Oracle Application Express and Oracle Real Application Clusters – Creating a Highly Available Environment for Apex Applications

An Oracle White Paper
June 2008
(Corrected April 2012)

Note: *The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.*

ORACLE

Oracle Application Express and Oracle Real Application Clusters

Oracle Application Express and Oracle Real Application Clusters

INTRODUCTION

Oracle Application Express (APEX), now a standard component with Oracle Database 11g, is a powerful and easy to use web application development platform. It has seen tremendous uptake by customers, with thousands of secure, scalable web-applications deployed into production. Its tight integration with the Oracle Database ensures that application built using Oracle APEX benefit from the inherent reliability, security, scalability and availability offered by the database.

With Oracle APEX increasingly becoming the choice for many commercial ventures, it is extremely important for customers to realize the necessity to plan for ensuring maximum uptime of their applications. This is one of the prime challenges that must be factored into the design of a "highly available" environment.

As organizations come to rely even more on these applications for mission critical operations and delivery of services, it is extremely important that measures be incorporated to mitigate risks from "Unplanned Downtime" due to "systems failures". This unscheduled loss of service can impact essential business process, damage customer confidence and lead to loss of revenue.

While there are a host of solutions available to ensure maximum uptime, this paper intends to provide organizations with an overview of steps on configuring an Oracle APEX instance for a Oracle Real Application Clusters (RAC) database. Oracle RAC is recommended for applications that require high availability (HA), scalability, , and workload management at lower total cost of ownership (TCO). Oracle RAC forms the foundation for Oracle's Grid Technology.

OVERVIEW

Oracle APEX provides users with an ideal rapid web-application development tool, with minimal time taken from planning to deployment of custom business applications.

Oracle Application Express Architecture

Oracle APEX resides completely within the Oracle database and is comprised of data in tables and large amounts of PL/SQL code. An APEX Application is essentially stored as metadata within these tables. When an application is run, the APEX engine reads this metadata and displays the application. In order to access the Oracle APEX engine, and process Client (Web-browser) requests, the "mod_plsql" extension to the HTTP server is utilized. This extension is the communication broker between the Web server and the Oracle APEX objects within the database. The Oracle HTTP Server contains the mod_plsql extension.

Note: Using Oracle Database 11g, you can replace the Oracle HTTP Server with the embedded PL/SQL gateway (EPG).

Oracle Real Application Clusters Architecture Overview

Oracle "Real Application Clusters" (RAC) is part of Oracle's High Availability (HA) Solutions. It is designed to provide "Continuous availability of data for all applications". An Oracle RAC Database is a clustered database, where a group of independent servers cooperate as a single system. Clustering provides fault resilience and modular incremental systems growth over a single symmetric multi-processor (SMP) system. This ensures high availability to users without loss of service or access to mission-critical data. Redundant hardware components allow Oracle RAC Databases to provide HA by avoiding single points-of-failure. Thus an Oracle RAC database is a "Single Database" that can be accessed by multiple instances concurrently. Each instance runs on separate servers or nodes.

The Oracle Clusterware, included with Oracle RAC, provides the infrastructure for a RAC environment by enabling the monitoring and management of Oracle resources within the cluster. Each instance within the cluster has its own SGA and background processes running on separate servers or nodes.

Combining the processing power of servers within a cluster provides greater throughput and scalability than normally available from a single server. Each cluster database instance in an Oracle RAC cluster utilizes its own memory structures and background processes.

Incorporating this technology as part of your IT-infrastructure ensures that all applications, including those built using Oracle APEX draw upon these benefits. Importantly, Oracle RAC ensures that applications that run well on a single

instance database will be highly available, scalable and have improved performance on a RAC setup without requiring any code changes.

High Availability

Oracle RAC, together with Oracle Clusterware and Automatic Storage Management (ASM), provides high availability for applications by removing the database instance, and database server as single points of failure. If a node in the cluster fails, the Oracle Database continues running on the remaining nodes. Individual nodes can be shutdown for maintenance while application users continue to work.

Additionally, Oracle Clusterware can be used to protect application components that exist outside of the standard Oracle database infrastructure protected by RAC. In the implementation described in this paper, Oracle Clusterware is used to protect the Oracle HTTP server from node failure.

ASSUMPTIONS

The audience for this paper is expected to have a basic understanding of the following topics:

- Installation and Configuration of Oracle Database. To know more on how to install and configure Oracle Database 11g, please refer to http://www.oracle.com/pls/db111/to_toc?pathname=install.111/b28264/toc.htm.
- Installation and Configuration of Oracle Application Express. To know more on how to install and configure Oracle Application Express, please refer http://download.oracle.com/docs/cd/E10513_01/doc/install.310/e10496/toc.htm.
- An understanding of Oracle RAC concepts and terminologies. To know more about Oracle RAC, please refer to the Oracle Database 2 Day + Real Application Cluster Guide.

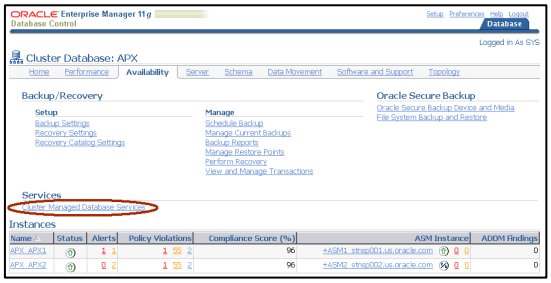
The paper further assumes that you have at least a 2 Node RAC clustered Database (Oracle Database 11g) installed and configured in your environment.

Installing Oracle Application Express

The steps to install Oracle Application Express in an Oracle RAC Environment are the same as in a single instance database, with one small addition. We now have to connect to the database via one of the Oracle RAC Database nodes.

Note: Please verify whether the database meets all the pre-requisites required for installing Oracle Application Express.

