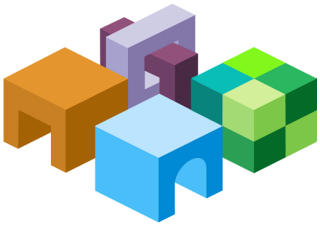


# HYPERION® SHARED SERVICES

*RELEASE 9.3.1*

# SHARED SERVICES AND REPORTING AND ANALYSIS HIGH AVAILABILITY (UNIX ENVIRONMENTS)

ORACLE | Hyperion

Contents In Brief

# About Shared Services and Reporting and Analysis High Availability on UNIX

To make Oracle's Hyperion® Shared Services and Oracle's Hyperion® Reporting and Analysis – System 9 highly available, you must use clustering solutions to ensure that none of these components is a single point of failure:

- Database
- Native Directory and other user directories
- Web Applications (Shared Services, Oracle's Hyperion® Workspace, Oracle's Hyperion® Web Analysis – System 9, Oracle's Hyperion® Financial Reporting – System 9)
- Reporting and Analysis services (Core, IR, DAS, FR)

Database clustering solutions depend on the relational database management system (RDBMS) that you use. Hyperion products support Oracle Real Application Clusters (RAC) and third-party RDBMS software. See the documentation for your RDBMS. For the other components (OpenLDAP Native, Shared Services Web application, Reporting and Analysis Service, Reporting and Analysis Web Applications), the supported configuration uses Oracle Clusterware to create an active-passive failover cluster. See "Configuring Oracle Clusterware Failover Cluster" on page 4.

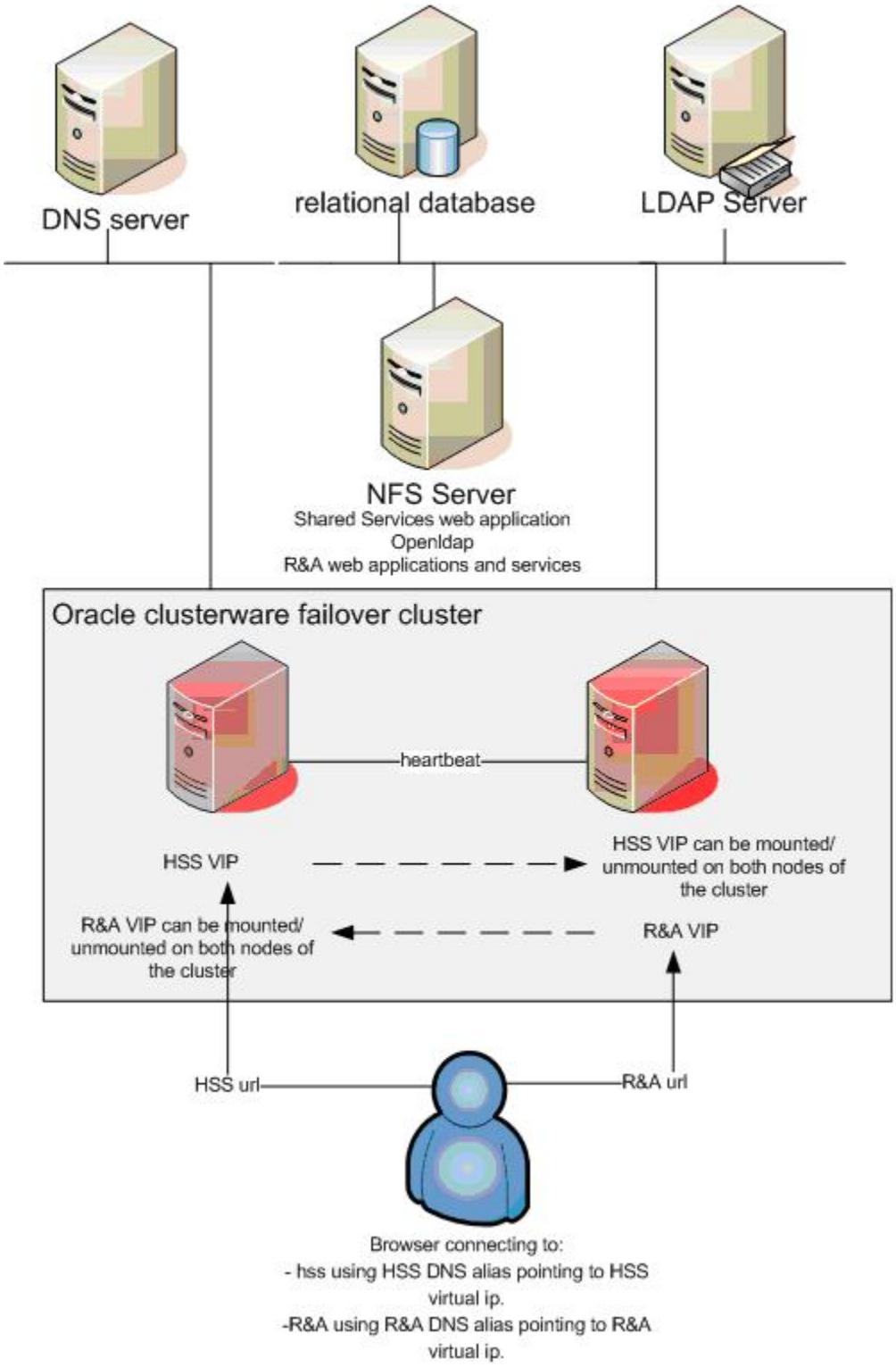
You cluster Shared Services and Reporting and Analysis for failover using Oracle Clusterware 11.1, which is available for free to protect Hyperion components.

You can download Oracle Clusterware from http://www.oracle.com/technology/software/products/database/index.html, under Database 11g Release 1. Information about Oracle Clusterware is available from http://www.oracle.com/technology/products/database/clusterware/index.html. Licensing information is available at http://download.oracle.com/docs/cd/B28359_01/license.111/b28287.pdf.

# Strategy for Deploying Shared Services and Reporting and Analysis in a Failover Cluster on UNIX

The Shared Services and Reporting and Analysis deployment strategy described in this document has been tested successfully on multiple platforms, including Solaris 10 global zone with Korn shell and Oracle Enterprise Linux 5 with Bash shell.

Figure 1 Shared Services and Reporting and Analysis Oracle Clusterware Active-Passive Failover Cluster



## Two-Node Topology

The Oracle Clusterware installation is described for a two-node topology where one node stays active and the other node is passive for a given application. Both nodes can service different applications at the same time, but only one node can serve a specific application at a time.

In a failover cluster, the Shared Services processes (Web application and Native Directory) and Reporting and Analysis processes (Core, IR, FR, WA services) are accessible at specific virtual IP (VIP) address referenced by a floating cluster host name (*hsscluster* and *racluster*) or DNS alias. If the primary node fails, the VIPs and the Shared Services or Reporting and Analysis processes move automatically and independently to the secondary node, as shown in Figure 1. Both nodes mount the Hyperion files stack from a network file system (NFS) server that is separate from the Clusterware nodes. Both nodes and the NFS server use the same user name per application to manage the entire lifecycle of Shared Services and Reporting and Analysis, from installation to the operational behavior under Oracle Clusterware control

## Files Location

Although the Oracle Clusterware installation is performed from one node, the Oracle Clusterware binaries are laid out locally on each node, outside the shared storage NFS server. Together, the NFS mounted file systems hold the following elements:

- Shared Services configured binaries
- Reporting and Analysis configured binaries
- Oracle Clusterware data files—Oracle Clusterware Registry (OCR) and the voting file
- Shell scripts to create VIP, Shared Services, and Reporting and Analysis profiles and to register them with the Oracle Clusterware Registry

The Shared Services and Reporting and Analysis profiles register appropriate perl action scripts that define the start, stop, and health-check behavior of processes.

There is no restriction on the underlying shared storage (NAS, local NFS server, and so on), which should be made highly available in production environments.

# Configuring Oracle Clusterware Failover Cluster

Before clustering Shared Services and Reporting and Analysis for failover, you must meet Oracle Clusterware prerequisites and then install and configure Oracle Clusterware. This document refers to the "Oracle Clusterware Preinstallation Tasks," "Configuring Oracle Clusterware Storage," and "Installing Oracle Clusterware" sections of the Oracle Clusterware installation guides for UNIX, which are available from http://www.oracle.com/pls/db111/homepage:

- Linux—http://download.oracle.com/docs/cd/B28359_01/install.111/b28263.pdf
- Solaris—http://download.oracle.com/docs/cd/B28359_01/install.111/b28262.pdf
- HP-UX—http://download.oracle.com/docs/cd/B28359_01/install.111/b28259.pdf
- AIX—http://download.oracle.com/docs/cd/B28359_01/install.111/b28258.pdf

Review Chapter 1, "Summary List: Installing Oracle Clusterware," in the *Oracle Clusterware Installation Guide* before proceeding with this section.

