direct work requests to the instance in the cluster that provides the best performance based on the workload management directives that you have defined

for that service. Also, when an instance is restarted, Oracle RAC uses FAN events to notify the application's connection pool so that the connection pool can create connections to the recently started instance and take advantage of the additional resources that this instance provides

The load balancing advisory is integrated with the Automatic Workload Repository built into Oracle Database 11*g*. The Automatic Workload Repository measures response time and CPU consumption for each service.

The advice given by the Load Balancing Advisory takes into account the power of the server and the current workload of the service on the server. Enabling the Load Balancing Advisory helps improve the throughput of applications by not sending work to instances that are overworked, running slowly, not responding, or have failed.

Your application can take advantage of the Load Balancing Advisory without any programmatic changes if you use an integrated Oracle client, one that has the Run-time Connection Load Balancing feature. Due to the integration with FAN, Oracle integrated clients are more aware of the current status of an Oracle cluster. This prevents client connections from waiting or trying to connect to an instance that is no longer available. The integrated clients for FAN events include Oracle Database 11*g* JDBC, Oracle Database 11*g* ODP.NET, and Oracle Database 11*g* Oracle Call Interface (OCI).

You configure your Oracle RAC environment to use the Load Balancing Advisory by defining service-level goals for each service used. Defining a service-level goal enables the Load Balancing Advisory for that service and enables the publication of FAN load balancing events. There are two types of service-level goals for Run-time Connection Load Balancing:

- Service Time—The Load Balancing Advisory attempts to direct work requests to instances according to their response time. Load Balancing Advisory data is based on the elapsed time for work done by connections using the service, and the available bandwidth to the service. This goal is best suited for workloads that require varying lengths of time to complete, for example, an internet shopping system.
- Throughput—The Load Balancing Advisory measures the percentage of the total response time that the CPU consumes for the service. This measures the efficiency of an instance, rather than the response time. This goal is best suited for workloads where each work request completes in a similar amount of time, for example, a trading system.

If you do not select the Enable Load Balancing Advisory option, then the service-level goal is set to None, which disables load balancing for that service.

About Connection Load Balancing

Oracle Net is a software component that resides on the client and on the Oracle database server. It establishes and maintains the connection between the client application and the server, and exchanges messages between them using industry standard protocols. For the client application and a database to communicate, the client application must specify location details for the database it wants to connect to, and the database must provide some sort of identification or address.

On the database server, the Oracle Net Listener, commonly known as the listener, is a process that listens for client connection requests. The configuration file for the listener is the `listener.ora`.

Oracle Database 11*g* database clients use SCAN and the easy connect method to connect to the database. SCAN can resolve to multiple IP addresses, reflecting multiple listeners in the cluster handling public client connections. When using the easy connect method, you do not have to configure any client network files. You simply specify a connect identifier with the following format:

SCAN[:*port*]/*service_name*

SCAN represents the SCAN for your cluster. Specifying the TCP port identifier is optional. The *service_name* is the name of a database service.

You can also use Net Configuration Assistant (NETCA) to create a **net service name**, a simple name for the database service. The net service name resolves to the **connect descriptor**, which is the network address of the database and the name of the database service. The address portion of the connect descriptor is actually the protocol address of the listener. The client uses a connect descriptor to specify the database or instance to which the client wants to connect.

When a net service name is used, establishing a connection to a database instance takes place by first mapping the net service name to the connect descriptor. This mapped information is stored in one or more repositories of information that are accessed using naming methods. The most commonly used naming method is Local Naming, where the net service names and their connect descriptors are stored in a localized configuration file named `tnsnames.ora`.

When the client connects to the cluster database using a service, you can use the Oracle Net connection load balancing feature to spread user connections across all the instances that are supporting that service. There are two types of load balancing that you can implement: client-side and server-side load balancing. In an Oracle RAC database, client connections should use both types of connection load balancing. When you create an Oracle RAC database using Oracle Database Configuration Assistant (DBCA), DBCA configures and enables server-side load balancing by default.

See Also:

- "Verifying Oracle Net Supports Newly Created Services"
- *Oracle Database 2 Day DBA*

About Client-Side Load Balancing

Client-side load balancing balances the connection requests across the listeners. When the listener receives the connection request, the listener connects the user to an instance that the listener knows provides the requested service.

Client-side load balancing is defined in your client connection definition by setting the parameter `LOAD_BALANCE=yes` in the `tnsnames.ora` file. When you set this parameter to `yes`, the Oracle client randomly selects an address from the address list, and connects to that node's listener. This balances client connections across the available listeners in the cluster.

When you create an Oracle RAC database using DBCA, the assistant creates a sample load balancing connection definition in the `tnsnames.ora` file.

Client-side load balancing includes connection failover. With connection failover, if an error is returned from the chosen address, then Oracle Net Services tries the next address in the address list until either a successful connection is made or it has exhausted all the addresses in the list.

About Server-Side Load Balancing

With server-side load balancing, the listener directs a connection request to the best instance currently providing the service by using information from the Load Balancing Advisory.

For each service, you can define the method the listener uses for load balancing by setting the connection load balancing goal. You can use a goal of either long or short for connection load balancing. These goals have the following characteristics:

- Short—Connections are distributed across instances based on the amount of time that the service is used. Use the Short connection load balancing goal for applications that have connections of brief duration.
- Long—Connections are distributed across instances based on the number of sessions in each instance, for each instance that supports the service. Use the Long connection load balancing goal for applications that have connections of long duration. This is typical for connection pools and SQL*Forms sessions. Long is the default connection load balancing goal.

