Oracle9*i* Application Server Infrastructure: Improved Availability with Hardware Clusters

An Oracle White Paper
Updated: August 2003

1. Introduction to Oracle9*i*AS Infrastructure

Oracle9*i*AS Release 2 (version 9.0.2) introduced a new component called "Oracle9*i*AS Infrastructure". This infrastructure component provides centralized product metadata, security and management services, configuration information, and data repositories for middle tier installations. The middle tier instances use the infrastructure for the following main services

Product Metadata Service: All product metadata required by the middle tier is bundled as part of the infrastructure.

Security Service: This provides a single source of security metadata containing all administration and user privileges for all Oracle9*i*AS applications.

Management Service: Management Service is used by Oracle Enterprise Manager to manage the middle tier as well as the Infrastructure itself.

For more information on the Infrastructure and how the various components interact with each other and the middle tier refer to the Oracle9*i*AS Infrastructure paper at *http://otn.oracle.com/products/ias/pdf/9ias_infra.pdf*.

Since Oracle9*i*AS Infrastructure provides centralized services to the middle tiers, it becomes very important to guarantee that the infrastructure is always available and accessible by the middle tiers. This paper will explain how to setup Oracle9*i*AS Infrastructure on a hardware cluster to make it highly available.

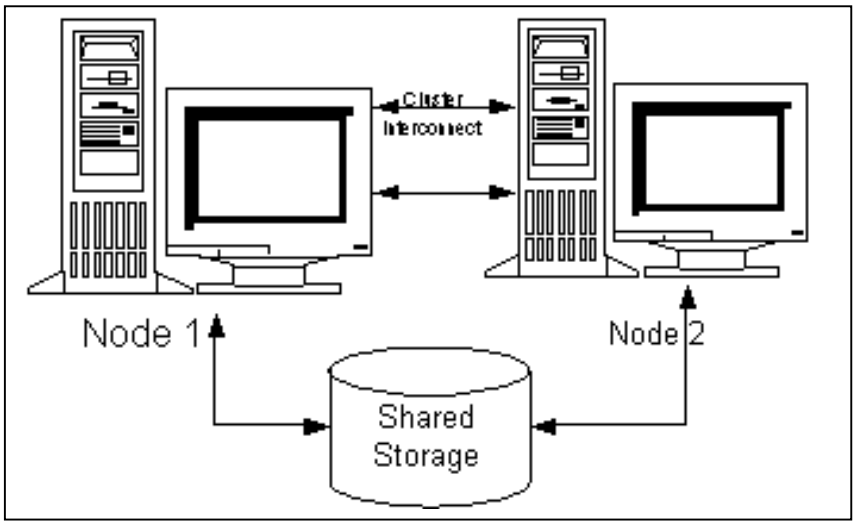
This paper divides the services provided by Oracle9*i*AS Infrastructure into the following components that can be started or stopped as one unit.

1. Metadata repository
2. Oracle Net Listener
3. Oracle Internet Directory Monitor
4. Oracle Internet Directory Process
5. Oracle HTTP Server
6. Delegated Administrative Services
7. Enterprise Manager Website
8. Oracle Management Server (optional)

The Oracle9*i*AS Infrastructure is considered available if all of the above non-optional components are up and running. While some of the individual processes like Oracle HTTP Server are monitored and restarted automatically by the Infrastructure framework, for other errors like an operating system failure or a node failure there needs to be a mechanism to restart or failover the Infrastructure quickly.

This paper explains the Cold Failover solution using hardware clusters to make the Infrastructure highly available.

2. Two-Node hardware cluster setup



A simple hardware cluster consists of two identical nodes attached to a shared storage device. While hardware clusters can have more than 2 nodes, the solution described in this paper is based on a 2-node cluster. Shared storage in this case could be a dual ported disk, addressable from both nodes. It is highly recommended that this storage be mirrored to provide protection from media failures. Most hardware vendors will provide storage with hardware or software mirroring. The simple configuration above allows applications to run on either node, with equal access to the shared storage. One of the nodes runs the application at a given time (i.e. is hot) while the other node is passive (i.e. is cold, hence the term cold failover) waiting to take over should the active node fail. This approach of having a redundant node ready for takeover can protect the application from any system failures. Stopping the application on the one node (the active node) and starting it on the other (passive node) is called failover. This could either be a planned operation where you stop all processes on the active node and restart them on the passive node or it could be an unplanned outage which causes the active node to fail, thus requiring a restart of the application on the passive node. The restart can be done manually or be automated through a cluster service. The application can failover a second time, often referred to as failback since it brings the application back to its original node. There is however no fundamental difference between a failover and failback.