## Oracle Clusterware Prerequisites

Perform the steps described in Chapter 2, "Oracle Clusterware Preinstallation Tasks," in the *Oracle Clusterware Installation Guide*, using these notes:

- Create the system group `oinstall`, and then create three users on all three nodes, all part of the `oinstall` group:
  - `oracle` for CRS to start, stop, and check resources
  - `orclhss` for the user running the Shared Services processes
  - `orclra` for the user running the Reporting and Analysis processes

  Each user must have the same UID on all nodes. For instance, if on one node, user `orclhss` has UID 108, that user must have UID 108 on all other nodes. This will be necessary for NFS mount points. For more information, see "Configuring Oracle Clusterware Shared Storage" on page 7 of this document.

  Example with `orclhss` and `orclra` (done on all nodes):

  ```
  more /etc/group
  more /etc/passwd
  #If uid 102,107,108 are not used:
  useradd -d /export/home/orclhss -m -u 108 -g oinstall -s /bin/ksh
  orclhss
  useradd -d /export/home/orclra -m -u 107 -g oinstall -s /bin/ksh orclra
  ```

  After you create `orclhss`, `oracle`, and `orclra`, change their passwords.

- Create an Oracle Clusterware home directory. As root on each cluster node, create a path:

  ```
  mkdir -p /vol1/app
  chown -R oracle:oinstall /vol1/app
  ```

  During installation, you can choose a location for `oraInventory` (owned by user `oracle`) and for *Oracle Clusterware*; for example, `/vol1/app/oraInventory` and `/vol1/app/11.1.0/crs`.

- Using NTP, ensure that the server clocks are synchronized.
- Ensure that each server has at least two network interfaces (more if the cards are teamed).
- The public and private networks must be created and must be physically distinct, on different subnets. VIPs are not created manually; the only manual task is to create the entries in `/etc/hosts`.
- Define IPs and VIPs, making sure to meet *Oracle Clusterware* requirements, as shown in Figure 2 on page 7. Plan VIPs for Shared Services (for example, `10.10.12.98/255.255.0.0`) andReporting and Analysis (for example, `10.10.12.97/255.255.0.0`) with a corresponding DNS entry (for example, *`hsscluster`* and *`racluster`*).

  The subnet of the VIP created and managed by Oracle Clusterware must be the same as the subnet of the physical IP on the interface to which the VIP is assigned. VIPs are not assigned

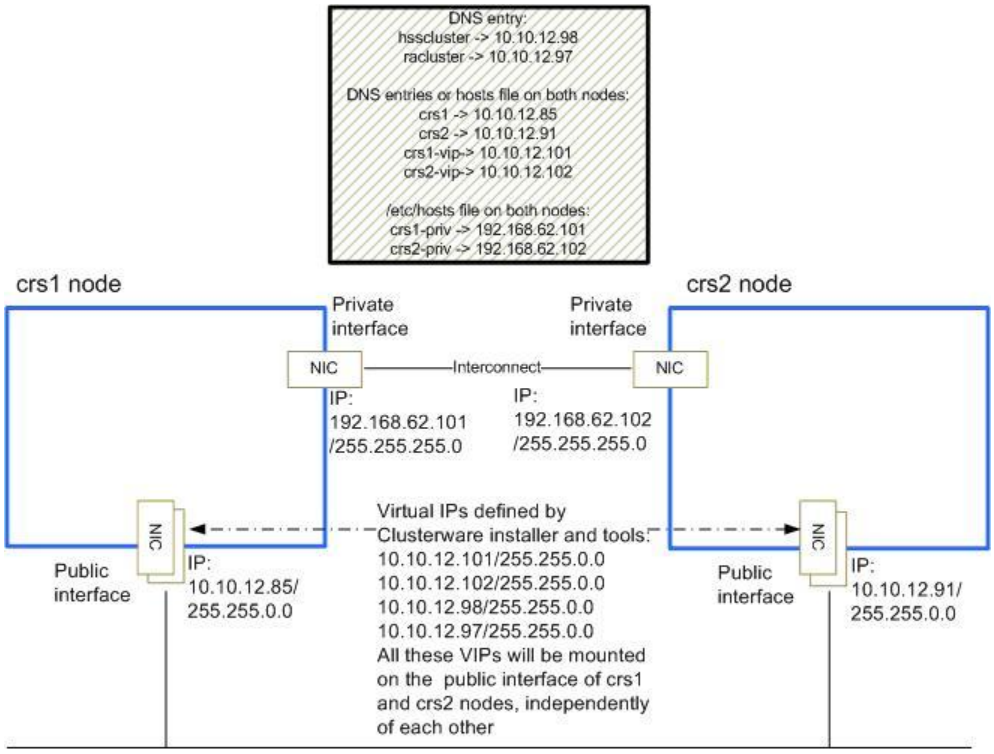
to a fixed machine. When running, a VIP is physically bound to a specific interface on one machine in the cluster. However, *Oracle Clusterware* requires that each node have one specific VIP. A third and fourth VIP are used for Shared Services and Reporting and Analysis. Oracle Clusterware migrates the mount point of VIPs when failover occurs.

Example, with `/etc/hosts` on all nodes:

```
#
# Internet host table
#
10.10.12.98 hsscluster #and dns name
10.10.12.97 racluster #and dns name
10.10.12.84 nfsserver #and dns name
#
10.10.12.85 crs1 #and dns name
192.168.62.101 crs1-priv
10.10.12.101 crs1-vip
#
10.10.12.91 crs2 #and dns name
192.168.62.102 crs2-priv
10.10.12.102 crs2-vip
```

**Note:** You need not create the `crs1-vip` and `crs2-vip` interfaces manually. When configuring network interfaces, set TCP/IP parameters only for the public and the private interface, without configuring VIPs.

**Figure 2 IP and VIP Definitions**



# Configuring Oracle Clusterware Shared Storage

## Important Rules

In the Oracle Clusterware context, you must observe two rules regarding the Shared Services and Reporting and Analysis configuration with NFS:

- The installation and the configuration are performed on a separate NFS server, whose host name must be changed during configuration, at the system level, to match the chosen cluster floating host names. (You could install and configure on any another node instead, but always through the NFS mount point, after having changed the node name to the cluster name.) For example, when configuring Shared Services, temporarily change the NFS server host name to `hsscluster`; when configuring Reporting and Analysis, temporarily change the host name to `racluster`.

  The Shared Services and Reporting and Analysis stack learns the floating cluster host names in all its internal properties, files, and database settings. When the configuration is completed, the NFS host name is reset to the original name.

  The installation and the configuration on the NFS server must use the same name as the mount point that the nodes will mount from the NFS server, so the NFS server mounts itself at Shared Services and Reporting and Analysis setup time. In other words the setup is performed through client mount settings, not the NFS sharing settings.

- The same user names (identical UIDs) and group names (identical GIDs) must be used for Shared Services servicing and Reporting and Analysis servicing on the three hosts: installation and configuration on the NFS server, and maneuvering (start, stop, relocation) on the nodes. This document refers to the Shared Services user name `orclhss` (UID 108), the Reporting and Analysis user name `orclra` (UID 107), and the group name `oinstall` (GID 100).

## Configuration

Perform the steps documented under "Configuring Oracle Clusterware Storage" in the *Oracle Clusterware Installation Guide*. The storage for Shared Services and Oracle Clusterware voting disks and OCR uses a separate NFS server in this configuration.

Successful NFS server options found in `/etc/dfs/dfstab`:

```
share -F nfs -o root=crs1: crs2,anon=102 /vol1/sharedcrs
share -F nfs -o root=crs1: crs2,anon=108 /vol1/sharedk2hss
share -F nfs -o root=crs1: crs2,anon=107 /vol1/sharedk2ra
```

where `anon=107` has the UID of the user `orclra`, `anon=102` has the UID of the user `oracle`, and `anon=108` has the UID of the user `orclhss`.

MetaLink document 359515.1, *Mount Options for Oracle Files When Used with NAS Devices*, provides the mount options required. The shared mount point can be created on a network attached storage (NAS) device or on a disk plain partition. However, redundancy is strongly advised for high availability of the physical file system.

This MetaLink document separates binaries, data files, and CRS voting disk and OCR mount options:

- For the partition containing the Shared Services and Reporting and Analysis binaries and data, use Oracle data files mount options.