1. Connect to the database with DBA privileges using one of the available Oracle RAC nodes



./sqlplus sys/******* as sysdba@APXRAC

2. Verify that the instance is part of the RAC

SQL> select instance from v$thread;

3. Start the Oracle Application Express Installation

@apexins sysaux sysaux temp /i/

The above script is located in the database ORACLE_HOME/apex directory

4. Connect via the second instance and verify whether the FLOWS_030100 user was created

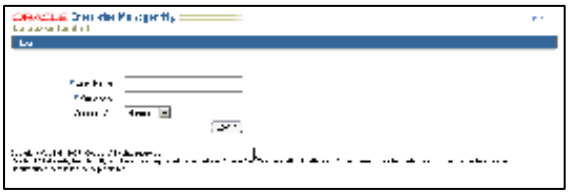
SELECT username FROM dba_users WHERE username = 'FLOWS_030100';

CONFIGURATION POINTS

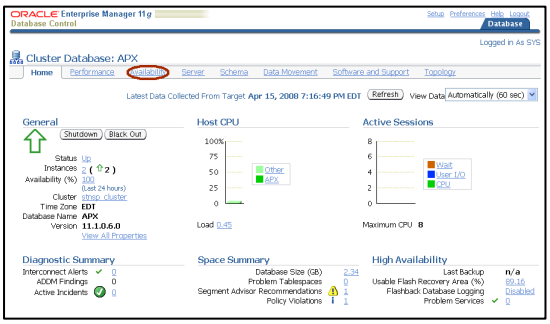
Oracle RAC Service Configuration

Use the Oracle Entreprise Manager 11g to configure a cluster managed database service

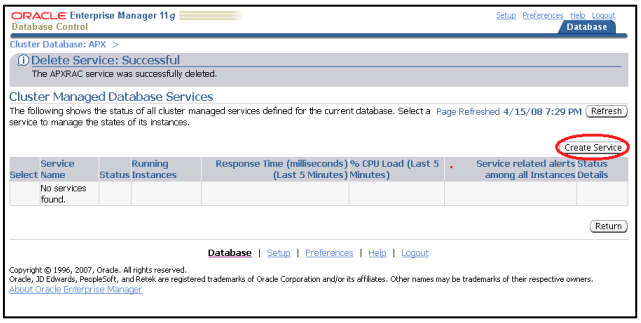
1. Login to the Oracle Enterprise Manager 11g



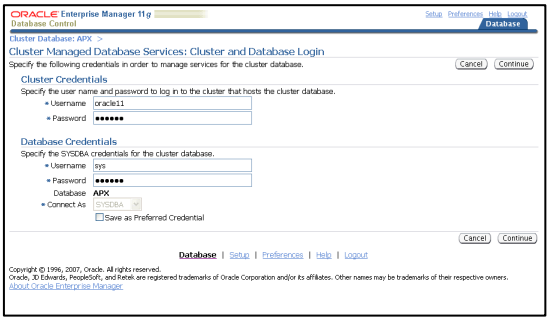
2. Select the "Availability" Tab on the Enerprise Manager Home page



3. Under the “Services” section, select the “Cluster Managed Database Services”

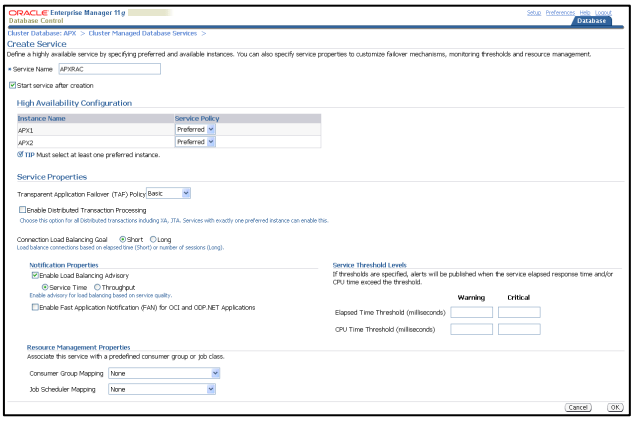


4. Enter the OS and Database credentials

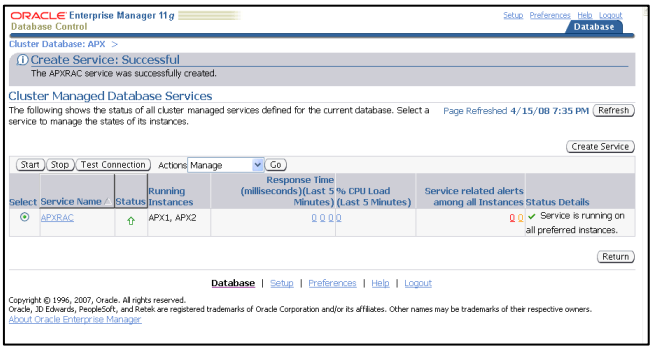


5. Select “Create Service” to add a new Database Service

6. Define the service properties for a high availability configuration



7. Database service creation complete



You may choose to create a service via command line using the following steps. We will configure a database service “APXRAC” for our 2 Node RAC clustered Database.

1. Create a database service “APXRAC” for our database “APX” on nodes “APX1” and “APX2”

   srvctl add service -d APX -s APXRAC -r APX1,APX2 -P BASIC

   Note: srvctl creates the service and client side TAF policies (Basic in the example above)

2. Start the database service “APXRAC”

   srvctl start service -d APX -s APXRAC

3. Verify the status of the service on the database

   srvctl status service -d APX

   Service APXRAC is running on instance(s) APX1, APX2

4. Query the Database for the service_id associated with the newly created service

   SQL> select name,service_id from dba_services where name = 'APXRAC';

   ```
   NAME          SERVICE_ID
   ---------------  ----------
   APXRAC                9
   ```

5. Verfify the RAC configuration entries associated with the service “APXRAC”

   select SERVICE_ID, NAME, FAILOVER_METHOD, FAILOVER_TYPE, CLB_GOAL from dba_services where name = 'APXRAC';