In addition to being able to failover and failback the application on a hardware cluster, it is also important that clients accessing this application be able to connect transparently, regardless of the node on which the application is running. This can be achieved by having a virtual hostname for the nodes in the cluster. The active node in the cluster enables a logical IP on the same public network interface as the nodes physical IP. This logical IP is mapped to a virtual hostname. Clients access the application via the virtual hostname. When the application fails over to the second node, the logical IP is disabled on the first node and enabled on the second node. Clients accessing the virtual name then transparently get routed to the second node.

The solution described in the previous version of this paper required changing the physical hostname of the node to map to the virtual name. This paper uses an alternate technique called Library Interpositioning to eliminate the need to change the physical hostname. Library Interpositioning works by telling the dynamic linker (ld.so.1) to load a certain library before loading the system libraries. This paper uses this technique to interpose the gethostname() call for Oracle9*i*AS Infrastructure so that the call returns the virtual hostname. This way, the Oracle9*i*AS Infrastructure components will function correctly on any node of the cluster using the virtual hostname without affecting other applications. Note that interpositioning only works within the shell that sets the environment variable LD_PRELOAD. Hence there is no impact on the system outside the shell that sets LD_PRELOAD.

Operating system vendors provide clustering software that runs on top of this hardware cluster setup and provides the monitoring facilities necessary to detect failures on the active node and take appropriate action to restart the failed services on the passive node. Since the starting/stopping/monitoring of application processes is unique to each application, this often requires some programming or scripting to tailor the automatic failure detection and failover of your particular application. Most vendors provide pre-packaged services for well-known applications like Oracle Database, NFS etc.

The rest of the paper will provide an example of how to setup an HA configuration for Oracle9*i*AS Infrastructure.

This paper provides an explanation and the basic manual steps necessary to run the Oracle9*i*AS Infrastructure on a hardware cluster. The examples provided in the paper are intended to relay the concept of this implementation. Customers can use these steps as a guideline, but should consult their appropriate vendors as needed. Basic failover testing has been performed using this configuration, however customers must perform their own series of functional and fault testing as part of the implementation. The examples in this paper were developed and tested in a Sun Solaris, Sun Cluster 3.0 environment.

3. Pre-Install Setup

Before performing the install, prepare the required information and environment by doing the following:

1. Consult the included README file
2. Request Logical IP address
3. Request Virtual Hostname
4. Add a DNS Entry
5. Add a Local Entry
6. Setup the File System
7. Stop the cluster software
8. Enable the virtual hostname setup on the first node
9. Enable the virtual hostname setup on the second node (Repeat step 7 on the second node)

Each of these steps is described in detail below

1. Consult the included README file

The zip (or tar) file you download contains a README file. Please consult that file for instructions in addition to the ones listed in this paper.

2. Request Logical IP address

This can be any valid IP address for the subnet that has the 2-node cluster. Note that there is no difference between the logical IP address and the physical IP address of the machine, except in the way they are used. The physical IP address is always tied to a physical host, whereas the logical IP address can move from one machine to another. Please check with your network administrators for a valid unused IP address that can be used for the 2 machines. Make sure that this IP address is then reserved for you and is not inadvertently re-used as a physical or logical IP address on your network.

3. Request Virtual Hostname

Most applications will access the 2 nodes via a hostname, not the IP address, so even though the IP address is logical, it still needs to be mapped to a hostname. Check with your network administrator for a valid hostname for your organization and pick a valid hostname that can be used as a virtual hostname.

4. Add a DNS entry for virtual hostname

Once you are assigned a logical IP address and virtual hostname, add the entry to your corporate DNS, so all hosts can access the Infrastructure via the virtual hostname.

5. Add a Local Entry for virtual host

Also add the virtual hostname and the logical IP address to your local /etc/hosts file

Logical IP <Fully Qualified virtual hostname> <virtual hostname>

```
144.25.245.196  infrahost.oracle.com  infrahost
where 144.25.245.196 is the logical IP address
      infrahost.oracle.com is the virtual hostname
```

Add the *same* virtual hostname and associated IP address in /etc/hosts on the other node.