  **Note:** For performance reasons, it is essential that you skip `noac` mount option when mounting the Shared Services and Reporting and Analysis files. Be sure to countercheck the mount options on all cluster nodes, using the mount command as root.

- For the partitions containing voting disks and OCR, use CRS voting disks and OCR mount options.

### Example 1: Creating the NFS Mount Point for Oracle Clusterware Files, OCR and CRS Voting Disk, and Action Scripts.

On the NFS server:

```
# chown -R oracle:oinstall /vol1/sharedcrs
drwxrwx---    oracle   oinstall      sharedcrs/
```

The following must be performed on all cluster nodes:

```
# mkdir -p /mtk2crs
# mount -F nfs -o
rw,hard,bg,nointr,rsize=32768,wsize=32768,noac,proto=tcp,vers=3,xattr
nfsserver:/vol1/sharedcrs /mtk2crs
```

where `nfsserver` is the name of your NFS server.

**Note:** You can use `/etc/dfs/dfstab` to automate the mount after machine startup.

When installing Oracle Clusterware, you choose an NFS mounted location for the OCR file (`/mtk2crs/ocr`) and for the voting file (`/mtk2crs/voting`).

### Example 2: Creating the NFS Mount Point for Shared Services Files

On the NFS server:

```
# chown -R orclhss:oinstall /vol1/sharedk2hss

drwxrwx---    orclhss  oinstall     sharedk2hss/
```

This must be performed on all cluster nodes:

```
# mkdir -p /mtk2sshss
# mount -F nfs -o
rw,hard,bg,nointr,rsize=32768,wsize=32768,proto=tcp,vers=3,xattr
nfsserver:/vol1/sharedk2hss /mtk2sshss
```

where `nfsserver` is the name of your NFS server

### Example 3: Creating the NFS Mount Point for Reporting and Analysis Files

On the NFS server:

```
# chown -R orclra:oinstall  /vol1/sharedk2ra
drwxrwx---  orclra   oinstall     sharedk2ra/
```

This must be performed on all cluster nodes:

```
# mkdir -p /mtk2ssra
# mount -F nfs -o
rw,hard,bg,nointr,rsize=32768,wsize=32768,proto=tcp,vers=3,xattr
nfsserver:/vol1/sharedk2ra /mtk2ssra
```

where `nfsserver` is the name of your NFS server.

## Installing and Configure Oracle Clusterware

➤ To install and configure Oracle Clusterware:

1 **Perform the steps documented under "Installing Oracle Clusterware" in the *Oracle Clusterware Installation Guide*.**

2 **To ensure that the nodes are secure shell (SSH)-accessible, run**

```
$ exec /usr/bin/ssh-agent $SHELL
$ /usr/bin/ssh-add
```

3 **Launch the Oracle Clusterware installer on one node only, with the user name `oracle`.**

Start the `runInstaller` command that can launch an X11 GUI:

```
$ export DISPLAY=yourx11ip:0.0
$ ./runInstaller &
```

There is no console mode.

4 **On the Specify Cluster Configuration screen, enter the public, private, and virtual host names for both nodes.**

**Note:** You must click Add to enter information about `crs2`.

5 **In Specify Network Interface Usage, specify the private and public interface.**

6 **On the Cluster Configuration Storage screen, to ensure that the mount point `/mtk2crs` is mounted. Specify these locations:**

- OCR external redundancy and OCR location; for example, `/mtk2crs/ocr`
- voting external redundancy and voting location; for example, `/mtk2crs/voting`

**Note:** This `/mtk2crs` folder (mount point) cannot be changed after installation. This information is stored in `/var/opt/oracle/ocr.lock` (Solaris) and `/etc/oracle/ocr.lock` (Linux) but cannot be changed after configuration.

7 **Click Finish.**

8 **From a shell, check the Clusterware installation as user `oracle`:**

```
$ /vol1/app/11.1.0/crs/bin/cluvfy stage -post crsinst -n crs1,crs2
```

**Note:** Postconfiguration validation of virtual IPs may fail with Hyperion Configuration Utility or the `cluvfy` command. This is not a problem if you can see the virtual IP on the nodes using the `ipconfig /all` command.

9 **Add `/vol1/app/11.1.0/crs/bin/` to the `$PATH` statement for the users `oracle`, `orclra`, and `orclhss`.**

## Creating an Oracle Clusterware Application VIP

Create two application VIPs for Shared Services and Reporting and Analysis that are started and stopped on the Public interface by Oracle Clusterware. The VIP resources are owned and started by root. See http://download.oracle.com/docs/cd/B28359_01/rac.111/b28255/crschp.htm#sthref369.

You can use the `createviphss.sh` and `createvipra.sh` templates provided in the Appendix of this document to create a virtual IPs. Put your scripts in the NFS mount point directory script path:

```
/mtk2crs/crs_actions/hss/createviphss.sh
/mtk2crs/crs_actions/ra/createvipra.sh
```

**Caution!** When copying the script from this document, make sure that the `crs_profile` command is not truncated on multiple lines.

Edit the variables as required.

For Shared Services

- VIPIP default—`10.10.12.98`
- VIPSUBNET default—`255.255.0.0`
- CRS_HOME default—`/vol1/app/11.1.0/crs`
- ADAPTER—The physical interface on which the VIP will be mounted

For Reporting and Analysis:

- VIPIP default—`10.10.12.97`
- VIPSUBNET default—`255.255.0.0`
- CRS_HOME default—`/vol1/app/11.1.0/crs`
- ADAPTER—The physical interface on which the virtual VIP will be mounted

**Note:** The default adapter is `eri0`. You can check the adapter using `ifconfig -a`. When the Public interface is built upon teamed NICs, the `ADAPTER` variable should reflect the team's name.

Run the scripts as `root # /mtk2crs/crs_actions/hss/createviphss.sh` and `/mtk2crs/crs_actions/ra/createvipra.sh`.

Verify that the VIPs have been successfully registered, using the `crs_stat -t -v` command You can start them using `crs_start hssvip` or `crs_start ravip`.

If you need to reregister the VIPs, you must first unregister them, using the following commands:

```
$CRS_HOME/bin/crs_profile -delete VIP
```

```
$CRS_HOME/bin/crs_unregister VIP
```

where *VIP* is `hssvip` for Shared Services or `ravip` for Reporting and Analysis.

# Oracle Clusterware Postinstallation Procedures

## Shared Services Installation

Install Shared Services on one node of the cluster, following the instructions in the *Hyperion Shared Services Installation Guide* and these guidelines:

- Download Shared Services Release 9.3.1 Service Pack 1.

  On the NFS server where you will install Shared Services:

  - Change the NFS server host name to `hsscluster` (`/etc/nodename` and `/etc/hostname.eri0` for Solaris).
  - Populate `/etc/hosts` with the `hsscluster` entry on the first line of the file, and ensure that you can ping `hsscluster`:

    `10.10.12.84 hsscluster`, where `10.10.12.84` is the IP of the NFS server
  - Reboot the system (init 6)
  - Clean up the `$HOME` directory for user `orclhss`. There should be no entries such as these:

    ```
    .hyperion.*
    ```

    ```
    vpd.properties
    ```

- As user `root`, mount the NFS server on itself:

  - `mkdir -p /mtk2sshss`
  - `chown -R orclhss:oinstall /vol1/sharedk2hss`
  - `mount -F nfs -o rw,hard,bg,nointr,rsize=32768,wsize=32768,proto=tcp,vers=3,xattr nfsserver:/vol1/sharedk2hss /mtk2sshss`

  Make sure that user `orclhss` has the correct rights to the `/mtk2sshss` mount point after mount is done. After every reboot, make sure that you mount the NFS mount points if you didn't add entries in `/etc/dfs/dfstab`.

- As user `orclhss`, install Shared Services on the NFS server using the shared NFS client mount point:

`$ ./setup.bin -console` (or `./setup.sh -console` on Linux)

If the mount point is `/mtk2sshss`, you can, for example, use `/mtk2sshss/Hyperion` as the Hyperion home directory.

Do not launch Hyperion Configuration Utility from the installer when the installation is complete.

**Note:** After Shared Services is installed, the installation user (`orclhss`) should check or create three identical files in the home directories: `.hyperion.hsscluster`, `.hyperion.crs1` (where `crs1` is the node name of the first node in the cluster), and `.hyperion.crs2` (where `crs2` is the node name of the second node in the cluster). These files contain the *`HYPERION_HOME`* path built onto the NFS mount point; for example: `HYPERION_HOME=/mtk2sshss/Hyperion`. This must be done on all nodes of the cluster in the home directory of the `orclhss` user.