   ```
   SERVICE_ID  NAME  FAILOVER_METHOD  FAILOVER_TYPE  CLB_G
   ----------        ---------- -------------------        --------------------  -----
         9           APXRAC                                                    LONG
   ```

6. Modify the service “APXRAC” to set the failover attributes for server side TAF policies.

   ```
   SQL> execute dbms_service.modify_service (service_name => 'APXRAC'
   , goal => DBMS_SERVICE.GOAL_SERVICE_TIME
   , clb_goal => dbms_service.CLB_GOAL_SHORT
   , failover_method => dbms_service.FAILOVER_METHOD_BASIC
   ```

```
, failover_type => dbms_service.FAILOVER_TYPE_SESSION
, failover_retries => 3
, failover_delay => 5
, aq_ha_notifications => true );

PL/SQL procedure successfully completed.
```

7. Verify the updated service attributes

```
SQL> select SERVICE_ID, NAME, FAILOVER_METHOD, FAILOVER_TYPE,
CLB_GOAL from dba_services  where name = 'APXRAC';

SERVICE_ID NAME  FAILOVER_METHOD   FAILOVER_TYPE      CLB_G
---------- --------  ----------------------  -------------------  -----
         9 APXRAC  BASIC                   SESSION              SHORT
```

Note: The above is applicable only for server-side configuration. To enable clinet-side settings, please create/modify the relavent tnsnames.ora entry on the Database nodes.

Transparent Application Failover

The Transparent Application Failover (TAF) feature of Oracle Net Services is a runtime failover for high-availability environments. It enables client applications to automatically reconnect to the database if the connection fails and, optionally, resume a SELECT statement that was in progress. The reconnection happens automatically from within the Oracle Call Interface (OCI) library. For applications that do insert, update or delete transactions, the application must trap the error when the failure occurs, rollback the transaction, and then resubmit. If the application is not written to be TAF aware, the session will get disconnected.

The TAF policy can be configured using the database service, thus eliminating the need to change all your tnsnames.ora files on your clients. Fast Application Notification events propagated by the Oracle RAC are captured by TAF.

Note: To know more about using Transparent Application Failover in Oracle Database 10g RAC, refer to the following link:

http://www.oracle.com/technology/obe/10gr2_db_vmware/ha/rac/rac.htm#t2

mod_plsql Extension

The mod_plsql extension maintains a pool of connections to the database. Established database connections are reused for subsequent requests. If there is no response from a database connection in a connection pool, mod_plsql detects this, discards the dead connection, and creates a fresh database connection for subsequent requests.

The dead database connection detection feature of mod_plsql eliminates the occurrence of random errors when a database node or instance goes down. This

feature is extremely useful in high availability configurations like Oracle Real Application Cluster (RAC). If a node in an Oracle RAC cluster has gone down, mod_plsql detects this and immediately starts servicing requests using the other Oracle RAC nodes.

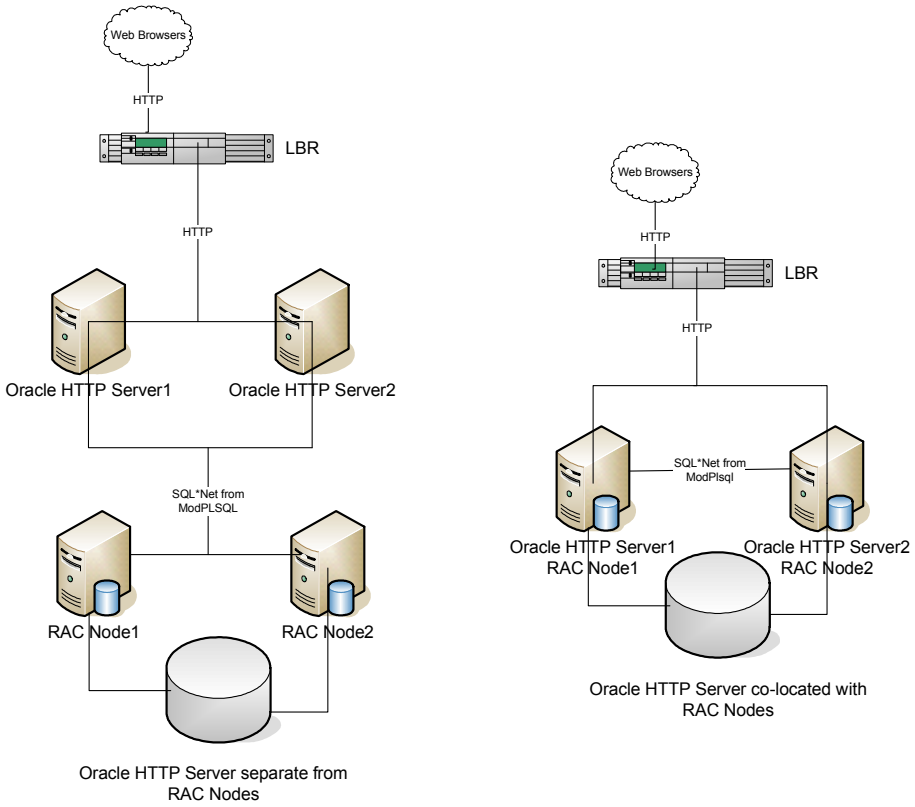
By default, when an Oracle RAC node or database instance goes down and mod_plsql had previously pooled connections to the node, the first mod_plsql request which uses a dead connection in its pool will result in a failure response of HTTP-503 being sent back to the end-user. This failure is then used by mod_plsql to trigger the detection and removal of all dead connections in its pool. mod_plsql pings all connection pools that were created before the node failure, when processing the next request that uses a pooled connection. If the ping operation fails, the database connection is discarded, and a new connection is created and processed.

There is no configuration or programmatic changes necessary for our environment as this extension is shipped with the Oracle HTTP Server.