Any services created by using DBCA use the Long connection load balancing goal by default.

Note:
If you did not use DBCA to create your database, or if you are using listener ports other than the default of 1521, then you must configure the `LOCAL_LISTENER` and `REMOTE_LISTENER` database initialization parameters for your cluster database to point to *`SCAN:port`*.

About Run-time Connection Load Balancing

Run-time Connection Load Balancing is a feature of Oracle connection pools that can distribute client work requests across the instances in an Oracle RAC database based on the Load Balancing Advisory information. The connection allocation is based on the current performance level provided by the database instances as indicated by the Load Balancing Advisory FAN events. This provides load balancing at the transaction level, instead of load balancing at the time of the initial database connection.

With Run-time Connection Load Balancing, applications use Load Balancing Advisory information to provide better performance to users. OCI Session pools and ODP.NET connection pools support Run-time Connection Load Balancing. For Java applications, Oracle recommends the Universal Connection Pool (UCP). The Universal Connection Pool is integrated to take advantage of Load Balancing Advisory information. UCP, introduced in Oracle Database 11*g* patch set 1 (11.1.0.7), can be used against Oracle Database 10*g* or Oracle Database 11*g*.

You must enable the client data source for Run-time Connection Load Balancing with a service that has the following configuration:

- The Load Balancing Advisory is enabled and the service-level goal is set to either Service Time or Throughput.
- The service connection load balancing goal is set to Short.

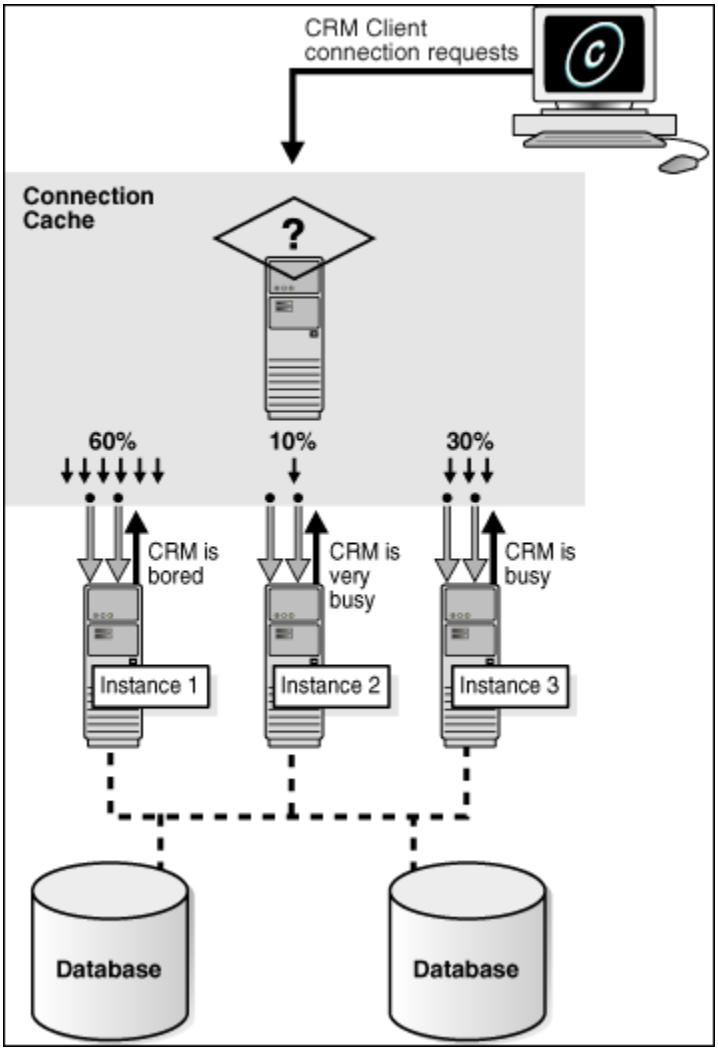
Figure 7-1, "Run-time Connection Load Balancing" illustrates Run-time Connection Load Balancing. In this illustration, the Oracle RAC database has three instances. Suppose that the Load Balancing Advisory indicates that Instance1 and Instance3 have the best performance, while Instance2 currently has less than optimal performance. When Run-time Connection Load Balancing is enabled on the implicit connection cache, the following process occurs:

1. A client requests a connection from the connection pool.
2. Run-time Connection Load Balancing selects the connection that belongs to the most efficient (best) instance from the connection pool. In Figure 7-1, there are three possible nodes to which the connection can be routed. Instance1, which

has the least amount of CPU workload, is currently being assigned about 60 percent of the incoming connections. Instance2, which is currently overloaded, is only being assigned around 10 percent of the incoming connections. Instance3, which has a high workload, is being assigned around 30 percent of the incoming connections. The best instance to handle the connection request in this case would be Instance1.