## Configuring Shared Services

Use Hyperion Configuration Utility to configure Shared Services on the NFS server. See the *Hyperion Shared Services Installation Guide*.

**Prerequisite**—Ensure that no Shared Services processes are running.

➤ To configure Shared Services:

1. **As user `orclhss`, launch Hyperion Configuration Utility:**

```
$ cd /mtk2sshss/Hyperion/common/config
$ ./configtool.sh -console
```

2. **Select Configure database and Deploy to Application Server.**
3. **Finish the configuration.**
4. **Start Shared Services.**
5. **Go to http://10.10.12.84:58080/interop/index.jsp and verify that you can log on.**
6. **Stop the Hyperion Foundation Shared Services – Web application and Hyperion Foundation OpenLDAP processes, if they were started.**

   The NFS server host name can be reset, as well as `/etc/hosts` file. From this point, all the operations regarding Shared Services (registration scripts; start, stop, check in perl action script) are performed on the Oracle Clusterware nodes mounting the NFS files:

```
# mkdir -p /mtk2sshss
# mount -F nfs -o
rw,hard,bg,nointr,rsize=32768,wsize=32768,proto=tcp,vers=3,xattr
nfsserver:/vol1/sharedk2hss /mtk2sshss
```

   See the mount options for Shared Services files in "Configuring Oracle Clusterware Shared Storage" on page 7.

**Note:** For performance reasons it is essential to skip the `noac` mount option when mounting the Shared Services files. Installation and deployment can take significant time if the mount option does not cache files attributes.

## Registering Shared Services in the Cluster

Registering Shared Services in the cluster requires these tasks:

- Create an action script to start / stop and check Shared Services services.

  You can use the `hss931.pl` action script template provided in Appendix of this document. Put your scripts in the NFS mount point directory script path, `/mtk2crs/crs_actions/hss/`, and ensure that the `oinstall` group has read-write and execute access to this folder.

  For portability, the perl action script `hss931.pl` must be updated to reflect these items:

  - The exact path of the user's home directory

    For example, for the user used for Shared Services configuration, the perl action script contains `my $home = "/export/home/orclhss".`

  - The exact path of the `wget` command.

    Example:

    ```
    /usr/bin/wget on OELinux 5
    /usr/sfw/bin/wget on Solaris 10
    ```

  - `my $SCRIPT_PATH = "/mtk2crs/crs_actions/hss";`
  - `my $HSS_URL = "http://hsscluster:58080/interop/index.jsp";`
  - `my $LDAP_HOST = "hsscluster";`
  - `my $HYPERION_HOME = "/mtk2sshss/Hyperion/";`
  - `my $DEBUG_ENABLED = "false"; # do not switch to true in production environment`
  - Change `my $HSS_STARTSCRIPT2` and (and `my $HSS_STARTSCRIPT1`, if applicable) to the correct value for your Web application server.
  - Change `my $HSS_STOPSCRIPT1`(and `my $HSS_STOPSCRIPT2` if applicable) to the correct value for your Web application server.

  Make sure that perl is in the `$PATH` environment variable of the `orclhss` user. When creating the perl script `hss931.pl`, make sure that `orclhss` and `oinstall` have execute rights on it.

- Create and register a Shared Services application profile in Oracle Clusterware.

You can use the `register_hss.sh` template that is provided in the Appendix to create a Shared Services application profile and to register the profile in the Oracle Clusterware Registry. The profile points to the perl action script.

Copy `register_hss.sh` in the NFS mount point directory script path: `/mtk2crs/crs_actions/hss/`.

➤ To register Shared Services in the cluster:

1 **Check the Shared Services perl action script:**

a. Start `hssvip` on one node:

```
# crs_start hssvip
# crs_stat -t -v
```

Check on which node `hssvip` runs.

b. Ensure that the action program works before registering in a cluster:

i. On the node where the VIP is mounted, as user `orclhss`, launch from a shell:

```
$ perl hss931.pl start
$ echo $?
0 #denotes success
1 #denotes failure
```

Try to connect to `http://hsscluster:58080/interop/index.jsp`.

ii. Check the other commands:

```
$ hss931.pl check
$ echo $?
0    #denotes success of the Shared Services processes
1    #denotes failure of the Shared Services processes
$ hss931.pl stop
```

2 **Modify the Shared Services registration script:**

- Edit the variables in `register_hss.sh` as required:

```
ACTION_SCRIPT=/mtk2crs/crs_actions/hss/hss931.pl
CRS_HOME=/vol1/app/11.1.0/crs
```

and

```
START_TIMEOUT=120
STOP_TIMEOUT=120
SCRIPT_TIMEOUT=120
```

if the startup time of Shared Services does not exceed 120 seconds.

- Ensure that Shared Services is started under the correct user. The process that is started on behalf of the resource runs with the user specified through the `crs_setperm` command. The default script uses the user name `orclhss`:

```
$CRS_HOME/bin/crs_setperm $APPNAME -o orclhss
$CRS_HOME/bin/crs_setperm $APPNAME -g oinstall
```

3 **Run the script as** `root # /mtk2crs/crs_actions/hss/register_hss.sh`.

**Note:** If you need to correct an error and reregister, you can use `unregister_hss.sh`, which is provided in the provided in this document's Appendix.

4 **Test starting, stopping, and relocating Shared Services services using the cluster commands:**

```
- crs_stat -t -v  # check applications status
- crs_start -f  hss_app
- crs_stop -f hss_app
- crs_relocate -f hss_app
```

**Note:** if you have issues starting the `hss_app` resource using `crs_start`, check the `crsd` log in `$CRS_HOME/crs/log/nodename/crsd`.

## Reporting and Analysis Installation

Install Reporting and Analysis on the NFS server, following the instructions in the *Hyperion Shared Services Release 9.3.1.0.0x Installation Guide* and these guidelines:

- Stop the cluster on both cluster nodes using `crsctl stop crs`.
- Change the NFS server host name to `racluster` (`/etc/nodename` and `/etc/hostname.eri0` for Solaris) .
- Populate `/etc/hosts` with the `racluster` entry on the first line of the file, and ensure that you can ping `racluster`:

  `10.10.12.84 racluster`, where `10.10.12.84` is the IP of the NFS server (not the `racluster` VIP)

  and add

  10.10.12.98 `hsscluster`, where 10.10.12.98 is the `hsscluster` VIP.
- Reboot the system (init 6)

  **Note:** It is good practice to stop the cluster on both nodes using `crsctl stop crs` before rebooting the NFS server. When the NFS server is running again, you can start the cluster by issuing the command as `root: crsctl start crs` on both nodes.
- Clean up the `$HOME` directory for user `orclra`. There should be no entries such as these:

```
.hysl
.hyperion.*
vpd.properties
```

- As user root on the NFS server, mount the NFS server on itself:
  - `mkdir -p /mtk2ssra`
  - `chown -R orclra:oinstall /vol1/sharedk2ra`
  - `mount -F nfs -o rw,hard,bg,nointr,rsize=32768,wsize=32768,proto=tcp,vers=3,xattr nfsserver:/vol1/sharedk2ra /mtk2ssra`

  Ensure the user `orclra` has the correct rights to the `/mtk2ssra` mount point after mount is done,

  After every reboot, ensure that you mount the NFS mount points if you did not add entries in /`etc/dfs/dfstab`.
- As user `orclra`, install Reporting and Analysis components (services, repServices, and uiservices) on the NFS server using the shared NFS client mount point for services, uiservices and production reporting services:

  `$ ./ setup`*`OS`*`.bin –console`

If the mount point is `/mtk2ssra`, you can, for example, use `/mtk2ssra/Hyperion` as *HYPERION_HOME* and `/mtk2ssra/Hyperion/BIPlus` as *BIPLUS_HOME*.

**Note:** Installation and deployment can take significant time if the mount option does not cache files attributes (skip `noac`).

**Note:** After Reporting and Analysis is installed, the installation user (`orclra`) should check or create three identical files in the home directories: `.hyperion.racluster`, `.hyperion.crs1`(where `crs1` is the node name of the first node in the cluster), and `.hyperion.crs2`(where `crs2` is the node name of the second node in the cluster). These files contain the *HYPERION_HOME* path built onto the NFS mount point; for example, `HYPERION_HOME=/mtk2ssra/Hyperion`. This must be done on all nodes of the cluster in the home directory of the `orclra` user.

## Configuring Reporting and Analysis

Use Hyperion Configuration Utility to configure Reporting and Analysis on the NFS Server. See the *Hyperion Shared Services Release 9.3.1.0.0x Installation Guide*.