Oracle HTTP Server

For the purpose of this paper, we will only discuss configuring the DAD with the Oracle HTTP Server. The paper does not intend to discuss steps to install and configure the Middle-Tier for high availability but will give an overview of configuration considerations. To know more about configuring the Oracle HTTP Server for high availability please references the Oracle Application Server High Availability Guide http://download.oracle.com/docs/cd/B25221_04/core.1013/b15977/framewk.htm and the Oracle Maximum Availability Architecture White Paper http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP.pdf.

You may choose to install your Oracle HTTP Server on either one of the RAC Nodes, or on separate hardware. In order to provide a scalable and highly available environment to complement the Oracle RAC environment, Oracle HTTP Server must also be configured for High-Availably. This is typically accomplished by installing two or more Oracle HTTP Servers on two or more physical servers (either the same servers as the database or separate servers) with traffic load balanced between the nodes by a load balancing router (LBR).



An example of a typical entry for Oracle Application Express in the `dads.conf` file for a single instance database is as follows:

```
<Location /pls/apex>
        Order                           deny, allow
        PlsqlDocumentPath               docs
        AllowOverride                   None
        PlsqlDocumentProcedure          wwv_flow_file_mgr.process_download
        PlsqlDatabaseConnectString      localhost:1523:orcl ServiceNameFormat
        PlsqlNLSLanguage                AMERICAN_AMERICA.AL32UTF8
        PlsqlAuthenticationMode         Basic
        SetHandler                      pls_handler
        PlsqlDocumentTablename          wwv_flow_file_objects$
        PlsqlDatabaseUsername           APEX_PUBLIC_USER
        PlsqlDefaultPage                apex
        PlsqlDatabasePassword           welcome1
       PlsqlRequestValidationFunction wwv_flow_epg_include_modules.authorize
       Allow from all
</Location>
```

Note: Although Oracle Application Express can be configured with the Oracle XML DB HTTP Server, it is not recommended in a High-Availability environment.

The *PlsqlDatabaseConnectString* attribute is critical in determining the way we connect to our database. In a single instance database we would typically use the "*ServiceNameFormat*" (ex. localhost:1521:orcl). However, in the case of a RAC environment where we have 2 or more instances that connect to a Database, we can use either the *TNSFormat* or the *NetServiceNameFormat* to determine the value for the *PlsqlDatabaseConnectString* attribute.

To configure the DAD with the *TNSFormat* entry, we can give the Connect Description entry from the tnsname.ora.

The following is the connect string descriptor in our tnsnames.ora configuration file:

```
APXRAC =  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = stnsp001-vip)(PORT = 1521))
   (ADDRESS = (PROTOCOL = TCP)(HOST = stnsp002-vip)(PORT = 1521))
   (LOAD_BALANCE = yes)
   (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = apxrac.us.oracle.com))  )
```

Note: It is encouraged that the Virtual IP of the respective Oracle RAC instances be specified for the HOST connection in the tnsnames.ora file. The Oracle Virtural IP is used to mitigate TCP/IP timeout delays on client connections

1. Stop the Oracle HTTP server

   ```
   Host$> opmnctl stopproc ias-component=HTTP_Server
   ```

2. Make the change to the PlsqlDatabaseConnectString Attribute

   ```
   PlsqlDatabaseConnectString  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = stnsp001-vip)(PORT = 1521))
   (ADDRESS = (PROTOCOL = TCP)(HOST = stnsp002-vip)(PORT = 1521))
      (LOAD_BALANCE = yes)
      (CONNECT_DATA =
        (SERVER = DEDICATED)
        (SERVICE_NAME = apxrac.us.oracle.com) )  ) TNSFormat
   ```

3. Save the changes to the dads.conf file and restart the Oracle HTTP Server

   ```
   Host$> opmnctl startproc ias-component=HTTP_Server
   ```

4. Connect to your APEX instance to verify the dads.conf entry

5. If using Oracle Application Server with a repository based middle tier, update the repository configuration
   Host$> $ORACLE_HOME/dcm/bin/dcmctl updateconfig

Although this connect string format for the *PlsqlDatabaseConnectString* is acceptable, it is not recommended in a HA environment. Instead, the

PlsqlDatabaseConnectString should use the *NetServiceNameFormat* connect string. This enables the name resolution through an LDAP lookup of the Oracle Internet Directory (OID). Further, OID provides a central repository to configure the database information and makes it easier to add or remove RAC nodes during maintenance. This is discussed later in the document.

In addition to configuring the dads.conf entries required to allow each HTTP Server to communicate with each RAC node, the Oracle HTTP Servers should be configured to propagate the correct logical host name, for example www.mycompany.com. If you install the Oracle HTTP Server on machines named myhost1.mycompany.com and myhost2.mycompany.com the Oracle HTTP Servers will install with those names. Adding a VirtualHost entry with the httpd.conf file of each Oracle HTTP Server installation will accomplish this. For each Oracle HTTP Server do the following:

1. Stop the Oracle HTTP server

   Host$> opmnctl stopproc ias-component=HTTP_Server

2. Add the following lines to the bottom of httpd.conf file, replacing 80 with the port your load balancer is listening on and www.mycompany.com with the logical name of your site.

   PlsqlDatabaseConnectString (DESCRIPTION =
   NameVirtualHost *:80

   <VirtualHost *:80>

   ServerName www.mycompany.com

   ServerAlias mycompany.com

   </VirtualHost>

3. Save the changes to the httpd.conf file and restart the Oracle HTTP Server

   Host$> opmnctl startproc ias-component=HTTP_Server

4. If using Oracle Application Server with a repository based middle tier, update the repository configuration
   Host$> $ORACLE_HOME/dcm/bin/dcmctl updateconfig

Finally, the load balancing router can be configured to ensure that all aspects of the High Availability environment are running. In this scenario we only want the LBR to forward traffic to Oracle HTTP Servers that can communicate with the Application Express database. By default most LBR's will either ping a node or rely on timeouts as a heartbeat mechanism. Configuring the LBR's heartbeat mechanism to point to the url associated with the mod_plsql DAD will ensure that the server is responding, the Oracle HTTP Server is running and can communicate with the Application Express database. Please see your LBR manual to configure the heartbeat for /pls/apex/apex.