3. The client receives the connection that would process the work request with the best response time.

Figure 7-1 Run-time Connection Load Balancing



Description of "Figure 7-1 Run-time Connection Load Balancing"

Oracle Database 11*g* introduces an additional flag in the load balancing advisory event called affinity hint. The affinity hint is automatic when load balancing advisory is turned on through setting the goal on the service. This flag is for temporary affinity that lasts for the duration of a web session. Web conversations often connect and disconnect

many times during the entire session. During each of these connects, it may access the same or similar data, for example, a shopping cart, Siebel, and so on. Affinity can improve buffer cache efficiency, which lowers cpu usage and transaction latency. The Affinity Hint is a flag that indicates if Affinity is active or inactive for a particular instance and service combination. Different instances offering the same service can have different settings for the Affinity Hint.

Applications using Oracle Database 11*g* and UCP, can take advantage of this new affinity feature. If the affinity flag is turned on in the Load Balancing Advisory event, then UCP creates an Affinity Context for the Web session such that when that session does a get connection from the pool, the pool always tries to give it a connection to the instance it connected to the first time it acquired a session. The choice of instance for the first connection is based on the current load balancing advisory information.

Creating Services

To manage workloads, you can define services that you assign to a particular application or to a subset of an application's operations. You can also use services to manage the workload for different types of work. You can create a service using Oracle Enterprise Manager Database Control.

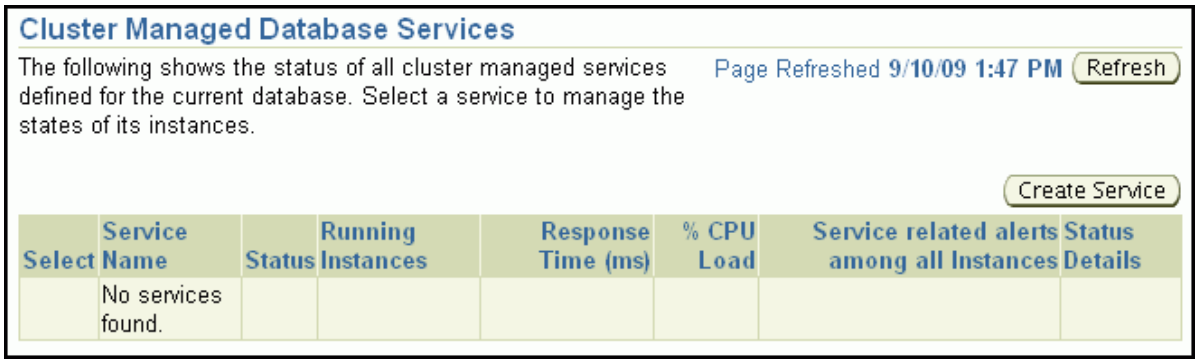
To create a service:

1. On the Cluster Database Home page, click **Availability**.

   The Availability page appears.

2. Click **Cluster Managed Database Services** in the Services section. Enter or confirm the credentials for the Oracle RAC database and host operating system and click **Continue**.

   The Cluster Managed Database Services page appears.

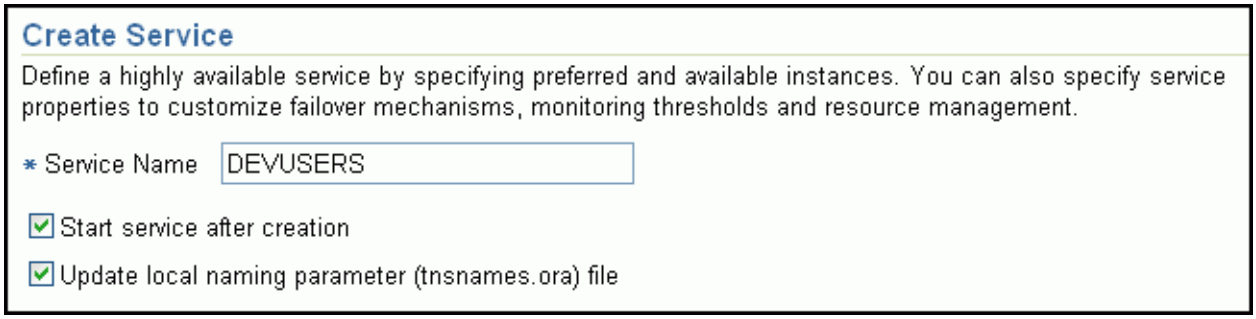


Description of the illustration wlm005.gif

3. Click **Create Service**.

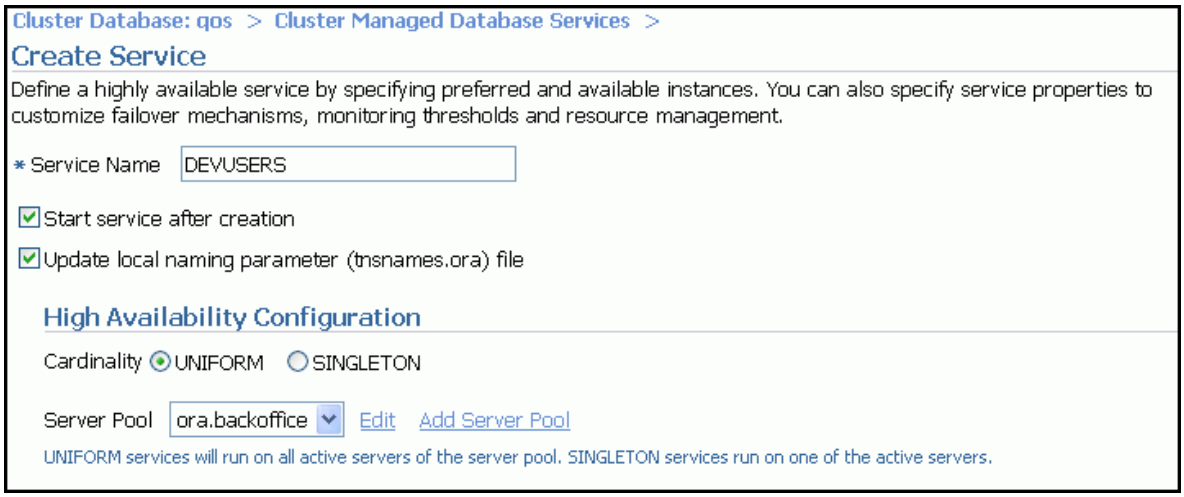
   The Create Service page appears.