➤ To configure Reporting and Analysis:

1 **Make sure Shared Services is running in the cluster:**

- Use `crsctl check crs` to check if cluster is running on both nodes. Start it using `crsctl start crs` on both nodes if not already running.
- Use `crs_stat -t -v` to check the status of Shared Services resources in the cluster.

2 **As user `orclra`, launch the Hyperion Configuration Utility:**

```
$ cd /mtk2ssra/Hyperion/common/config/
$ ./configtool.sh -console
```

3 Select **Register with Shared Services**, then **Product Options**, **Configure database**, **Configure Financial reporting**, **Deploy to Application Server**, and **Configure WebServer**:

- For **Shared Services Server**, enter `hsscluster`.
- For **Workspace Web URL**, enter `http://racluster:19000`
- For **Financial Reporting host name**, enter `racluster`. Do not change the Workspace, Financial Reporting, Web Analysis Web applications root contexts, also called "Server Name" in the configurator.
- For **Financial Reporting Print Service Location**, **Scheduler Service**, and **Financial Reporting Server**, enter `racluster`.

**Note:** If you have an error message about `/var/tmp/SharedServicesSecurityClient.log`, modify the access rights to this file so that the `oinstall` group can read and write this file.

4 **Finish the configuration.**

5 **Update** *BIPLUS_HOME*/bin/startCommonServices.sh startup script **to add** -Drm.ignore_host=true **to** BP_JRE **parameters:**

```
nohup ${BP_JRE} -Drm.ignore_host=true -cp ${BP_CLASSPATH} ${BP_FLAGS} $
{BP_CMD} >${BP_CONSOLELOG} 2>&1 &
```

6 **In** *HYPERION_HOME*/common/httpServers/Apache/2.0.52/conf/httpd.conf **:**

a. Update group permissions to run as a different user or group

User orclra

Group oinstall

b. Add the following line:

```
ServerName racluster
```

The NFS server host name can now be reset, as well as /etc/hosts. From this point, all the operations regarding Reporting and Analysis (registration scripts; start, stop, check in perl action script) are performed on the Oracle Clusterware nodes mounting the NFS files:

```
# mkdir -p /mtk2ssra
# mount -F nfs -o
rw,hard,bg,nointr,rsize=32768,wsize=32768,proto=tcp,vers=3,xattr
nfsserver:/vol1/sharedk2ra /mtk2ssra
```

## Registering Reporting and Analysis in the Cluster

Registering Reporting and Analysis in the cluster requires these tasks:

- Create an action script to start / stop and check Reporting and Analysis services.

  You can use the ir931.pl action script template provided in Appendix of this document. Put your scripts in the NFS mount point directory script path, /mtk2crs/crs_actions/ra , and ensure that the oinstall group has read-write and execute access to this folder.

  At a minimum, validate and change if necessary the following variables:

```
my $SCRIPT_PATH     = "/mtk2crs/crs_actions/ra";
my $WEB_URL       = "http://racluster:19000
my $home = "/export/home/orclra/";   # home directoy of user orclra
my $HYPERION_HOME = "/mtk2ssra/Hyperion";
my $BIPLUS_HOME = "/mtk2ssra/Hyperion/BIPlus";
my $SLEEP_INTERVAL  = 120;
$res |= process_check("IRM", "$BIPLUS_HOME/bin/run/
ir_process_IRM1_racluster.pid");
```

  where *racluster* must be changed if your cluster name is different.

  **Note:** If you change the cluster name, it must be changed inside the perl script.

- Create and register a Reporting and Analysis application profile in Oracle Clusterware.

  You can use the register_ir.sh template provided in the Appendix to create a Reporting and Analysis application profile and to register the profile in the Oracle Clusterware registry. The profile points to the perl action script. Put your script in the NFS mount point directory script path: /mtk2crs/crs_actions/ra.

Make sure that perl is in the `$PATH` statement.

➤ To register Reporting and Analysis in the cluster:

1 **Check the Reporting and Analysis perl action script:**

a. Start the `hss_app` application:

```
# crs_start hss_app
# crs_stat -t -v
```

b. Start `ravip` on one node:

```
# crs_start ravip
# crs_stat -t -v
```

Check on which node `ravip` runs.

c. Ensure that the Reporting and Analysis action program works before registering Reporting and Analysis in a cluster:

i. On the node where the `ravip` is mounted, as user `orclra`, launch from a shell:

```
$ ir931.pl start
$ echo $?
0 #denotes success
```

ii. Check the other commands:

```
$ ir931.pl check
$ echo $?
0 #denotes success of the R&A processes
1 #denotes failure of the R&A processes
$ ir931.pl stop
```

2 **Create and register an application profile for Reporting and Analysis:**

- Edit the variables in `register_ir.sh` as required:

```
ACTION_SCRIPT=/mtk2rs/crs_actions/ra/ir931.pl
CRS_HOME=/vol1/app/11.1.0/crs
```

- Ensure that Reporting and Analysis is started under the correct user. The process that is started on behalf of the resource runs with the user specified through the `crs_setperm` command.

  The default script uses the user name `orclra`:

```
$CRS_HOME/bin/crs_setperm $APPNAME -o orclra
$CRS_HOME/bin/crs_setperm $APPNAME -g oinstall
```

- Run the script as `root # /mtk2crs/crs_actions/ra/register_ir.sh`.

**Note:** For Reporting and Analysis deregistration, use the `unregister_ir.sh` that is provided in the Appendix.

# Managing the Cluster

You use Oracle Clusterware to manage the Shared Services and Reporting and Analysis cluster. You can control Oracle Clusterware manually with the Command Line tool `crsctl`. You use

`crsctl` commands to start and stop Oracle Clusterware. Using `crsctl` options, you can perform tasks such as enabling online debugging and dynamically adding, removing, and backing up voting disks.

Put `/vol1/app/11.1.0/crs/bin/` in user's `$PATH` statement.

## Checking the Cluster

Use the command `crs_stat -t -v` to check the cluster status. Running `crs_stat -t -v` provides the status of the cluster resource, including VIPs, Shared Services, and Reporting and Analysis applications.

Perform login checks from a Web browser, using `http://hsscluster:58080/interop/index.jsp`, to ensure that Shared Services runs correctly.

## Monitoring the Cluster

The logical application name registered in the cluster with `register_hss.sh` is `hss_app`, and with `register_ir.sh` it is `irapp`. Resources monitored include Native Directory, Reporting and Analysis services, and Web applications. The following commands are available to for monitoring your Shared Services and Oracle's Hyperion® Reporting and Analysis – System 9 cluster, where `hss_app` and `ir_app` are the application names:

- `# crsctl start crs #`—Starts Oracle Clusterware and related resources on the current node.
- `# crsctl stop crs #`—Stops Oracle Clusterware and related resources on the current node.

  If the other node is running, all applications that are managed by Oracle Clusterware are relocated from the current node to the other node. If the command is launched on all nodes, all applications and services are stopped.
- `$ crsctl check crs #`—Verifies the health of `crs` and related daemons on the node from which you are checking.
- `$ crs_stat hss_app` or `$ crs_stat ir_app`—Starts the process related to the specified application. This causes Oracle Clusterware to call the action programs with the start parameter.

  **Caution!** Start the application only with this command.

  You can start the Oracle Clusterware registered application using the option `-f` (force) when there is relationship between applications (for example, `ir_app` relates to `ravip` and `hss_app` relates to `hssvip`):

  ```
  $ crs_start -all -f
  ```
- `$ crs_stat hss_app -f` or `$ crs_stat ir_app -f`—Calls the action programs (such as `hss931.pl` or `ir931.pl`) with the check parameter. Using the option `-t` or `-v`, you can

query the state of all Oracle Clusterware registered applications to view additional information about their status.

- `$ crs_stop hss_app` or `$ crs_stop ir_app`—Stops the application. This causes Oracle Clusterware to call the action programs with the `stop` parameter.

  **Caution!** Use only this command to stop the application. When you use this command for a specific application, its target status is changed to Offline, and the application stays offline after an Oracle Clusterware startup.

  You can stop the Oracle Clusterware registered application using the option `-f` (force) when there is relationship between applications (for `example, ir_app`, `hss_app` and `hssvip/ravip`).

  **Note:** Oracle Clusterware does not restart the application in case of a failure.

  When you stopping the application using this command, the action program is called with the `stop` parameter, and the related application services are shut down on the server where you are working.

- `$ crs_relocate` *`hss_app`* `-f -c crs2 #`——Stops the specified application and starts it on another node in the cluster.

The `crs_start`, `crs_stop`, and `crs_stat` commands become unavailable when the `crsd` daemon is stopped.