Oracle Internet Directory (OID)

OID is a component of Oracle Identity Management Suite. The Identity Management Suite provides an integrated infrastructure that provides distributed security and management services for Oracle products and other business applications. The OID and provisioning component enables synchronization between OID and the automatic provisioning services for Oracle components and applications. OID utilizes the LDAP standard to simplify the management of directory information.

With OID, users and applications within an organization can leverage a single, well-defined, standard interface to a single, extensible directory service. This makes it easier to rapidly develop and deploy directory-enabled applications. It further reduces the need to enter and coordinate redundant information in multiple services that would otherwise be scattered across an organization.

Note: The Oracle Internet Directory is a critical component in an enterprise and deployment should take failure recovery and high availability into consideration. To know more on High Availability considerations for Oracle Internet Director, Please refer to the following link http://download.oracle.com/docs/cd/B28196_01/idmanage.1014/b15991/deploy.htm#i386129 **from the Administration Guide.**

Registering the Database with Oracle Internet Directory (OID)

OID provides a well-defined protocol and array of programmatic interfaces making it ideal for deploying Internet-ready applications that leverage the directory. In addition to using OID to resolve the Oracle RAC Database services, developers can utilize LDAP based authentication for applications built using Oracle Application Express. To know more about using Oracle LDAP Authentication with Oracle APEX, please refer to the users guide.

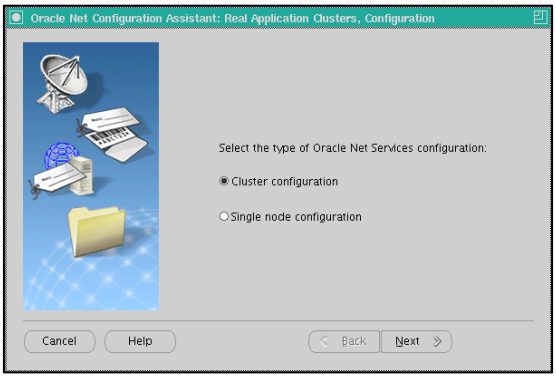
1. Configure the "Directory Usage". A directory server provides a centralized tool for managing and configuring a distributed Oracle network. The directory server can replace client-side and server-side localized tnsnames.ora files.

    a. Connect to one of the Oracle RAC nodes

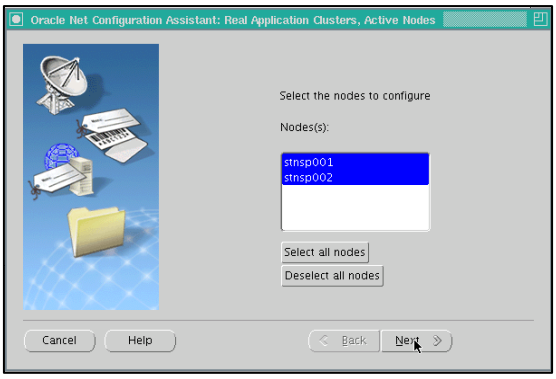
    b. Start the Oracle Net Configuration Assistant

    Host$> $ORACLE_HOME/bin/./netca

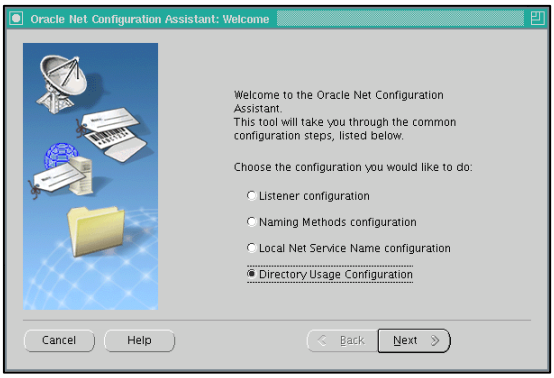
c. Select “Cluster Configuration”



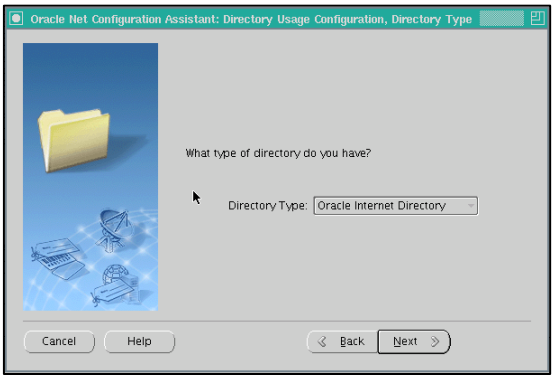
d. Select all the RAC nodes



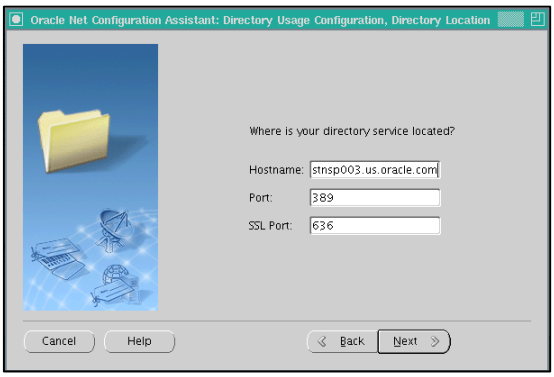
e. Select Directory Usage Configuration



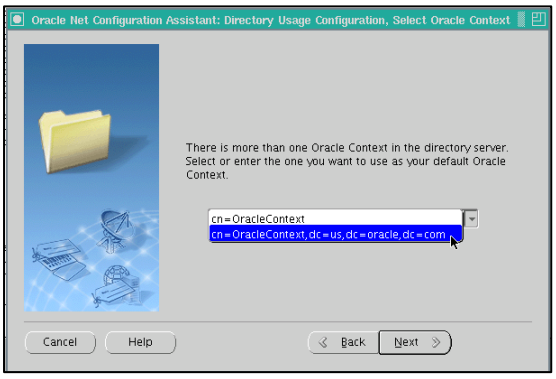
f. Select the Directory Type



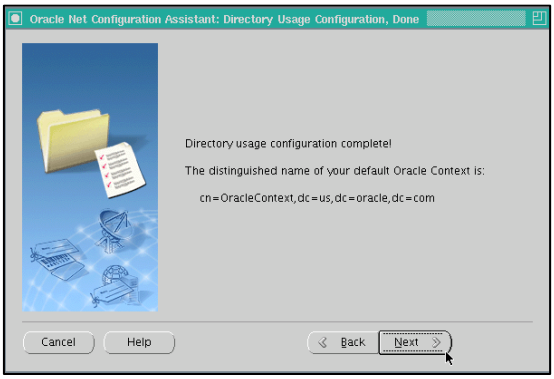
g. Identify the Oracle Internet Directory instance