4. Enter the name of your service in the Service Name field, for example, `DEVUSERS`.
5. Select **Start Service after creation** if you want the service to be started after it is created. Select **Update local naming parameter (tnsnames.ora) file** if you want DB Control to add the new service to the local Oracle Net Services `tnsnames.ora` file.



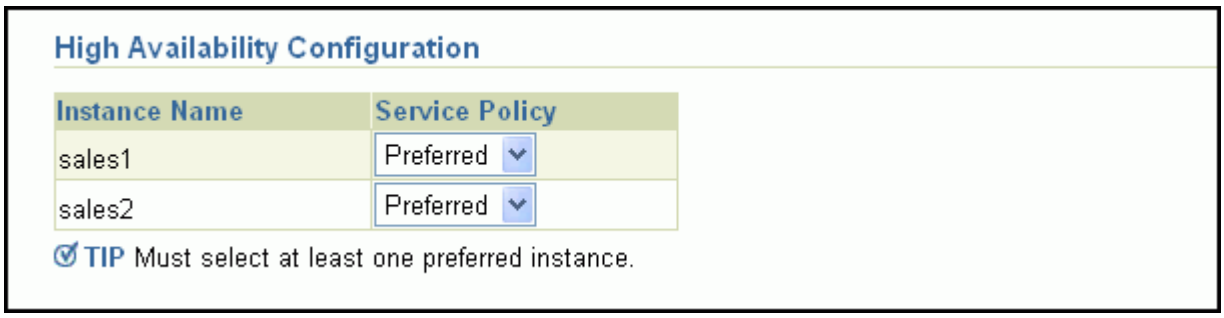
Description of the illustration wlm006_new.gif

6. (Policy-managed databases only) For the service type, select **UNIFORM** or SINGLETON.



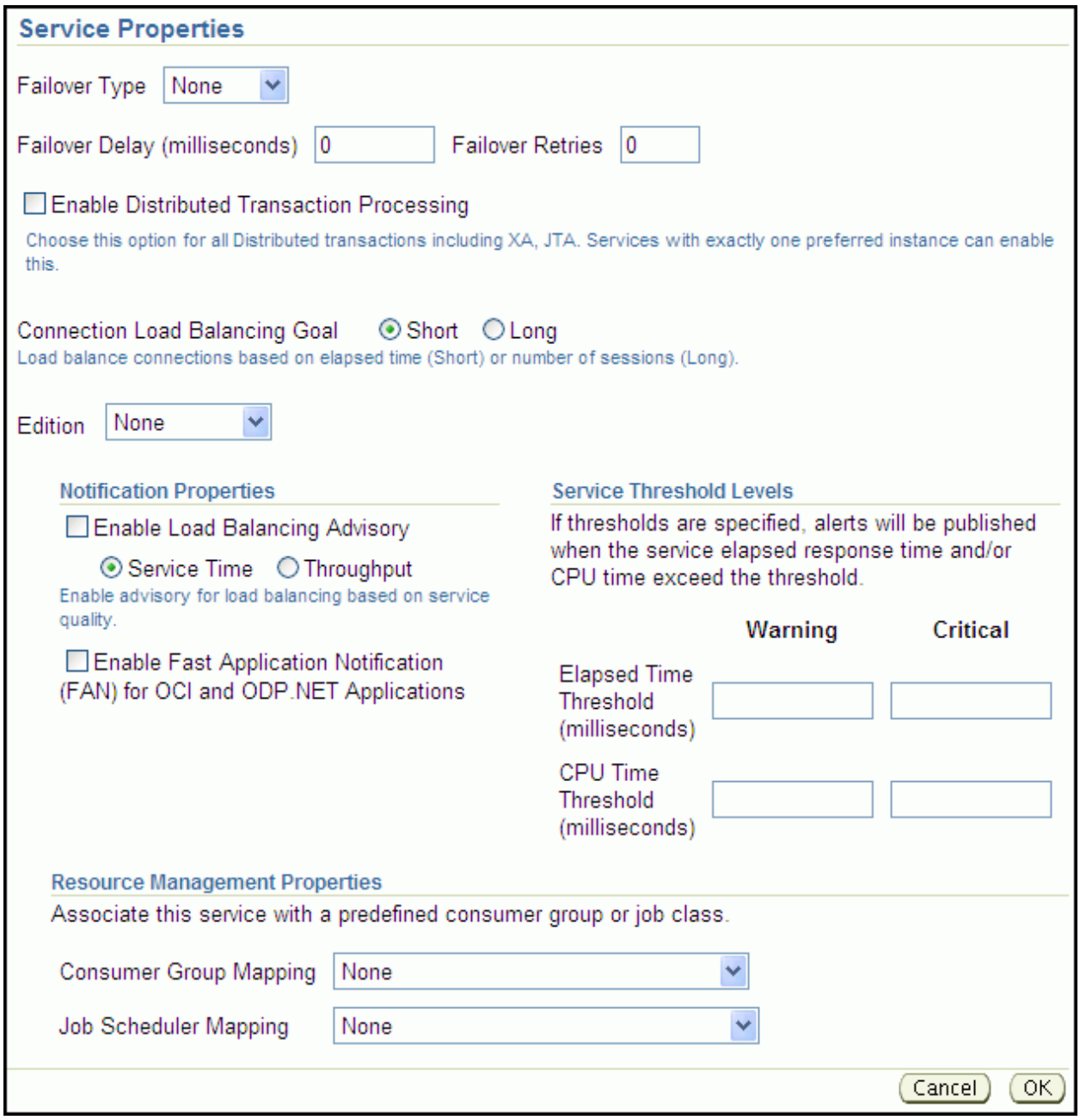
Description of the illustration wlm006.gif