# Oracle Clusterware Backup and Recovery

These Oracle Clusterware components should be backed up so you can recover them in case of failure:

- Voting disk—Manages information about node membership.
- OCR—Manages cluster and configuration information.

Oracle recommends that you back up the voting disk and OCR immediately after the initial installation.

See the "Administering Oracle Clusterware" section in the *Oracle Clusterware Administration and Deployment Guide*, which you can download from http://download.oracle.com/docs/cd/B28359_01/rac.111/b28255/votocr.htm.

# Log Files

For log files location, see the "Troubleshooting" section of the *Oracle Clusterware Administration and Deployment Guide*, which you can download from http://download.oracle.com/docs/cd/B28359_01/rac.111/b28255/troubleshoot.htm.

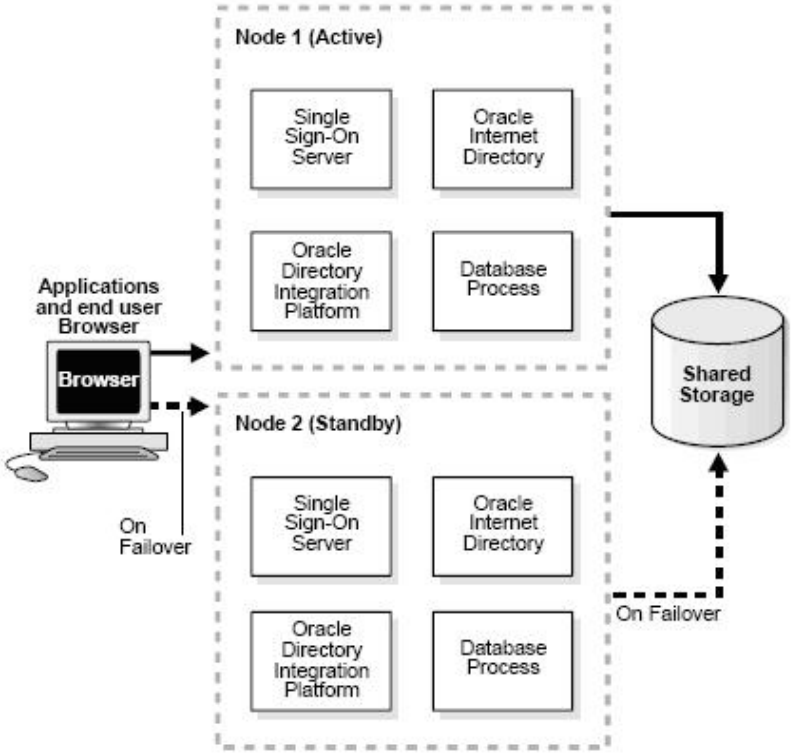
# Oracle Internet Directory Clustering

You can cluster Shared Services OID for high availability. OID is supported as native provider in any active-passive mode that OID supports.

You can use several techniques for making OID highly available. References:

- http://www.oracle.com/technology/products/oid/pdf/oid-largescaledirectory-ha-performance.pdf
- http://www.oracle.com/technology/products/oid/pdf/oid_tuning_configuration_quickreference_01.pdf

Figure 3 Oracle Internet Directory Deployment Using Cold Failover




# Appendix

## Script Template for createviphss.sh

```
#!/bin/sh
APPVIPNAME=hssvip
ACTIONSCRIPT=/vol1/app/11.1.0/crs/bin/usrvip
```

```
ADAPTER=eri0
VIPIP=10.10.12.98
VIPSUBNET=255.255.0.0
START_TIMEOUT=30
STOP_TIMEOUT=30
SCRIPT_TIMEOUT=30
RESTART_ATTEMPTS=2
CRS_HOME=/vol1/app/11.1.0/crs
# profile application VIP
$CRS_HOME/bin/crs_profile -create $APPVIPNAME -t application -a
$ACTIONSCRIPT -o rt=$START_TIMEOUT,pt=$STOP_TIMEOUT,st=$SCRIPT_TIMEOUT,ra=
$RESTART_ATTEMPTS,oi=$ADAPTER,ov=$VIPIP,on=$VIPSUBNET
# register application VIP
$CRS_HOME/bin/crs_register $APPVIPNAME
# application VIP address script must run as the root user
$CRS_HOME/bin/crs_setperm $APPVIPNAME -o root
# enable the orclhss user to run application VIP
$$CRS_HOME/bin/crs_setperm $APPVIPNAME -u user:orclhss:r-x
# see permissions
$CRS_HOME/bin/crs_getperm $APPVIPNAME
# start the resource
# $CRS_HOME/bin/crs_start $APPVIPNAME
```

## Script Template for createvipra.sh

```
#!/bin/sh
APPVIPNAME=ravip
ACTIONSCRIPT=/vol1/app/11.1.0/crs/bin/usrvip
ADAPTER=eri0
VIPIP=10.10.12.97
VIPSUBNET=255.255.0.0
START_TIMEOUT=30
STOP_TIMEOUT=30
SCRIPT_TIMEOUT=30
RESTART_ATTEMPTS=2
CRS_HOME=/vol1/app/11.1.0/crs
# profile application VIP
$CRS_HOME/bin/crs_profile -create $APPVIPNAME -t application –a
$ACTIONSCRIPT -o rt=$START_TIMEOUT,pt=$STOP_TIMEOUT,st=$SCRIPT_TIMEOUT,ra=
$RESTART_ATTEMPTS,oi=$ADAPTER,ov=$VIPIP,on=$VIPSUBNET
# register application VIP
$CRS_HOME/bin/crs_register $APPVIPNAME
# application VIP address script must run as the root user
$CRS_HOME/bin/crs_setperm $APPVIPNAME -o root
# enable the orclra user to run application VIP
$CRS_HOME/bin/crs_setperm $APPVIPNAME -u user:orclra:r-x
# see permissions
$CRS_HOME/bin/crs_getperm $APPVIPNAME
# start the resource
# $CRS_HOME/bin/crs_start $APPVIPNAME
```

## Script Template for register_hss.sh

```
#!/bin/sh
APPNAME=hss_app
```

```
ACTION_SCRIPT=/mtk2crs/crs_actions/hss/hss931.pl
DESCRIPTION=HyperionSharedServices
REQUIRED_RESSOURCES=hssvip
CHECK_INTERVAL=120
FAILOVER_DELAY=120
FAILURE_THRESHOLD=0
FAILURE_INTERVAL=0
START_TIMEOUT=120
STOP_TIMEOUT=120
SCRIPT_TIMEOUT=120
RESTART_ATTEMPTS=2
CRS_HOME=/vol1/app/11.1.0/crs
# profile application
$CRS_HOME/bin/crs_profile -create $APPNAME -t application -a $ACTION_SCRIPT
-d $DESCRIPTION -r $REQUIRED_RESSOURCES -o fd=$FAILOVER_DELAY,ft=
$FAILURE_THRESHOLD,fi=$FAILURE_INTERVAL,rt=$START_TIMEOUT,pt=
$STOP_TIMEOUT,st=$SCRIPT_TIMEOUT,ra=$RESTART_ATTEMPTS
# register application
$CRS_HOME/bin/crs_register $APPNAME
# run application as the orclhss user
$CRS_HOME/bin/crs_setperm $APPNAME -o orclhss
$CRS_HOME/bin/crs_setperm $APPNAME -g oinstall
# see permissions
$CRS_HOME/bin/crs_getperm $APPNAME
# start the resource
# $CRS_HOME/bin/crs_start $APPNAME
```

## Script Template for unregister_hss.sh

```
#!/bin/sh
APPNAME=hss_app
CRS_HOME=/vol1/app/11.1.0/crs
# profile application
$CRS_HOME/bin/crs_profile -delete $APPNAME
$CRS_HOME/bin/crs_unregister $APPNAME
```