6. Setup File System

The cluster setup already has shared storage but you need to create a file system on this storage to house the Infrastructure software and the metadata repository, such that any node can mount the file system. If you are running some kind of volume manager on the system to manage the shared storage, please refer to the volume manager documentation for steps to create a volume. Once the volume is created you can create the file system on that volume.

If you don't have a volume manager, you can create a file system on the shared disk directly but check that the vendor supports this, that the file system can be mounted from either node, and that the file system is repairable (able to do fsck on the file system) from either node, in case of a crash.

Example using Veritas Volume Manager on Sun Solaris 8:

```
# vxdg -C import <Veritas Diskgroup Name>
# vxvol -g <Veritas Diskgroup Name> startall
# mount /dev/vx/dsk/<Veritas Diskgroup Name>/<volume_name>
<mount-point>
# vxdg -C import infra-dg
# vxvol -g infra-dg startall
# mount /dev/vx/dsk/infra-dg/infra_vol /iasv2_infra
- infra-dg is the Veritas diskgroup
- infra_vol is a volume big enough to fit the
Infrastructure install
- /iasv2_infra is the file system created on the
volume
```

Note that only one node will have the file system mounted at given anytime. Alternatively, if this setup will be used to build a failover dataservice then the file system can also be a Sun Cluster Global File System or a file system managed by SUNW.hastorageplus resource type (requires Sun Cluster 3.0 05/02 update)

7. Stop the cluster software

Oracle9*i*AS version 9.0.2.x does not support installation on a hardware cluster. So for the duration of the install and the post install configuration the clusterware must be stopped. You should have access to the console before the next step .

```
# scshutdown -g 0 y   (from one node for SC3.0)
          this will halt the 2 nodes leaving them in the boot  (OK) prompt,
where  they can be booted in non-cluster mode
```

On the console for the 2 nodes you will have the boot prompt. Type "boot –x" on both the nodes.

OK > boot -x

to boot in exclusive mode

8. Enable the virtual hostname setup on the first node

The accompanying zip file contains the following files to setup the virtual hostname, under the setup sub-directory

libloghost.so.1_32 – 32 bit library for virtual hostname setup
libloghost.so.1_64 – 64 bit library for virtual hostname setup
hostname.exe - C executable to validate virtual hostname setup
hostname.class – Java class to validate virtual hostname setup
iASVrtlHst.c – Source corresponding to the shared libraries
hostname.c - Source for hostname.exe
hostname.java – Source for hostname.class
Makefile – makefile for generating 32 bit shared object
Makefile.64 – makefile for generating 64 bit shared object

As mentioned in the earlier section, this solution provides a mechanism for Oracle9*i*AS Infrastructure components to see a virtual hostname. This functionality is provided by shared library libloghost.so.1 and is explained later.
Copy the shared libraries to the appropriate location and create the symbolic links.

The commands below describe this setup:

```
# cp libloghost.so.1_32  /usr/lib/libloghost.so.1
# cp libloghost.so.1_64 /usr/lib/sparcv9/libloghost.so.1
# cd /usr/lib/secure
# ln –s /usr/lib/libloghost.so.1 libloghost.so.1
# cd /usr/lib/secure/sparcv9
# ln –s /usr/lib/sparcv9/libloghost.so.1 libloghost.so.1
```

Note : It is necessary to have the library in a secure location for some processes like oidldapd that are setuid root and hence will not load the interposed library from an insecure location. However, the only function in this library is gethostname(), which is not believed to pose any threats if other applications on the system use the interposed library.

Prepare for Oracle9*i*AS install. This requires enabling the logical IP on the node where the install will be run.

To enable the logical IP address, run the following command

```
# ifconfig <interface> addif <logical-ip>
```

Example -

```
# ifconfig hme0 addif 144.25.245.196 up
```

where hme0 is the public interface on which the logical IP address is enabled. This is the interface that typically has your physical host ip also.

Before starting the install verify that the logical IP and virtual hostname is setup. To verify this the zip file also provides two files

C program – hostname.exe

Java class file hostname.class.

Run the C program and Java program as the user who would do the Oracle9iAS software install

```
$ hostname.exe
$ java hostname
```

The two programs should return the physical hostname of the host.