7. (Administrator-managed databases only) For each instance choose whether that instance is a Preferred instance or an Available instance. If you do not want the service to run on an instance under any circumstances, then set the Service Policy for that instance to Not Used.



Description of the illustration wlm006_admin_new.gif

8. In the Service Properties section, select **Short** for Connection Load Balancing Goal to distribute the connection workload based on elapsed time instead of the overall number of connections. Otherwise, choose **Long**.
9. Select **Enable Load Balancing Advisory** under the sub-heading Notification Properties to enable the Load Balancing Advisory for this service, as shown in the following screenshot. Choose a service-level goal of either **Service Time** or **Throughput**.



Description of the illustration wlm007_editions.gif

10. Select **Enable Fast Application Notification** under the heading Notification Properties if this service is used by an Oracle Call Interface (OCI) or ODP.NET application, and you want to enable FAN.
11. In the Service Threshold Levels section, you can optionally set the service-level thresholds by entering a value in milliseconds for Warning and Critical thresholds for the Elapsed Time and CPU Time metrics.
12. If you want to use a Resource Plan to control the resources used by this service, then select the name of the consumer group from the Consumer Group Mapping list in the Resource Management Properties section. For example, you might

choose the `LOW_GROUP` consumer group to give development users low priority to database resources.

Note:

You cannot change the consumer group name for a service on the Edit Service page. This is because there may be several consumer groups associated with a given service. However, the Edit Service page contains a link to the Resource Consumer Group Mapping page, where you can modify the consumer group mapping for the service.

13. If this service is used by a specific Oracle Scheduler job class, then you can specify the mapping by selecting the name from the Job Scheduler Mapping list in the Resource Management Properties.
14. Click **OK** to create the service.

See Also:

- "About Workload Management"
- "About Connection Load Balancing"
- "About the Load Balancing Advisory"
- "About Fast Application Notification (FAN)"
- "Administering Services"
- *Oracle Database Administrator's Guide*

Administering Services

You can create and administer services using Enterprise Manager. You can also use the SRVCTL utility to perform most service management tasks.

The following sections describe how to manage services for your cluster database:

- About Service Administration Using Enterprise Manager
- Using the Cluster Managed Database Services Page
- Verifying Oracle Net Supports Newly Created Services

About Service Administration Using Enterprise Manager

The Cluster Managed Database Services page is the master page for beginning all tasks related to services. To access this page, go to the Cluster Database Maintenance page, then click **Cluster Managed Database Services** in the Services section. You can use this page and links from this page to do the following:

- View a list of services for the cluster.
- View the instances on which each service is currently running.

- View the status for each service.
- Create or edit a service.
- Start or stop a service.
- Enable or disable a service.
- Perform instance-level tasks for a service.
- Delete a service.

See Also:

- "Administering Services"
- "About Oracle Services"
- "Creating Services"

Using the Cluster Managed Database Services Page

When managing services using Enterprise Manager, you use the Cluster Managed Database Services page.

On the Cluster Managed Database Services page you can perform the following tasks:

- View a list of services for the cluster, the instances on which each service is currently running, the status of each service, the server pool associated with the service, the responsiveness of the service, and any alerts or messages for that service.
- Start or stop a service.
- Test the connection for a service, or display the Oracle Net TNS strings being used.
- Access the Create Service page.
- Edit, delete, enable, or disable a service.
- Access the Edit Server Pool page for the server pool associate with each service.

To access the Cluster Managed Database Services page:

1. On the Cluster Database Home page, click the **Availability** tab.
2. On the Availability subpage, under the Services heading, click **Cluster Managed Database Services**.

   The Cluster and Database Login page appears.

3. Enter credentials for the database and for the cluster that hosts the Oracle RAC database, then click **Continue**.

   The Cluster Managed Database Services page appears and displays the services that are available on the cluster database instances.

See Also:

- "About Service Administration Using Enterprise Manager"
- "About Oracle Services"
- "Creating Services"

Verifying Oracle Net Supports Newly Created Services

By using GNS and configuring a SCAN for your cluster, you no longer have to modify the network setup on each client. This section shows you how to verify that the new service is recognized by Oracle Net and available to the database clients.