h. Identify the default Oracle Context



i. Confirm the DN (Distinguished Name) for the Oracle Context

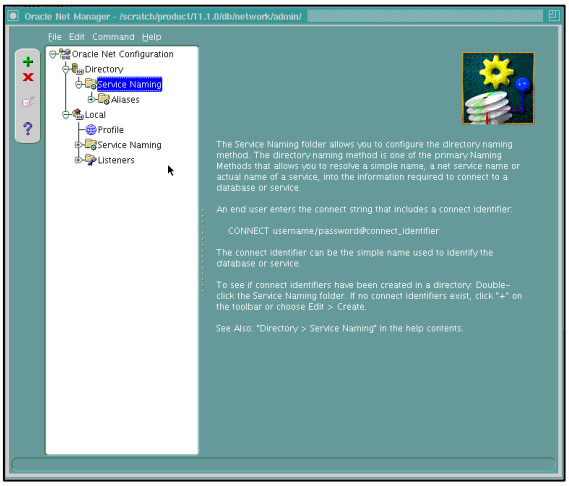


2. Add a database net service to the Directory Server

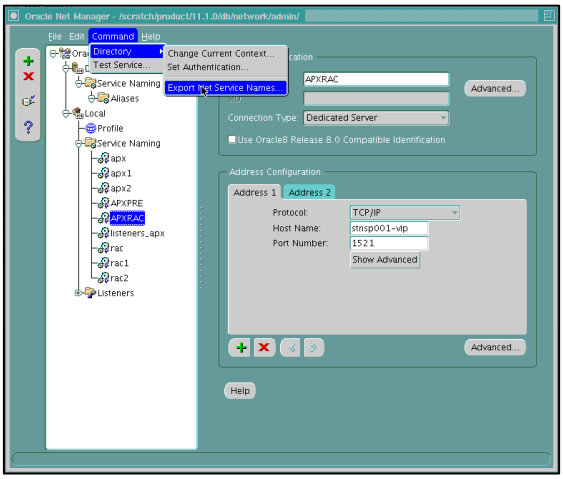
a. Connect to one of the Oracle Database RAC nodes

b. Start the Oracle Net Manager

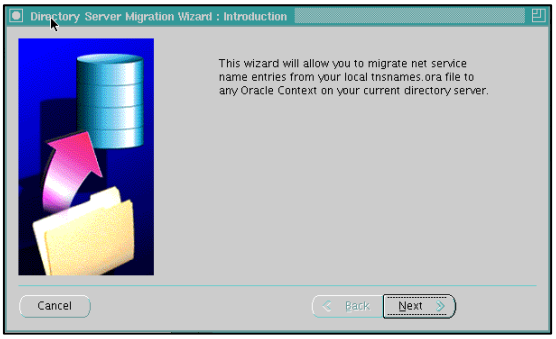
Host$> $ORACLE_HOME/bin/netmgr



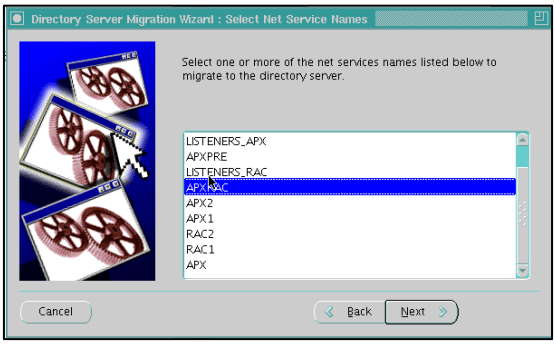
c. Export the Database Service to OID



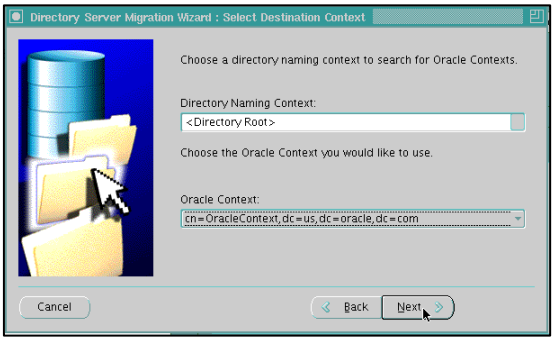
d. Select Next and proceed with identifying a database service to be migrated



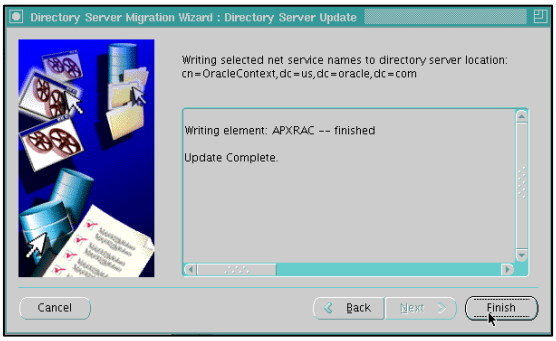
e. Identify the database service to be exported, for example APXRAC