Now setup the following environment variables for the user

```
$ LD_PRELOAD=libloghost.so.1
$ LHOSTNAME=<virtual_hostname>
$ export LD_PRELOAD LHOSTNAME
```

Note : the virtual name set in LHOSTNAME is the short name not the fully qualified name. For example if your /etc/hosts entry looks like –

144.25.245.196 infrahost infrahost.mycompany.com

then LHOSTNAME must be set to infrahost

As mentioned earlier the library libloghost.so.1 provides the functionality that enables the Oracle9iAS components to see the virtual hostname. This is done by redefining the gethostname() call, so that when the LHOSTNAME variable is set, the hostname returned by this call to the caller is the virtual hostname. The gethostname() function in this shared library is only called if the LD_PRELOAD variable is set.

Rerun the two programs again to verify that both C and Java code will return the virtual hostname.

$ hostname.exe

$ java hostname

If the setup of your virtual hostname is correct (entry in /etc/hosts, libraries in proper places, symbolic links to the libraries, LD_PRELOAD and LHOSTNAME set properly) the two programs should return the virtual hostname. If not, then go back and make sure all the steps for setting up the virtual hostname is correct before proceeding to the Oracle9iAS Infrastructure install.

9. Enable virtual hostname setup on the second node

Disable the logical IP on the first node and enable it on the second node as follows:

To disable the logical IP address run the following command

```
# ifconfig <interface> removeif <logical-ip>
```

Example -

```
# ifconfig hme0 removeif 144.25.245.196
```

Re-enable on the second node

```
# ifconfig hme0 addif 144.25.245.196 up
```

Now repeat step #7 on the second node and verify the virtual hostname setup is properly working. After verifying it, disable the logical IP address and virtual hostname on the second node, re-enable them on the first node and proceed with Oracle9*i*AS Infrastructure install.

4. Infrastructure Install and Setup

Before running the Oracle9*i*AS installer for installing infrastructure, cleanup the two nodes of a cluster from any prior attempt of an install. Remove any iasdb entry from oratab and remove the emtab file. These files are located under /var/opt/oracle.

The Install/Setup and validation of failover/failback for Cold Failover Cluster involves:

1. Consult the included README file
2. Install Oracle9*i*AS Infrastructure
3. Install any required patches
4. Edit opmn.xml
5. Stop Infrastructure on node 1
6. Prepare the second node of the cluster to run Infrastructure
7. Test Failover Steps

Each of the above steps is described below in detail.

1. Consult the included README file

The zip (or tar) file you download contains a README file. Please consult that file for instructions in addition to the ones listed in this paper.

2. Install Oracle9*i*AS Infrastructure

Install the Infrastructure on the node that you prepared for install in step 7 of the previous section. At the time of install, this node should have the logical IP address, virtual hostname and the shared file system mounted. Follow the instructions in the Oracle9*i*AS Infrastructure Installation Guide to install. When prompted for the ORACLE_HOME during the install , point to a directory on the shared file system. The hostname picked by the installer should be the virtual hostname.

After installing the software, the Oracle9*i*AS installer runs the configuration assistants for the various Oracle9*i*AS Infrastructure components. Prior to starting the config assistants, the installer will prompt to execute a script – root.sh. *Before* running the root.sh script, the following needs to be done in another window. This is because OHS uses configuration information defined in apachectl and does not inherit the values of variable (specifically, LD_PRELOAD and LHOSTNAME)

```
$ cd $ORACLE_HOME/Apache/Apache/bin

Add the following 3  lines to the CONFIGURATION section in
apachectl (you can add these 2 lines just before the PIDFILE= setting
in the file)

LD_PRELOAD=libloghost.so.1
LHOSTNAME=<virtual hostname>
export LHOSTNAME LD_PRELOAD

Note : the virtual name set in LHOSTNAME is the short name not
the fully qualified name. For example if your /etc/hosts entry looks
like –

144.25.245.196   infrahost infrahost.mycompany.com

then LHOSTNAME must be set to infrahost
```

3. Install Patches

Check the latest release notes for any post install patches. Instructions for applying the 9022 patch are in the included README file.