## Action Script Template for hss931.pl

**Note:** Update `hsscluster` name and uncomment appropriate AppServer settings.

**Note:** The minimal setting for `$SLEEP_INTERVAL` is `30`. For different application servers, the interval could be increased. If you increase `SLEEP_INTERVAL`, make sure you increase the `SCRIPT_TIMEOUT`, `START_TIMEOUT`, and `STOP_TIMEOUT` in the register script.

```
#!/usr/bin/perl
# Copyright (c) 2002, 2006, Oracle. All rights reserved.
#
#       NAME
#       hss931.pl
#
#       DESCRIPTION
```

```
#       This perl script is the action script for start / stop / check
#       the Shared Services and Openldap in a cold failover configuration.
#
#       Usage: hss931.pl [start | stop | check]

# Environment settings please adapt

my $SCRIPT_PATH     = "/mtk2crs/crs_actions/hss";
my $VER             = "1.0";
my $HSS_URL         = "http://hsscluster:58080/interop/index.jsp";
my $RESP_TIMEOUT    = "40"; #If $HSS_URL does not respond in RESP_TIMEOUT
my $LDAP_HOST       = "hsscluster";
my $LDAP_PORT       = 58089;
my $LOCAL_HOST      = qx(uname -n);

my $home = "/export/home/orclhss"; #save home directory of caller orclhss
my $HYPERION_HOME = "/mtk2sshss/Hyperion";
my $HSS_STARTSCRIPT1 = "$HYPERION_HOME/SharedServices/9.3.1/openLDAP/
startOpenLDAP.sh";
my $HSS_STARTSCRIPT2 = "$HYPERION_HOME/deployments/Tomcat5/bin/
startSharedServices9.sh";
# /mtk2ss/Hyperion/deployments/Tomcat5/bin/startSharedServices9.sh
my $HSS_STOPSCRIPT1 = "$HYPERION_HOME/deployments/Tomcat5/bin/
stopSharedServices9.sh";
my $HSS_STOPSCRIPT2 = "$HYPERION_HOME/SharedServices/9.3.1/openLDAP/
stopOpenLDAP.sh";

#WebLogic 8.1.x - update <HSS_HOME> by real path
#my $HSS_STARTSCRIPT2 = "<HSS_HOME>/AppServer/InstalledApps/WebLogic/8.1/
SharedServices9/bin/startSharedServices9.sh";
#my $HSS_STOPSCRIPT1 = "<HSS_HOME>/AppServer/InstalledApps/WebLogic/8.1/
SharedServices9/bin/stopSharedServices9.sh";

#WebLogic 9
#my $HSS_STARTSCRIPT2 = "$HYPERION_HOME/deployments/WebLogic9/bin/
startSharedServices9.sh";
#my $HSS_STOPSCRIPT1 = "$HYPERION_HOME/deployments/WebLogic9/bin/
stopSharedServices9.sh";

#WebSphere 6
#my $HSS_STARTSCRIPT2 = "$HYPERION_HOME/deployments/WebSphere6/bin/
startSharedServices9.sh";
#my $HSS_STOPSCRIPT1 = "$HYPERION_HOME/deployments/WebSphere6/bin/
stopSharedServices9.sh";

#Oracle
#my $HSS_STARTSCRIPT2 = "<ORAAS_HOME>>/opmn/bin/opmnctl startproc process-
type=<HSS_INSTANCE>
#my $HSS_STOPSCRIPT1 = "<ORAAS_HOME>>/opmn/bin/opmnctl stopproc process-
type=<HSS_INSTANCE>

my $CRS_ACTION_LOG  = "$SCRIPT_PATH/hss931.log";

# depending on the web app server startup speed, you may consider
# increasing this value to 60 secs or
# more. Note that the cumulated startup/stop time must not be greater
# than the clusterware timeouts
```

```
# defined when registering the application START_TIMEOUT=120
# STOP_TIMEOUT=120
# SCRIPT_TIMEOUT=120

my $SLEEP_INTERVAL  = 50;

my $LOG_ENABLED     = "true";
my $DEBUG_ENABLED   = "false";

if ($#ARGV != 0 ) {
    print "usage: hss931.pl [start | stop | check] \n";
    exit;
}

$command = $ARGV[0];

# hss web app and native directory start stop check

# Start Shared Services web application and openldap
if ($command eq "start" ) {
        &log("Starting Shared Services on $LOCAL_HOST");
     &log("home= $home");

        system (" $HSS_STARTSCRIPT1 >/dev/null 2>&1 ");
        sleeping ($SLEEP_INTERVAL, "Starting OpenLDAP on $LOCAL_HOST");

        system (" HOME=$home ; export HOME ; $HSS_STARTSCRIPT2 >/dev/null
2>&1 ");
        sleeping ($SLEEP_INTERVAL, "Starting HSS Web application on
$LOCAL_HOST");
        exit 0;
}

# Stop Shared Services
if ($command eq "stop" ) {
        &log(" Stopping Shared Services on $LOCAL_HOST");
    &log("home= $home");

        system (" HOME=$home ; export HOME ; $HSS_STOPSCRIPT1 >/dev/null
2>&1 ");

        system (" $HSS_STOPSCRIPT2 >/dev/null 2>&1 ");
        sleeping ($SLEEP_INTERVAL, "Stopping HSS Services on $LOCAL_HOST");

        exit 0;
}

# Check Shared Services web app and openldap
if ($command eq "check" ) {

  if ($DEBUG_ENABLED eq "true") { &log("  Check command for HSS Native
directory"); }
  my $ldap_check=ldap_check();
  if ($DEBUG_ENABLED eq "true") { &log("Result openLDAP:$ldap_check"); }

  if ($DEBUG_ENABLED eq "true") { &log("  Check command for HSS web
application url $HSS_URL"); }
```

```
  my $http_check= http_check();
  if ($DEBUG_ENABLED eq "true") { &log("Result HSS:$http_check"); }

  if ($http_check == 0 && $ldap_check == 0) {
      exit 0; #OK
  } else {
      exit 1;
  }
}

#makes sense only on the node that runs slapd
sub ldap_check {
  my $process = "openLDAP/usr/local/libexec/slapd";
  my $check_proc = qx(ps -aef | grep slapd | grep -v grep | awk '{print
$8}');
  chomp($check_proc);
  if ($DEBUG_ENABLED eq "true") { &log("check_proc: $check_proc"); }

  if ($check_proc =~ /$process/) {
    return 0; #OK
  } else {
    return 1;
  }
}

sub http_check {
# /usr/sfw/bin/wget --delete-after --timeout=20 --tries=1 http://
hsscluster:58080/interop/index.jsp 2>&1 | grep -c "200 OK"
# Check, update wget path

  my $res = qx(/usr/sfw/bin/wget --delete-after --timeout=$RESP_TIMEOUT --
tries=1 $HSS_URL 2>&1 | grep -c "200 OK");
  chomp($res);
  if ($DEBUG_ENABLED eq "true") { &log("how many http 200 OK:$res"); }

  if ($res >= 1) {
    return 0; #OK
  } else {
    return 1;
  }
}

sub log {
  if ($LOG_ENABLED eq "true") {
    open LOGFILE , ">>$CRS_ACTION_LOG";
    my $now=localtime(time);
    print LOGFILE "$now - $_[0]\n";
    close LOGFILE
  }
}

#sleeping procedure sleeps and writes into log file dots...
sub sleeping() {
  my $timeout = $_[0];
  my $action = $_[1];
  my $interval = 30;
```

```
  open LOGFILE , ">>$CRS_ACTION_LOG";
  my $now=localtime(time);
  print LOGFILE "$now - $action";
  close LOGFILE;

  for($i = 1; $i <= $interval; $i++) {
   open LOGFILE , ">>$CRS_ACTION_LOG";
   print LOGFILE ".";
   sleep ($timeout/$interval);
   close LOGFILE;
  }
  open LOGFILE , ">>$CRS_ACTION_LOG";
  print LOGFILE "\n";
  close LOGFILE;
}
```

## Script Template for register_ir.sh

**Note:** Because IR Services requires that Oracle's Hyperion® Shared Services be running, `hss_app`is set as `REQUIRED_RESSOURCES`.

**Note:** The minimal `START/STOP/RESTART TIMEOUT` setting is `180`. For different application servers, it could be increased

```
#!/bin/sh
APPNAME=ir_app
ACTION_SCRIPT=/mtk2crs/crs_actions/ra/ir931.pl
DESCRIPTION=IRServices
REQUIRED_RESSOURCES=ravip
CHECK_INTERVAL=120
FAILOVER_DELAY=120
FAILURE_THRESHOLD=0
FAILURE_INTERVAL=0
START_TIMEOUT=180
STOP_TIMEOUT=180
SCRIPT_TIMEOUT=180
RESTART_ATTEMPTS=2
CRS_HOME=/vol1/app/11.1.0/crs
# profile application
$CRS_HOME/bin/crs_profile -create $APPNAME -t application -a $ACTION_SCRIPT
-d $DESCRIPTION -r $REQUIRED_RESSOURCES -o fd=$FAILOVER_DELAY,ft=
$FAILURE_THRESHOLD,fi=$FAILURE_INTERVAL,rt=$START_TIMEOUT,pt=
$STOP_TIMEOUT,st=$SCRIPT_TIMEOUT,ra=$RESTART_ATTEMPTS
# register application
$CRS_HOME/bin/crs_register $APPNAME
# run application as the orclra user
$CRS_HOME/bin/crs_setperm $APPNAME -o orclra
$CRS_HOME/bin/crs_setperm $APPNAME  -g oinstall
# see permissions
$CRS_HOME/bin/crs_getperm $APPNAME
# start the resource
# $CRS_HOME/bin/crs_start $APPNAME
```