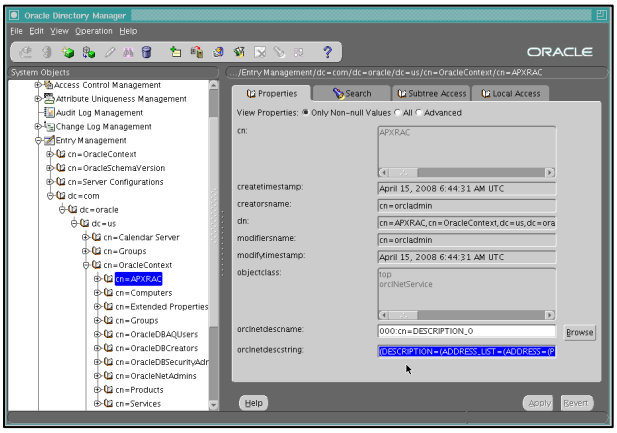
f. Identify the Directory naming context



g. Finish the Directory service

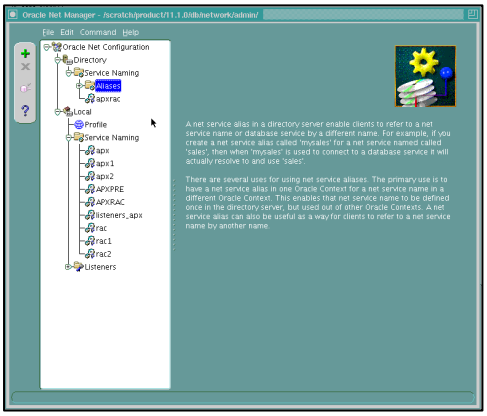


h. Verify the Database Services in OID by navigating to the corresponding DN entry under “Entry Management”. Also, verify the corresponding “orclnetdescstring” entry.

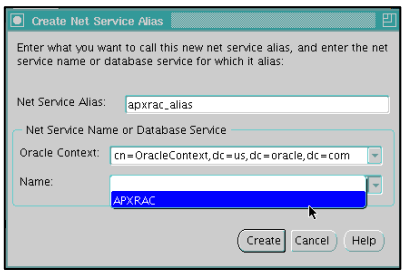


3. Add a Database Net Service Alias to the directory server. The alias, for a net service name or database service defined with the directory server, only references the location of the object for which it is an alias. When a client requests a directory lookup of a net service alias, the directory determines that the entry is a net service alias and completes the lookup as if it was actually the entry it is referencing

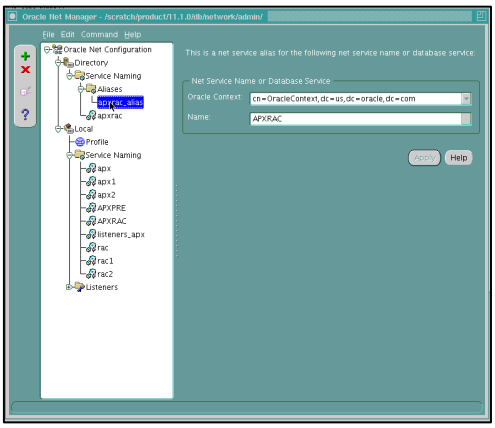
   a. Via one of the Oracle RAC Nodes, use the Net Manager to create the alias



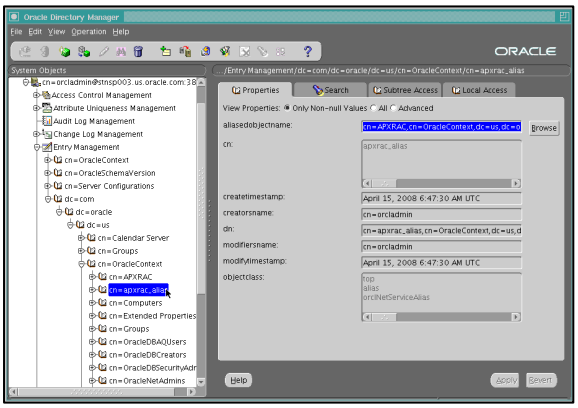
   b. Identify the Net Service for which the alias is to be created



c. Create the Net Service Alias



d. Verify the Database Services in OID by navigating to the corresponding DN entry under "Entry Management". Also, verify the corresponding "orclnetdescstring" entry.



4. Create a DAD entry for the OID HTTP Server

a. If you are using the "Net Service", your DAD's entry and the PlsqlDatabaseConnectString entry would be as follows:

<Location /pls/apex>

Order	deny,allow
PlsqlDocumentPath	docs
AllowOverride	None
PlsqlDocumentProcedure	wwv_flow_file_mgr.process_download
PlsqlDatabaseConnectString NetServiceNameFormat	cn=apxrac,cn=OracleContext,dc=us,dc=oracle,dc=com
PlsqlNLSLanguage	AMERICAN_AMERICA.AL32UTF8
PlsqlAuthenticationMode	Basic
SetHandler	pls_handler

```
        PlsqlDocumentTablename      wwv_flow_file_objects$
        PlsqlDatabaseUsername       APEX_PUBLIC_USER
        PlsqlDefaultPage            apex
        PlsqlDatabasePassword       welcome1
</Location>
```

b. If you are using the "Net Service Alias", replace the *PlsqlDatabaseConnectString* in the DAD's with the distinguishing name associated with the alias. For example:

```
PlsqlDatabaseConnectString
cn=apxrac_alias,cn=OracleContext,dc=us,dc=oracle,dc=com
NetServiceNameFormat
```

5. Save the changes to the dads.conf file and restart the Oracle HTTP Server

```
Host$> opmnctl startproc ias-component=HTTP_Server
```

6. Verify the configuration by connecting to the Oracle Application Express Instance

7. If using Oracle Application Server with a repository based middle tier, update the repository configuration
   Host$> $ORACLE_HOME/dcm/bin/dcmctl updateconfig

Configuring Multiple HTTP Servers on Oracle RAC

While an Oracle RAC database reduces the risk of failure due to loss of service, a single Oracle HTTP Server, a critical component that enables processing of client requests to the Oracle APEX engine, is a potential single point of failure.