When using utilities to manage your cluster, databases, database instances, Oracle Automatic Storage Management (Oracle ASM), and listeners, use the appropriate binary that is in the home directory of the object or component you are managing and set your `ORACLE_HOME` environment variable to point to this directory. For example, to use `lsnrctl` to manage a database instance or its listener, then use the binaries located in the Oracle home where the database instance or listener is running, and set the `ORACLE_HOME` environment variable to the location of that Oracle home.

To verify Oracle Net Services supports the newly created service:

1. Determine if the listener on the local node recognizes the new service by using the following command:

2. `lsnrctl status LISTENER_SCAN1`

   You should see a list for the new service, similar to the following:

   ```
   Services Summary...
   Service "DEVUSERS.example.com" has 2 instance(s).
     Instance "sales1", status READY, has 1 handler(s) for
   this service...
     Instance "sales2", status READY, has 1 handler(s) for
   this service...
   ```

   The displayed name for your newly created service, for example `DEVUSERS.example.com`, is the service name you use in your connection strings.

3. Test the Oracle Net Services configuration by attempting to connect to the Oracle RAC database using SQL*Plus and the SCAN. The connect identifier for easy connect naming has the following format:

4. `"[//]`*`SCAN`*`[:`*`port`*`]/`*`service_name`*`"`

   SCAN is the SCAN for your cluster, which defaults to *cluster_name*.*GNS_sub_domain*. The *service_name* is the name of the database

service you want to use for connecting to the database. You can optionally specify the TCP port number on which the Oracle SCAN listener listens for connections.

For example, you might use the following commands to connect to the DEVUSERS service for Oracle RAC database in the `docrac` cluster:

```
$ sqlplus /nolog
SQL> CONNECT system@"docrac/DEVUSERS.example.com"
Enter password: password
```

After you enter the password, you should see a message indicating you are successfully connected to the Oracle RAC database. If you get an error message, then examine the connect identifier and verify the user name, password, and service name were typed in correctly and all the information is correct for your Oracle RAC environment.

Configuring Clients for High Availability

There are two central elements to consider when automating failover for application clients. First, clients that are connected at the time of failure must be quickly and automatically notified that a failure has occurred to avoid waiting for TCP/IP network time-outs before attempting to connect to the new database instance (such time-outs range anywhere from eight minutes to two hours, depending on operating system). Oracle RAC configurations use Fast Application Notification (FAN) to notify JDBC clients, OCI clients, and ODP.NET clients. FAN event notifications and callouts enable automatic and fast redirection of clients after a site failure.

The second central element of client failover, is the redirection of clients to the new instance after a failure has occurred, which can be implemented using services. When you create services in an Oracle RAC database, if an instance to which you have assigned a service becomes unavailable, then Oracle RAC relocates the service to an available instance in the database. Users can access the service independent of the instance providing it because, using listener registration, all listeners in the cluster are aware of which instances are currently providing a service when a connection request comes in.

In addition to FAN, the client can limit the time it waits for a response from the server before trying the next address in the connection list without waiting for the TCP/IP network time out. To enable the time out limit, set the `sqlnet.ora` parameter `sqlnet.outbound_connection_timeout` = *x*, where *x* represents the maximum amount of time, in seconds, for a client to establish an Oracle Net connection to the database instance, for example three seconds.

```
sqlnet.outbound_connection_timeout = 3
```

Note:
You should not configure the sqlnet.outbound_connection_timeout parameter on the server `sqlnet.ora` file, as this can impact Oracle Clusterware operations.

This section deals with configuration FAN for application clients, and contains the following topics:

- Configuring JDBC Clients
- Configuring OCI Clients
- Configuring ODP.NET Clients

See Also:

- "About Fast Application Notification (FAN)"
- "About Oracle Services"

Configuring JDBC Clients

An application that uses JDBC connections can use a connection pool, so an existing connection can be used in the pool instead of making a new connection to the database every time you access data. The Universal Connection Pool (UCP) is a Java-based connection pool that supports any type of connection (JDBC, LDAP, JCA), to any type of database (Oracle or non-Oracle) with any middle tier (Oracle or non-Oracle). It also supports standalone deployments such as TopLink or BPEL. UCP includes integration features of Oracle Database such as Oracle RAC, including Fast Connection Failover, Run-time Connection Load Balancing, and Connection Affinity for Oracle RAC instances.

Without Fast Connection Failover, if a node goes down and the application attempts to connect to that instance, then the connection request might hang for several minutes waiting for the TCP time-out to be reached. If you chose not to use a connection pool with the integrated FCF feature, then your application can still be notified of Oracle RAC high availability events by configuring the Connection Failure Notification feature.

This section explains how to configure Java Database Connectivity (JDBC) clients for:

- Configuring JDBC Clients for Fast Connection Failover
- Configuring JDBC Clients for Connection Failure Notification

Note:
Connection Failure Notification is redundant with Fast Connection Failover as implemented by the UCP. You should not configure both within the same application.

See Also:

- "About Workload Management"
- "About Oracle Services"
- *Oracle Universal Connection Pool for JDBC Developer's Guide*

Configuring JDBC Clients for Fast Connection Failover

The Universal Connection Pool subscribes to the FAN Load Balancing events automatically when you configure fast connection failover. Instead of randomly assigning a free connection to a work request, the connection pool chooses the connection that gives the best service according to the latest information it has received. If a node becomes hung, then the connection pool gradually shifts connections from the hung node to other nodes in the cluster.