4. Edit opmn.xml

a) Stop opmn by issuing the following command:

```
$ORACLE_HOME/opmn/bin/opmnctl stopall
```

b) Edit the `$ORACLE_HOME/opmn/conf/opmn.xml` file and add entries for the LD_PRELOAD and LHOSTNAME environment variables in the environment section for OC4J_DAS, home, OC4J_Demos, CUSTOM, etc. For example:

```
<process-manager>
<ohs gid="HTTP Server" maxRetry="3">
<start-mode mode="ssl"/>
</ohs>
...
<oc4j instanceName="OC4J_DAS" gid="OC4J_DAS">
<config-file
path="/u02/oracle/AS/infra/j2ee/OC4J_DAS/config/server.xml"/>
<java-option value="-server -Xincgc -Xnoclassgc -Xmx256m "/>
<oc4j-option value="-properties"/>
<port ajp="3001-3100" rmi="3101-3200" jms="3201-3300"/>
<environment>
<prop name="DISPLAY" value="virtualhost:0.0"/>
<prop name="LD_LIBRARY_PATH" value="/u02/oracle/AS/infra/lib"/>
<prop name="LHOSTNAME" value="virtualhost.com"/>
<prop name="LD_PRELOAD" value="libloghost.so.1"/>
</environment>
</oc4j>
```

In the code above, the LD_LIBRARY_PATH value should be replaced by $ORACLE_HOME/lib of the Infrastructure install and "virtualhost.com" should be replaced with the fully qualified virtual hostname used for this solution; search for "environment" within opmn.xml for existing environment sections and search for "instanceName" to identify the beginning of each OC4J instance section.

c) Restart OPMN using the following command:

```
$ORACLE_HOME/opmn/bin/opmnctl startall
```

This OPMN command will startup two additional OPMN managed processes that are not typically started in a default install. You may choose to shutdown these two processes with the following commands:

```
$ORACLE_HOME/dcm/bin/dcmctl stop –co OC4J_Demos
```

```
$ORACLE_HOME/dcm/bin/dcmctl stop -co home
```

d) Update the configuration using the following command:

```
$ORACLE_HOME/dcm/bin/dcmctl updateConfig -ct opmn
```

5. Stop the Infrastructure process

After the install, all the Infrastructure processes will be started. Stop these processes and restart them once using the startup and shutdown steps described below in Appendices B and C below.

6. Prepare the second node

Since the ORACLE_HOME is on the shared storage , there is no need to install the software again on the second node. The file system can be unmounted from the first node and mounted on the second node to make the Infrastructure available to the second node. It is however necessary to update the oratab and the emtab (in /var/opt/oracle) files on the second node. If there are no other Oracle database instances on the 2 nodes, simply copy the 2 files from the first node to the second. If there are other Oracle database instances, then add the iasdb entries from the oratab and emtab file on node 1 to the same files on node 2. Create the files on node 2 if necessary.

7. Test Failover Steps

As explained earlier, failover means shutting the application on one node and starting it on the second node. Before you start installing the middle tier it is important to verify that you can manually failover and failback the Infrastructure.

In order to prepare for failover, manually stop all the Infrastructure processes on node 1 as described in Appendix C. The script described in Appendix C will halt all the Oracle9*i*AS Infrastructure processes and relinquish resources like file system, disk group and the logical IP address. Then restart the same processes on the other node by running the script described in Appendix B. Once you have verified that you can failover and failback the Infrastructure between the two nodes, shutdown the Infrastructure. Unmount the shared file system and disable the virtual name and logical IP. You can now boot the systems in cluster mode and then rerun the start script to start the Oracle9*i*AS Infrastructure components on one of the nodes of the cluster. It is important to note that only one of the two nodes can be running Oracle9*i*AS Infrastructure at any given time because they use the same virtual hostname and logical IP address, which can be active on one node at a time.

At this point you are ready to setup a Cluster Data Service so that the startup/shutdown and failover of the Infrastructure is handled by the cluster.

5. Middle Tier Install and Validation

After validating that the infrastructure can failover and failback between the 2 nodes of the cluster successfully, install your middle tier on a different node. Please refer to the appropriate middle tier install and setup guide for further details. When associating your middle tier with the Infrastructure, associate it with the logical IP address and the virtual hostname of the Infrastructure.