Leveraging the existing Oracle Clusterware that manages our Oracle RAC Database, the Oracle HTTP Server can be installed and configured onto the respective Database nodes within our cluster. The Oracle Clusterware can be configured to monitor the HTTP Server resource:

1. Copy the following act_http.pl action script to the $CLUSTERWARE_HOME/crs/public directory on all cluster nodes

   Please refer to the Appendix for the sample script.

2. Test the http action script before creating the http resource. With the http server running verify the script will stop the http server:

   ```
   $CLUSTERWARE_HOME/crs/public/act_http.pl stop
   ```

3. With the http server down, verify the script will check, and start the http server if it is not running:

   ```
   $CLUSTERWARE_HOME/crs/public/act_http.pl check
   ```

   If the action script is correct in your environment the act_http.pl action will start the http server automatically. If the action script is able to check and start the http server

resource, the Clusterware resource is ready to be created. If the http server is not started, DO NOT create the http resource in the clusterware.

4. Create the http resources for each http server node of the cluster

   crs_profile -create http1 -t application –a /scratch/11.1.0/crs/crs/public/act_http.pl -o ci=20,ra=5

   crs_profile -create http2 -t application -a /scratch/11.1.0/crs/crs/public/act_http.pl -o ci=20,ra=5

5. Register and start the http resource on each of the cluster nodes.

   crs_register http1

   crs_start -c stnsp001 http1

   crs_register http2

   crs_start -c stnsp002 http2

Note: An Oracle single instance database can be added in the Oracle RAC cluster, and the cluster could be configured to monitor these resources in the same way as the http server. The Using Oracle Clusterware to Protect A Single Instance Oracle Database 11g white paper is on OTN: http://www.oracle.com/technology/products/database/clusterware/pdf/SI_DB_Failover_11g.pdf

Configuring Oracle Embedded PL/SQL Gateway

The Oracle Embedded PL/SQL Gateway (EPG), installed with the 11g Database, is an alternate to the Oracle HTTP Server Configuration. It provides the Oracle database with a Web server and the necessary infrastructure to create dynamic web applications. The EPG runs in the Oracle XML DB HTTP server in the Oracle database and includes the core features of mod_plsql.

To configure the embedded PL/SQL gateway:

1. Connect to one of the RAC Nodes

2. Run apex_epg_config.sql passing the file system path to the base directory where the Oracle Application Express software was unzipped

3. Start SQL*Plus and connect to the database where Oracle Application Express is installed as SYS specifying the SYSDBA role:

   $ sqlplus /nolog

   connect sys as sysdba

   When prompted, enter the appropriate password.

4. Run apex_epg_config.sql script:

   @apex_epg_config /tmp

5. Enter the following statement to unlock the ANONYMOUS account:

   ALTER USER ANONYMOUS ACCOUNT UNLOCK;

6. Verify the port number where the Oracle XML DB HTTP Server is running:

   SELECT DBMS_XDB.GETHTTPPORT FROM DUAL;

   If the port number returns 0, the Oracle XML DB HTTP Server is disabled.

7. To enable the Oracle XML DB HTTP Server, enter the following

EXEC DBMS_XDB.SETHTTPPORT(port);

For example:

EXEC DBMS_XDB.SETHTTPPORT(8080);

Note: Port numbers less than 1024 are reserved for use by privileged processes on many operating systems

In an Oracle RAC environment, the above configuration steps would have to be configured on all the Database cluster nodes.

CONCLUSION

Combining the inherent flexibility and rapid application development environment provide by Oracle Application Express, with the high availability and scalability offered by Oracle Real Application Clusters, businesses can ensure high returns on their IT investments. This potent combination allows users to quickly build, modify and deploy their Applications with continued access to mission critical data while drawing benefit from a robust and transparent infrastructure provided by Oracle RAC.

APPENDIX

```
----------Script Begins ----------
#!/usr/bin/perl
# $Header: act_http.pl 18-mar-2008$
# act_http.pl
# Copyright (c) 2008, Oracle. All rights reserved.
# NAME
# act_http.pl - action script for the http server resource
# DESCRIPTION
# This perl script is the action script for start / stop / check
# the Oracle HTTP Server in a restart configuration.
# NOTES
# Edit the perl installation directory as appropriate.
# Place this file in <CLUSTERWARE_HOME>/crs/public/
# MODIFIED (MM/DD/YY)
# dismith 03/18/08 - Creation
# The ORACLE_HTTP_SERVER_HOME must be set to point to the HTTP Server Home directory
$ORACLE_HTTP_SERVER_HOME = "/scratch/product/11.1.0/ohs";  #This should point to the
ORACLE_HTTP_SERVER_HOME
if ($#ARGV != 0 ) {
print "usage: start stop check required \n";
exit;
}
$command = $ARGV[0];
# start http server
if ($command eq "start") {
system ("
$ORACLE_HTTP_SERVER_HOME/opmn/bin/opmnctl startall");
}
# stop http server
if ($command eq "stop") {
system ("
$ORACLE_HTTP_SERVER_HOME/opmn/bin/opmnctl stopall");
}
# check http server
if ($command eq "check") {
check_http();
}
sub check_http {
my($check_proc_http,$process_http) = @_;
$process_http = "$ORACLE_HTTP_SERVER_HOME/Apache/Apache/bin/httpd";
$check_proc_http = qx(ps -ef | grep $process_http | grep -v grep | wc -l);
if ($check_proc_http gt 1) {
exit 0;
} else {
system ("$ORACLE_HTTP_SERVER_HOME/opmn/bin/opmnctl startall");
}
}
----------Script End----------
```

A sample script is posted on the Oracle Real Application Clusters Sample code page on Oracle's Technology Network (otn.oracle.com).

Oracle Application Express and
Oracle Real Application Clusters
June 2008
Author: Amitabh Chhibber
Contributing Authors: Duane Smith, Anton Nielsen

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
www.oracle.com