To configure JDBC clients for Fast Connection Failover:

1. Use the Cluster Managed Services page in Oracle Enterprise Manager Database Control or Oracle Enterprise Manager Grid Control to create new services. See "Creating Services" for more information about creating services.
2. Enable fast connection failover for JDBC clients by setting the UCP DataSource property FastConnectionFailoverEnabled to `TRUE`. Also, Configure a remote Oracle Notification Services (ONS) subscription on the JDBC client so that an ONS daemon is not required on the client, as shown in the following example:

```
3. PoolDataSource pds =
   PoolDataSourceFactory.getPoolDataSource();
4. Pds.setConnnectionPoolName(”FCFSampleUCP”);
5. pds.setONSConfiguration(”nodes=racnode1:4200,racnode2:4
   200”);
6. pds.setFastConnectionFailoverEnabled(true);
7. pds.setConnectionFactoryClassName(”oracle.jdbc.pool.Ora
   cleDataSource”);
```

   The remote ONS subscription must contain every host that the client application can use for failover.

8. Set the `oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR` property to a nonzero value on the data source. When this property is set, if the JDBC client attempts to connect to a host that is unavailable, then the connection attempt is bounded to the time specified for `oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR`. After the specified time has elapsed and a successful connection has not been made, the client attempts to connect to the next host in the address list. Setting this property to a value of three seconds is sufficient for most installations.
9. Configure JDBC clients to use a connect descriptor that includes the SCAN and the service name, and optionally the port that the SCAN listener listens on, for example:

```
10. @//docrac.example.com:1521/orcl_JDBC
```

   Note:

Do not configure Transparent Application Failover (TAF) with Fast Connection Failover for JDBC clients as TAF processing can interfere with FAN ONS processing.

In the JDBC application, you would connect to the database using a connect string, such as the following one:

```
pds.setURL("jdbc:oracle:thin:@//docrac.example.com:1521
/orcl_JDBC")
```

If you are using a JDBC driver, then you must include the complete connect descriptor in the URL because the JDBC driver does not use Oracle Net.

11. Make sure that both `ucp.jar` and `ons.jar` are in the application CLASSPATH.

See Also:

- "About Fast Application Notification (FAN)"
- "Configuring Clients for High Availability"
- *Oracle Database JDBC Developer's Guide* for more information about fast connection failover and configuring ONS
- *Oracle Universal Connection Pool for JDBC Developer's Guide* for more information
- *Oracle Database 2 Day + Java Developer's Guide* for information about creating a method to authenticate users
- *Oracle Real Application Clusters Administration and Deployment Guide* for information about configuring client failover

Configuring JDBC Clients for Connection Failure Notification

JDBC Connection Failure Notification enables standalone Oracle JDBC connections to respond to node failures quicker. This feature enables a stand-alone connection to listen for Oracle RAC node down events and respond to a node down event by marking the connection closed, permitting user applications to begin recovery quicker. It does not alter the behavior of the application, except that JDBC method calls throw exceptions more quickly after a node goes down.

To configure Connection Failure Notification, you use the ConnectionFailureNotification connection property.

To configure JDBC clients for Connection Failure Notification:

1. Set the Connection Failure Notification property either as a system property using the -D option, or by including it in the connection properties argument when creating a connection.

   To set it as a system property, use a command similar to the following:

   ```
   java -Doracle.jdbc.ConnectionFailureNotification=true
   ```

   To set it within the connection properties argument, use code similar to the following:

   ```
   Properties props = new Properties();
   props.put("ConnectionFailureNotification", "true");
   Connection = DriverManager.getConnection(url, props);
   ```

2. Configure the transport mechanism by which the node down events are received. If ONS is the selected transport mechanism, then use the SetONSConfiguration property, as demonstrated in the following code, where `racnode1` and `racnode2` represent nodes in the cluster that have ONS running on them:

3. `props.setONSConfiguration(”nodes=racnode1:4200,racnode2:4200”);`
4. Make sure that `ons.jar` is in the application CLASSPATH.

See Also:

- "About Oracle RAC High Availability Framework"
- "Configuring Clients for High Availability"
- *Oracle Database JDBC Developer's Guide* for more information about configuring ONS
- *Oracle Database 2 Day + Java Developer's Guide* for information about creating a method to authenticate users

Configuring OCI Clients

The Oracle Call Interface (OCI) provides integration with FAN and Load Balancing Advisory events. To take advantage of the Load Balancing Advisory, you must enable the OCI Session pool. OCI clients can register to receive notifications about Oracle RAC high availability events and respond when events occur. This improves the connection failover response time in OCI and also removes terminated connections from connection and session pools. This feature works for all OCI client applications.

To configure OCI clients to receive FAN notifications:

1. Use the Cluster Managed Services page in Oracle Enterprise Manager Database Control or Oracle Enterprise Manager Grid Control to a create services for the OCI clients. See "Creating Services" for more information about creating services.

   You should configure the primary instance as preferred for that service. For Notification Properties, choose "Enable Fast Application Notification for OCI and ODP.NET Applications".

   See "Administering Services" for more information about modifying services using Enterprise Manager.