## Script Template for unregister_ir.sh

```
#!/bin/sh
APPNAME=ir_app
CRS_HOME=/vol1/app/11.1.0/crs
# profile application
$CRS_HOME/bin/crs_profile -delete $APPNAME
$CRS_HOME/bin/crs_unregister $APPNAME
```

## Action Script Template for ir931.pl

**Note:** Update `cluster hostname` and uncomment appropriate application server settings inside `check` function.

**Note:** The minimal setting for $SLEEP_INTERVAL is 120. For different application servers, it could be increased

```
#!/usr/bin/perl
# Copyright (c) 2002, 2009, Oracle. All rights reserved.
#
#       NAME
#       ir931.pl
#
#       DESCRIPTION
#       This perl script is the action script for start / stop / check
#       the IR&A Services in a cold failover configuration.
#
#       Usage: ir931.pl [start | stop | check]

# Environment settings please adapt

my $SCRIPT_PATH     = "/mtk2crs/crs_actions/ra";
my $VER             = "1.0";
my $WEB_URL       = "http://racluster:19000";
my $WS_URL         = "$WEB_URL/workspace/index.jsp";
my $WA_URL         = "$WEB_URL/WebAnalysis/WebAnalysis.jsp";
my $FR_URL         = "$WEB_URL/hr/status.jsp";
my $RESP_TIMEOUT    = "40"; #If $WS_URL does not respond in RESP_TIMEOUT

my $home = "/export/home/orclra/";   # home directoy of user orclra
my $HYPERION_HOME = "/mtk2ssra/Hyperion";
my $BIPLUS_HOME = "/mtk2ssra/Hyperion/BIPlus";
my $IR_STARTSCRIPT = "$HYPERION_HOME/BIPlus/bin/start_BIPlus.sh";
my $IR_STOPSCRIPT = "$HYPERION_HOME/BIPlus/bin/stop_BIPlus.sh";

my $APACHE_SCRIPT = "$HYPERION_HOME/common/httpServers/Apache/2.0.52/bin/
apachectl";

my $CRS_ACTION_LOG  = "$SCRIPT_PATH/ir931.log";

my $LOCAL_HOST         = qx(uname -n);
```

```
my $SLEEP_INTERVAL  = 120;
my $LOG_ENABLED     = "true";
my $DEBUG_ENABLED   = "false";


if ($#ARGV != 0 ) {
    print "usage: ir931.pl [start | stop | check] \n";
    exit;
}

$command = $ARGV[0];

# IR&A services and web app start stop check

# Start IR&A services
if ($command eq "start" ) {
      &log("");
      &log("---------------------    Begin START
---------------------------------");


        system (" HOME=$home ; export HOME ; $APACHE_SCRIPT start ;
$IR_STARTSCRIPT>/dev/null 2>&1 ");
        sleeping ($SLEEP_INTERVAL, "Starting IR&A Services on
$LOCAL_HOST");
    &log("---------------------    End START
---------------------------------");
        exit 0;
}

# Stop IR&A services
if ($command eq "stop" ) {
    &log("");
    &log("---------------------    Begin STOP
---------------------------------");
        system (" HOME=$home ; export HOME ; $APACHE_SCRIPT stop ;
$IR_STOPSCRIPT >/dev/null 2>&1 ");
        sleeping ($SLEEP_INTERVAL, "Stopping IR&A Services on
$LOCAL_HOST");

    my $res = 0;
    &log("---------------------    End STOP
---------------------------------");
        exit $res;
}

# Check IR&A Services
if ($command eq "check" ) {

  if ($DEBUG_ENABLED eq "true") {

  &log("");
  &log("---------------------    Begin CHECK
---------------------------------");
}
  my $res = 0;
```

```
  $res |= process_check("Common Services", "$BIPLUS_HOME/bin/run/
base_commonservices.pid ");

  $res |= process_check("IntelligenceService", "$BIPLUS_HOME/bin/run/
ir_process_IntelligenceService.pid");
  $res |= process_check("DAS", "$BIPLUS_HOME/bin/run/
ir_process_DataAccessService.pid");

#modify the pid based on your clustername
  $res |= process_check("IRM", "$BIPLUS_HOME/bin/run/
ir_process_IRM1_racluster.pid ");



  $res |= process_check("FR CommServer", "$BIPLUS_HOME/bin/run/
fr_commserver.pid ");
  $res |= process_check("FR RepServer", "$BIPLUS_HOME/bin/run/
fr_repserver.pid ");
  $res |= process_check("FR SchedServer", "$BIPLUS_HOME/bin/run/
fr_schedserver.pid ");

  #Tomcat
  $res |= process_check("WS", "$HYPERION_HOME/deployments/Tomcat5/
Workspace/temp/catalina.pid ");
  $res |= process_check("WA", "$HYPERION_HOME/deployments/Tomcat5/
WebAnalysis/temp/catalina.pid ");
  $res |= process_check("FR", "$HYPERION_HOME/deployments/Tomcat5/
FinancialReporting/temp/catalina.pid ");

  #WebSphere
  #$res |= process_check("WS", "$HYPERION_HOME/deployments/WebSphere6/
profile/logs/Workspace/Workspace.pid ");
  #$res |= process_check("WA", "$HYPERION_HOME/deployments/WebSphere6/
profile/logs/WebAnalysis/WebAnalysis.pid ");
  #$res |= process_check("FR", "$HYPERION_HOME/deployments/WebSphere6/
profile/logs/FinancialReporting/FinancialReporting.pid ");

  #Oracle and WebLogic: only http check is available


  $res |= http_check("$WS_URL");
  $res |= http_check("$WA_URL");
  $res |= http_check("$FR_URL");

  if ($DEBUG_ENABLED eq "true") {

  &log("return value: $res");

  &log("----------------------    End CHECK
---------------------------------");
}

  exit $res;
}

sub process_check() {
   my $process = $_[0];
```

```
  my $pid_file = $_[1];
  my $process_name = $_[2];
  my $pid = "";


  if ($pid_file) {
    $pid = `cat $pid_file | tr -d '\n'`;
  if ($DEBUG_ENABLED eq "true") { &log($process . ": process pid is: " .
$pid); }

} else {
    $pid = $process_name;
  if ($DEBUG_ENABLED eq "true") { &log($process . ": process name is: " .
$pid);}

}

  my $check_proc = qx(ps -aef | grep $pid | grep -v grep | awk '{print
$8}');
  chomp($check_proc);

  if ($check_proc) {
   if ($DEBUG_ENABLED eq "true") {  &log($process . " is running..."); }
    return 0;
  } else {
      if ($DEBUG_ENABLED eq "true") {  &log($process . " is
stopped..."); }
    return 1;
  }
}


sub http_check() {
  &log("");
  my $url = $_[0];
#  &log($url);

  my $res = qx(/usr/sfw/bin/wget --delete-after --timeout=$RESP_TIMEOUT --
tries=1 $url 2>&1 | grep -c "200 OK");
  chomp($res);
#  &log("http number of 200 OK:$res");


  if ($res >= 1) {
    &log($url . " is running...");
    return 0;
  } else {
    &log($url . " is stopped...");
    return 1;
  }
}

sub log {
  if ($LOG_ENABLED eq "true") {
    open LOGFILE , ">>$CRS_ACTION_LOG";
    my $now=localtime(time);
    print LOGFILE "$now - $_[0]\n";
```

```
    close LOGFILE;
  }
}

#sleeping procedure sleeps and writes into log file dots...
sub sleeping() {
  my $timeout = $_[0];
  my $action = $_[1];
  my $interval = 30;

  open LOGFILE , ">>$CRS_ACTION_LOG";
  my $now=localtime(time);
  print LOGFILE "$now - $action";
  close LOGFILE;

  for($i = 1; $i <= $interval; $i++) {
   open LOGFILE , ">>$CRS_ACTION_LOG";
   print LOGFILE ".";
   sleep ($timeout/$interval);
   close LOGFILE;
  }
  open LOGFILE , ">>$CRS_ACTION_LOG";
  print LOGFILE "\n";
  close LOGFILE;
}
```

# Additional Information

*Oracle® Clusterware Administration and Deployment Guide*

11g Release 1 (11.1)

B28255-04

August 2008

http://download.oracle.com/docs/cd/B28359_01/rac.111/b28255.pdf

## COPYRIGHT NOTICE



Authors: Eric Belmon, Thierry Dubois, Cheryl Morrison