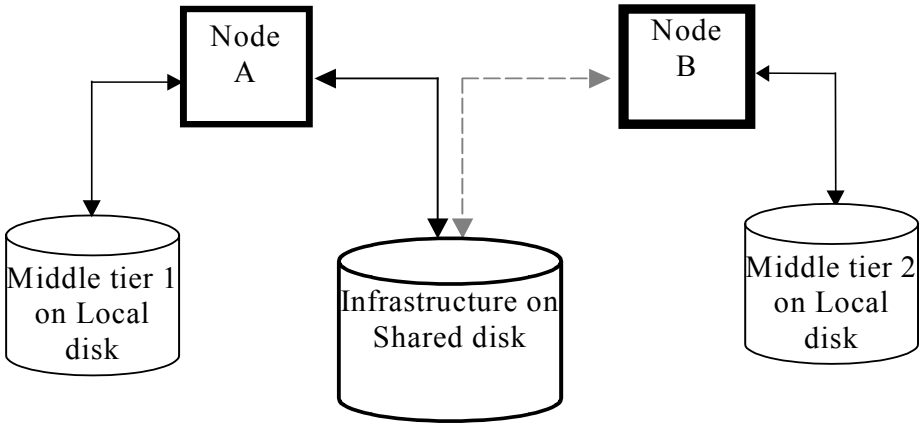
```
For example your Infrastructure uses logical IP address
144.25.245.196 and the virtual hostname infrahost.

If middle tier install prompts for Infrastructure IP address or
hostname enter 144.25.245.196 and infrahost.
```

This should make the physical location of the Infrastructure transparent to the middle tier. Failover and failback the Infrastructure between the 2 nodes of the cluster and verify that the middle tiers continue to run. The middle tiers will experience a small window of downtime, when the Infrastructure is failing over from one node to another, after which they should be able to continue normally.

Installing the Middle Tier on the same nodes as the Infrastructure

Unlike the infrastructure, the middle tier may be installed on multiple active nodes to achieve high availability. Typically the middle tier and infrastructure are installed on separate machines. However, you may wish to install the middle tier on the node(s) where the clustered Infrastructure has been installed. This can be done by installing the middle tier *on local disks* of these nodes as shown in the diagram below. In this scenario, the middle tier is installed on local disks of cluster nodes "A" and "B", but the infrastructure is on a shared disk accessible by both nodes:



Installing Middle Tier on a local disk of clustered nodes

Since the middle tiers are not in a cluster, they cannot be managed by the clusterware and hence do not achieve high availability using the cold failover clustering technique.

The rest of this section describes the steps to install the middle tier(s) as shown in the diagram above (assuming that you have already installed the infrastructure in a cold failover cluster as described earlier):

1. Startup the Infrastructure on the cluster, if it isn't already running.

2. Shut down the cluster software to perform the middle tier installation.

Unlike the infrastructure that is installed only once on the shared disk of the cluster, the middle tiers must be individually installed on local disks of each of the cluster nodes.

3. Rename the infrastructure's "/var/opt/oracle" to "/var/opt/oracle_shared". When Oracle9*i*AS infrastructure and middle tier(s) share a node in a cold failover cluster environment, two "/var/opt/oracle" areas must be maintained: one for the infrastructure where the "oraInst.loc" points to the "oraInventory" directory *on the shared disk* and another for the middle tier instances installed locally on each node where "oraInst.loc" points to another "oraInventory" *on the corresponding local disk*. These directories are needed for applying patches and performing other upgrades or maintenance tasks and therefore must be saved. When applying patches or upgrades on the infrastructure or the middle tiers, the corresponding original copy of "/var/opt/oracle" must be reinstated and made available before the Oracle installer is invoked.

4. Ensure the environment variables referring to the virtual IP and virtual hostname are undefined in the shell where you start the middle Tier installation on a cluster node. While the infrastructure installation uses the virtual hostname and virtual IP shared by the nodes, the middle tier must use the physical hostname and IP of the node where it is installed.

5. Install the middle tier on cluster node "A" while the infrastructure is mounted on that node and then switch (failover) the infrastructure to node "B" to install the middle Tier on node "B". This is done to ensure that the ports selected by the middle tier during installation are different from the ones being used by the Infrastructure.

6. Using the Oracle Universal Installer, perform the middle tier install on the local disk of each node. When prompted for the hostname of the infrastructure in the "EXISTING ORACLE9*i*AS SINGLE SIGN-ON" window of the installer, enter the fully qualified virtual hostname (hostname.domain) used for the infrastructure installation. Proceed to answer the rest of the questions pertaining to the middle tier installation. For the installation to succeed, the infrastructure must be accessible by using the virtual hostname.