2. Enable FAN for OCI clients by initializing the environment with the `OCI_EVENTS` parameter, as in the following example:

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

4. Link the OCI client applications with thread library `libthread` or `libpthread`.
5. In your application, you must check if an event has occurred, using code similar to the following example:

```
6. void evtcallback_fn(ha_ctx, eventhp)
7. ...
8. printf("HA Event received.\n");
9.   if (OCIHandleAlloc( (dvoid *)envhp, (dvoid **)&errhp,
   (ub4) OCI_HTYPE_ERROR,
10.                        (size_t) 0, (dvoid **) 0))
11.     return;
12.   if (retcode = OCIAttrGet(eventhp, OCT_HTYPE_EVENT,
   (dvoid *)&srvhp, (ub4 *)0,
13.                             OCI_ATTR_HA_SRVFIRST,
   errhp))
14.     checkerr (errhp, (sword)retcode;
15.   else {
16.      printf("found first server handle.\n");
17.      /*get associated instance name */
18.      if (retcode = OCIAttrGet(srvhp, OCI_HTYPE_SERVER,
   (dvoid *)&instname,
19.                               (ub4 *)&sizep,
   OCI_ATTR_INSTNAME, errhp))
20.        checkerr (errhp, (sword)retcode);
21.      else
22.        printf("instance name is %s.\n", instname);
```

23. After a HA event is received, clients and applications can register a callback that is invoked whenever a high availability event occurs, as shown in the following example:

```
24. /*Registering HA callback function */
25.   if (checkerr(errhp, OCIAttrSet(envhp, (ub4)
  OCI_HTYPE_ENV,
26.                                (dvoid *)evtcallback_fn,
  (ub4) 0,
27.                                (ub4)OCI_ATTR_EVTCBK,
  errhp)))
28.   {
29.     printf("Failed to set register EVENT
  callback.\n");
30.     return EX_FAILURE;
31.   }
32.   if (checkerr(errhp, OCIAttrSet(envhp, (ub4)
  OCI_HTYPE_ENV,
33.                                   (dvoid *)evtctx, (ub4)
  0,
34.                                   (ub4)OCI_ATTR_EVTCTX,
  errhp)))
35.   {
36.     printf("Failed to set register EVENT callback
  context.\n");
37.     return EX_FAILURE;
38.   }
39. return EX_SUCCESS;
```

After registering an event callback and context, OCI calls the registered function once for each high availability event.

Configuring ODP.NET Clients

Oracle Data Provider for .NET (ODP.NET) connection pools subscribe to FAN notifications from Oracle RAC that indicate when nodes are down and when services are up or down. Based on these notifications, ODP.NET connection pools make idle connections, connections that were previously connected to node that failed, available again. It also creates new connections to healthy nodes if possible.

ODP.NET provides Run-time Connection Load Balancing to provide enhanced load balancing of the application workload. Instead of randomly selecting an available connection from the connection pool, it chooses the connection that can provide the best service based on the current workload information.

The procedures for enabling ODP.NET are similar to the procedures for enabling JDBC in that you must set parameters in the connection string to enable FCF. This section

explains how to configure Oracle Data Provider for .NET (ODP.NET) clients for failover using FAN events.

To configure ODP.Net clients to receive FAN notifications:

1. Use the Cluster Managed Services page in Oracle Enterprise Manager Database Control or Oracle Enterprise Manager Grid Control to a create services for the ODP.NET clients. See "Creating Services" for more information about creating services.

   For Notification Properties, choose "Enable Fast Application Notification for OCI and ODP.NET Applications". Set the Connection Load Balancing Goal to Long.

2. Enable Fast Connection Failover for ODP.NET connection pools by subscribing to FAN high availability events. Do this by setting the `ha events` connection string attribute to `true` either at connection time or in the data source definition. Note that this only works if you are using connection pools (the `pooling` attribute to `true`).

   You can also enable Run-time Connection Load Balancing by setting the `load balancing` connection string attribute to `true`.

   Use code similar to the following, where *username* is the name of the database user that the application uses to connect to the database, *password* is the password for that database user, and the service name is `odpserv`:

```
// C#
using System;
using Oracle.DataAccess.Client;
class HAEventEnablingSample
{
  static void Main()
  {
    OracleConnection con = new OracleConnection();

    // Open a connection using ConnectionString
attributes
    // Also, enable "load balancing"
    con.ConnectionString =
      "User Id=username;Password=password;" +
      "Data Source=//docrac.example.com/odpserv;" +
      "Min Pool Size=10;Connection
Lifetime=120;Connection Timeout=60;" +
```

```
      "HA Events=true;Incr Pool Size=5;Decr Pool
Size=2";

    con.Open();
    // Carry out work against the database here.
    con.Close();
    // Dispose OracleConnection object
    con.Dispose();
    }
 }
```

3. The HA events are published as messages in the SYS.SYS$SERVICE_METRICS queue. You must grant dequeue permission on this queue to the database user that the application uses to connect to the database.

   Use a command similar to the following, where *username* represents the database user that the .NET application uses to connect to the database:

   ```
   execute
   dbms_aqadm.grant_queue_privilege('DEQUEUE','SYS.SYS$SER
   VICE_METRICS', username);
   ```

   The *username* specified in this step equals the *username* used for the User Id argument in the previous step.

See Also:

- "About Fast Application Notification (FAN)"
- "Configuring Clients for High Availability"
- *Oracle Data Provider for .NET Developer's Guide* for more information about event notification and user-registered callbacks
- *Oracle Real Application Clusters Administration and Deployment Guide* for more information about configuring fast application notification for ODP.NET clients.