7. After successful installation and configuration of the middle tier, start the middle tier using the Enterprise Manager Web site (usually http://<hostname.domain>:1810). Please refer to the Oracle9*i*AS Administration documentation for details.

8. Repeat the middle tier install on other nodes if required.

Appendices

The following appendices provide a set of sample scripts to startup and shutdown all the Infrastructure processes. These scripts are provided in the zip file in the scripts sub-directory. The files include the following -

ias_infra.conf - Defines macros for the scripts
ias_infra.env - Defines the environment variables for the Oracle9*i*AS components
scripts/startiAS.sh – Script owned by root which invokes other scripts to start IAS
scripts/stopiAS.sh – Script owned by root which invokes another script to halt IAS
scripts/startall_1.sh – Script which starts the listener, database and OID
scripts/startall_2.sh – Script which starts EM Website, HTTP Server and OC4J instances
scripts/stopall.sh - Stops all Oracle9*i*AS components and processes.
scripts/reregister.sh – script to re-register mod_osso after the failover

Note: startIAS.sh and stopIAS.sh scripts MUST be owned by root and the permissions must be set to 500 . This is because the startIAS.sh scripts runs the resetiASpasswd.sh script that has the ias_admin password on the command line. To avoid exposing the ias_admin password the script must have the above-mentioned protection. Also, the stopiAS,sh script needs the EM password and hence must be protected to hide that password.

The scripts and the environment settings below are provided as examples and will need to be edited to setup your environment properly before use.

```
#
# Create the following files. Edit and enter the
# appropriate values. The entries below are
# samples. The actual files and scripts in the zip
# file describe the parameters.
#
# Filename : ias_infra.conf
#
FQDN=iascfc.us.oracle.com
IAS_SERVICE_NAME=iasdb.iascfc.us.oracle.com
VERITAS_DISKGROUP_NAME=IAS-dg
VERITAS_VOLUME_NAME=/dev/vx/dsk/IAS-dg/ias902vol
ORACLE_HOME_FS=/ias_infra
IAS_SOFTWARE_OWNER=oracle
IAS_SCRIPTS_BASEDIR=/home/oracle/PACKAGE/scripts
INTERFACE=hme0
LOGICAL_IP=144.25.245.196
IAS_ENV_FILE=/home/oracle/PACKAGE/ias_infra.env

export FQDN IAS_SERVICE_NAME VERITAS_DISKGROUP_NAME
VERITAS_VOLUME_NAME ORACLE_HOME_FS IAS_SOFTWARE_OWNER
IAS_SCRIPTS_BASEDIR INTERFACE LOGICAL_IP IAS_ENV_FILE
#
# Filename : ias_infra.env
#
ORACLE_HOME=/ias_infra
ORACLE_SID=iasdb
PATH=$ORACLE_HOME/bin:/opt/SUNWspro/bin:/usr/ccs/bin:/usr/sbin:
$PATH
LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
DISPLAY=mysun:0
LHOSTNAME=iascfc
LD_PRELOAD=libloghost.so.1
export ORACLE_HOME ORACLE_SID PATH LD_LIBRARY_PATH DISPLAY
LHOSTNAME LD_PRELOAD
```

The next 2 sections describe the scripts to start and stop all of the Oracle9*i*AS components. You will need to edit these scripts as follows

startiAS.sh – Edit the line that does the resetiASpasswd and enter the ias_admin password where it says "<IAS password here>"

stopiAS.sh – Edit the line that says "<EM password here>" and enter the proper password.

With just those 2 changes the start and stop scripts will work as is. It is however recommended to review and test them thoroughly for your implementation.

Appendix B: Starting Infrastructure

```
#!/bin/ksh
#
# Filename : startiAS.sh
# Change the next line to point to the appropriate file with
# the correct location
#
. ./ias_infra.conf
#
# Enable volume management software and mount FS if necessary
#
vxdg -C import ${VERITAS_DISKGROUP_NAME}
vxvol -g  ${VERITAS_DISKGROUP_NAME} startall
mount ${VERITAS_VOLUME_NAME} ${ORACLE_HOME_FS}
#
# Enable logical IP.
#
/usr/sbin/ifconfig ${INTERFACE} addif ${LOGICAL_IP} up

su - ${IAS_SOFTWARE_OWNER} <<!
. ${IAS_ENV_FILE}
${IAS_SCRIPTS_BASEDIR}/startall_1.sh
!
su - ${IAS_SOFTWARE_OWNER}<<!
. ${IAS_ENV_FILE}
resetiASpasswd.sh "cn=orcladmin" <IAS password here>
<ORACLE_HOME here>
${IAS_SCRIPTS_BASEDIR}/reregister.sh
${IAS_SCRIPTS_BASEDIR}/startall_2.sh
!

#!/bin/ksh
#
# Filename : startall_1.sh
#
lsnrctl start
sqlplus "/ as sysdba" <<!
startup
!
oidmon start
oidctl server=oidldapd configset=0 instance=1 start

# !/bin/ksh
#
# Filename : startall_2.sh
#
${ORACLE_HOME}/dcm/bin/dcmctl start -ct ohs -v
${ORACLE_HOME}/dcm/bin/dcmctl start -co OC4J_DAS -v
$ emctl start

# !/bin/ksh
#
# Filename : reregister.sh
#
${ORACLE_HOME}/jdk/bin/java -jar
${ORACLE_HOME}/sso/lib/ossoreg.jar -site_name
${IAS_SERVICE_NAME} -success_url
http://$FQDN:7777/osso_login_success -cancel_url
http://$FQDN:7777 -logout_url
http://$FQDN:7777/osso_logout_success -home_url
http://$FQDN:7777 -config_mod_osso TRUE -oracle_home_path
${ORACLE_HOME} -u root -apache_server_root
${ORACLE_HOME}/Apache/Apache -config_file
${ORACLE_HOME}/Apache/Apache/conf/osso/osso.conf -
sso_server_version v1.2
```

Appendix C: Stopping Infrastructure

```
#!/bin/ksh
#
# Filename : stopiAS.sh
#
# Change the next line to point to the appropriate file with
# the correct location
#
. ./ias_infra.conf
su - ${IAS_SOFTWARE_OWNER} <<!
. ${IAS_ENV_FILE}
emctl stop <<EOF
<EM PASSWORD HERE>
EOF
${IAS_SCRIPTS_BASEDIR}/stopall.sh
!

umount ${ORACLE_HOME_FS}
vxdg deport ${VERITAS_DISKGROUP_NAME}
/usr/sbin/ifconfig ${INTERFACE} removeif ${LOGICAL_IP}

#!/bin/ksh
#
# Filename : stopall.sh
#
$ORACLE_HOME/dcm/bin/dcmctl stop  -v
oidctl server=oidldapd configset=0 instance=1 stop
sleep 15
oidmon stop
sqlplus "/ as sysdba" <<!
shutdown immediate
!
lsnrctl stop
kill -9 `ps -ef | grep opmn | grep -v grep | awk '{print $2}'`
# kill opmn processes
```

Appendix D: Enabling Sun Cluster Dataservices

Describing a complete Sun Cluster dataservice, which automatically starts, stops, monitors and fails over the Infrastructure components is beyond the scope of this paper. However, there are a few important things worth mentioning for the benefit of the data service developer.

1. The startiAS.sh and stopiAS.sh scripts can be used with little or no modification as the start and stop methods of the dataservice. However, for a more robust service the scripts should be modified to handle errors and even do internal retries before returning control to the cluster software.
2. The virtual hostname that was used in the Infrastructure install must be used as the logical hostname for the dataservice. Once this is setup, the startiAS.sh and stopiAS.sh scripts must no longer enable and disable the IP address, since the cluster will automatically do this for you. Remember to modify the scripts and remove the "ifconfig addif " and "ifconfig removeif " lines .
3. Consider using SUNW.hastorage plus resource type or Global File system for housing the ORACLE_HOME file system, so that it is also managed automatically by Sun Cluster. Mounting and unmounting of file systems is not recommended in the start/stop methods of a Sun Cluster Data Service.
4. Depending on the complexity of your dataservice , you can monitor for Oracle9*i*AS processes like HTTP Servers, OID, database background processes and database listener and perhaps include a few retries on the local node in the dataservice logic, before a failover to the other node.

Authors: Ashish Prabhu, Rabah Mediouni, Thomas Fillenwarth and Ashesh Parekh

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

Oracle Corporation provides the software
that powers the internet.